Basic WebSphere Application Server V5.0 and V5.1 wsadmin Programmers Guide

Examples for the WebSphere® Application Server V5.x wsadmin tool are included in the information center, Redbooks and technotes. Many clients want to advance their skills, doing more than making minor modifications to these examples. A common goal for clients is to use wsadmin to create commands that duplicate the functionality of the administrative console.

The building blocks in this document help you build the skills necessary to correctly formulate commands that administer a WebSphere Application Server V5.x environment. This document explains how simple techniques that use wsadmin commands, and other available documentation, can help you reach your administrative design goals.

Basic WebSphere Application Server V5.x wsadmin
Programmers Guide

Index

1. Purpose

Many examples for the WebSphere Application Server V5.x wsadmin tool are provided in the information center, Redbooks or technotes. Many clients want to advance their skills beyond the making minor enhancements to examples, without calling IBM® Support.

The final goal for the client is the ability to create commands to mimic the administrative console and beyond. The *Basic WebSphere Application Server V5.x Wsadmin Programmers Guide* advances the client skills by pointing out important documentation that IBM provides, and the simple techniques needed to correctly formulate any command to administer a WebSphere Application Server V5.x. This document shows how some public documentation and some simple techniques that use wsadmin commands, can reach the final administrative design goals.

2. Official Product Documentation

WebSphere Application Server V5.x Information Center:
V5.0: http://publib.boulder.ibm.com/infocenter/wasinfo/index.jsp
V5.1: http://publib.boulder.ibm.com/infocenter/ws51help/index.jsp

The WebSphere Application Server Information Center contains many examples for wsadmin commands, the migration commands from WSCP to wsadmin, and information about WebSphere Application Server MBeans, JavaDoc.

The recommend starting point for new wsadmin users is the information center's Contents view, where you can expand your WebSphere Application Server editions to find **All topics by feature**. Inside this view is the **System administration** section. Underneath System administration is **Scripting**, where new users can learn how to use wsadmin with the information center.

More advance users can employ good search strings to locate information directly.

One example of a generic search string is **Example wsadmin**; a more specific wsadmin search uses the specific command, for example, **installInteractive**. The search results show all the examples and descriptions that IBM has created. This is a powerful source, which IBM keeps up-to-date with current information.

WebSphere Application Server Support Web site:
http://www.ibm.com/software/webservers/appserv/was/support/
The IBM Redbooks, white papers, technotes and fixes for WebSphere Application Server are located on this Web site. The WebSphere Application Server Support Web site provides a variety of information to help you use and fix WebSphere Application Server.

The IBM Redbooks contain cook book examples for wsadmin; for example, the *IBM WebSphere Application Server 5.0 System Management and Configuration Redbook* contains "Command Line Administration and Scripting" in Chapter 22.

Technotes document specific examples that many customers run into that are not documented or that correct documented commands and examples from the Redbooks or other sources.

WebSphere developerWorks :
http://www.ibm.com/developerworks/websphere/
developerWorks is a Web site where advanced technical papers written by the architects and developers of WebSphere products explain the design and usage of the products.

3. Wsadmin Command Overview

Three major categories, or objects, comprise the WebSphere Application Server administration commands: AdminApp, AdminConfig and AdminControl. The most helpful command to use when lost in wsadmin is the help command.

3.1. Help

The help command is useful in learning about how to properly work with the program. The wsadmin help shows basic information about each command. Wsadmin describes the help command as follows:

```
wsadmin>$Help help
WASX7028I: The Help object has two purposes:

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

```
Second, provide a means to obtain interface information about MBeans running in the
system.  For this purpose, a variety of commands are available to get information
about the operations, attributes, and other interface information about particular
MBeans.

The following commands are supported by Help; more detailed information about each of
these commands is available by using the "help" command of Help and supplying the
name of the command as an argument.

attributes         given an MBean, returns help for attributes
...
wsadmin            returns general help text for the wsadmin script launcher
message            given a message id, returns explanation and user action message
```

3.2. AdminApp

wsadmin help describes the AdminApp commands as follows:

```
wsadmin>$AdminApp help
WASX7095I: The AdminApp object allows application objects to be manipulated -- this
includes installing, uninstalling, editing, and listing.  Most of the commands
supported by AdminApp operate in two modes: the default mode is one in which AdminApp
communicates with the WebSphere Application Server to accomplish its tasks.  A local
mode is also possible, in which no server communication takes place.  The local mode
of operation is invoked by bringing up the scripting client with no server connected
using the command line "-conntype NONE" option or setting the
"com.ibm.ws.scripting.connectionType=NONE" property in the wsadmin.properties.

The following commands are supported by AdminApp; more detailed information about
each of these commands is available by using the "help" command of AdminApp and
supplying the name of the command as an argument.

edit            Edit the properties of an application
editInteractive Edit the properties of an application interactively
...
updateAccessIDs Updates the user/group binding information with accessID
                from user registry for a given application
deleteUserAndGroupEntries
                Deletes all the user/group information for all
                the roles and all the username/password information for RunAs
                roles for a given application.
```

3.3. AdminConfig

The AdminConfig category, or object, is used to manipulate the configuration of WebSphere Application Server. This modifies the *install_root*/config/cells directory because this contains the unique model of your WebSphere Application Server system. Wsadmin help explains AdminConfig as follows:

```
wsadmin>$AdminConfig help
WASX7053I: The AdminConfig object communicates with the Config Service in a WebSphere
Application Server to manipulate configuration data for a WebSphere Application
Server installation.  AdminConfig has commands to list, create, remove, display, and
modify configuration data, as well as commands to display information about
configuration data types.
```

```
Most of the commands supported by AdminConfig operate in two modes:
the default mode is one in which AdminConfig communicates with the WebSphere
Application Server to accomplish its tasks.  A local mode is also possible, in which
no server communication takes place.  The local mode of operation is invoked by
bringing up the scripting client with no server connected using the command line "-
conntype NONE" option or setting the "com.ibm.ws.scripting.connectionType=NONE"
property in the wsadmin.properties.

The following commands are supported by AdminConfig; more detailed information about
each of these commands is available by using the "help" command of AdminConfig and
supplying the name of the command as an argument.

attributes      Show the attributes for a given type
checkin         Check a file into the config repository.
...
types           Show the possible types for configuration
validate        Invokes validation
```

3.4. AdminControl

Wsadmin explains AdminControl as follows:

```
wsadmin>$AdminControl help
WASX7027I: The AdminControl object enables the manipulation of MBeans running in a
WebSphere Application Server process.  The number and type of MBeans available to the
scripting client depends on the server to which the client is connected.  If the
client is connected to a Deployment Manager, then all the MBeans running in the
Deployment Manager are visible, as are all the MBeans running in the Node Agents
connected to this Deployment Manager, and all the MBeans running in the application
servers on those nodes.

The following commands are supported by AdminControl; more detailed information about
each of these commands is available by using the "help" command of AdminControl and
supplying the name of the command as an argument.

Note that many of these commands support two different sets of signatures: one that
accepts and returns strings, and one low-level set that works with JMX objects like
ObjectName and AttributeList. In most situations, the string signatures are likely to
be more useful, but JMX-object signature versions are supplied as well.  Each of
these JMX-object signature commands has "_jmx" appended to the command name. Hence
there is an "invoke" command, as well as a "invoke_jmx" command.

completeObjectName
                Return a String version of an object name given a
                template name
getAttribute_jmx
                Given ObjectName and name of attribute, returns value of
                attribute
...
testConnection  Test the connection to a DataSource object
trace           Set the wsadmin trace specification
```

4. Wsadmin Application Commands, AdminApp

The information center contains an advanced example of installing an EAR file; this example is called "Example: Migrating - Installing an application." Using simple commands in the wsadmin AdminApp category can customize the example for your scripts. The Interactive section below explains how IBM Support creates and verifies complex AdminApp install and edit commands.

4.1. Options

The options command is helpful to use when installing an enterprise application. The command shows all application options generically or for a specific EAR file. The following is the wsadmin help output for options:

```
wsadmin>$AdminApp help options
WASX7098I: Method: options

Arguments: none

Description: Displays the general options available for every application.

Method: options

Arguments: filename

Description: Displays all the options available for installing the application in the
file specified by "filename."
```

4.2. Interactive

If you have difficulty creating an install or edit command, first using the installInteractive and editInteractive commands. These commands walk you through the options showing the default settings and allowing you to change them.

IBM support uses these commands to help create or verify complex install and edit commands by enabling tracing on wsadmin and looking for the addToCommandLine trace entry. If wsadmin tracing is enabled, uncommenting the traceString line in the wsadmin.properties file located in *install_root*/properties directory gives you the exact install or edit command to automate this process.

This example demonstrates how to use the preceding technique to create an install command. The sampleApp.ear needs to be installed where the web modules are mapped to a different virtual host. After enabling wsadmin tracing and using the installInteractive command on the sampleApp.ear, we modify specific option settings to match the problem requirements. The following shows the wsadmin changes:

```
wsadmin>$AdminApp installInteractive
C:/WebSphere50/AppServer/installableApps/sampleApp.ear
...
Web Module:  Default Application
URI:  default_app.war,WEB-INF/web.xml
Virtual Host:  [default_host]: host1
Setting "Virtual Host" to "host1"
Web Module:  Examples Application
URI:  examples.war,WEB-INF/web.xml
Virtual Host:  [default_host]: host2
Setting "Virtual Host" to "host2"
...
ADMA5013I: Application Sample Application installed successfully.
```

We do not need to save this configuration change because the goal is to determine the actual install command. The wsadmin.traceout file, located in *install_root*/logs directory, is searching for addToCommandLine. The following is found:

```
AdminAppClien < addToCommandLine: install
C:/WebSphere50/AppServer/installableApps/sampleApp.ear {  -MapWebModToVH {{"Default
Application" default_app.war,WEB-INF/web.xml host1} {"Examples Application"
examples.war,WEB-INF/web.xml host2}}
```

The important part of the trace is everything starting from install to the end of the line. Start with $AdminApp, then add the information from the trace, which results in the exact wsadmin install command.

Note: Based on the WebSphere Application Server level running the addition of a closing bracket } to the end of command might be required for this to correctly work. This is how to create or verify the **$AdminApp install** or **$AdminApp edit** commands quickly, easily and correctly.

5. Wsadmin Configuration Commands, AdminConfig

The AdminConfig category is the command searched for most often. This section helps increase your understanding and vocabulary for the AdminConfig command. The basic subject areas are explained to make sure you have a strong foundation before jumping into advance configuration commands. The next section reviews techniques on how to create or improve your wsadmin commands.

5.1. Basic AdminConfig Subjects

There are four basics for the creation or adaptation of the AdminConfig command: types, attributes, required attributes, and configuration Ids.

You probably have an idea of which objects you want to configure in wsadmin. Each configuration object, like a Java object, belongs to a WebSphere Application Server type. The types are abstractions of configuration objects. A type is defined by its attributes. Configuration objects have the real objects used by WebSphere Application Server and have the attributes set to some user-defined values. So when you want to create a configuration object, you must find out the type of the object to find out what attributes you must set.

5.1.1. Types

All objects that are shown in the administrative console or read in documentation about WebSphere Application Server can be mapped to a WebSphere Application Server type. The AdminConfig command, $AdminConfig types, outputs all WebSphere Application Server types. To understand how to use this command, refer to this output from wsadmin:

```
wsadmin>$AdminConfig help types
WASX7068I: Method: types

Arguments: type

Description: Displays all the possible top-level configuration object types.
```

The help for this command states that there is a single argument; however, this is not correct. There are no arguments to execute the types command. To determine all the possible WebSphere Application Server types to configure in wsadmin, use the $AdminConfig types command. The following is an example output of this command:

```
wsadmin>$AdminConfig types
AdminService
Agent
AllAuthenticatedUsersExt
Application
```

```
...
WebContainer
WebModuleConfig
WebModuleDeployment
WorkloadManagementServer
```

5.1.2. Attributes

After the WebSphere Application Server type is found, or if more information is needed to better understand the appropriate type, seeing all attributes can help. There is a command in wsadmin to display all attributes for any WebSphere Application Server type. From wsadmin help, this output shows how to use the attributes command:

```
wsadmin>$AdminConfig help attributes
WASX7061I: Method: attributes

Arguments: type

Description: Displays all the possible attributes contained by an object of type
"type."  The attribute types are also displayed; when the attribute represents a
reference to another object, the type of the attribute has a suffix of "@."  When the
attribute represents a collection of objects, the type is listed with a suffix of
"*."  If the type represents a base type, possible subtypes are listed after the base
type in parenthesis.  If the type is an enumeration, it is listed as "ENUM," followed
by the possible values in parentheses.

Note when a suffix is added an object it represents important information to better
use and configure this type.
```

5.1.3. Required Attributes

With any object, there are required and optional configuration attributes. To see which attributes are required when creating a WebSphere Application Server type, use **required**:

```
wsadmin>$AdminConfig help required
WASX7360I: Method: required

Arguments: type

Description: Displays the required attributes contained by an object of type "type".
```

5.1.4. Configuration IDs

A configuration ID is a pointer to the specific object that you want to create, edit or view in the configuration. Wsadmin represents these IDs with the display name first, followed by the configuration data ID in parenthesis. Not all types have a display name.

Following is an example of a wsadmin configuration ID:
server1(cells/IBMNetwork/nodes/IBM/servers/server1:server.xml#Server_1).
The exact outline for the config ID is ***display name*(config path:filename#id)**. This directly correlates to the configuration xml files in WebSphere Application Server located in the *install_root*/config directory. The config path and the filename are the specific location, the exact path based off the config directory, and file where the pointer associates to the WebSphere Application Server configuration. The id and display name are actually the name= and xmi:id= xml parameters found inside the referenced file. From the initial config ID example, this represents the server.xml file for server1 located in *install_root*/config/cells/IBMNetwork/nodes/IBM/servers/server1 directory where the following entry is found inside this file:
<process:Server ...*Other information*... xmi:id="Server_1" name="server1">

There are many commands in wsadmin to retrieve a config ID. The most popular way is the $AdminConfig getid command. From wsadmin help you find the following information about $AdminConfig getid command:

```
wsadmin>$AdminConfig help getid
WASX7085I: Method: getid

Arguments: containment path

Description: Returns the config id for an object described by the given containment
path -- for example, /Node:myNode/Server:s1/JDBCProvider:jdbc1/
```

Only a *containment path* argument is needed to accurately issue this command. There are no other arguments for this command. The containment path is a string describing the type and name of objects to precisely find a config ID from the configuration directory.

The type is the WebSphere Application Server defined configuration types, learned previously in the Basic AdminConfig subjects.

The name is the display name that is system specific. The containment path string is formed by the user placing a colon (**:**) between each type and name noting to wsadmin when the type ends and the name begins. The forward slash (/) is used by wsadmin to note when the type:name object begins and ends. If the name is not inputted into the type:name object, a wildcard search is done, which returns all config IDs with that type. Multiple type:name objects can be chained together to narrow a search for a config ID allowing less user interaction. A containment path string can start at any specific type; only the direct child can be the next type:name object. If the next type:name object in the string is not the direct child, no config ID is ever returned.

On a side note, wsadmin notes a space as an end to an argument and a valid display name can have multiple spaces. When you use the JACL scripting language, you can use brackets (**{**) and (**}**) around the whole containment path when spaces are in the string. This stops wsadmin from confusing the containment path as multiple arguments, especially since the getid command accepts only one argument.

The wsadmin help command for getid showed the following example for a containment object: /Node:myNode/Server:s1/JDBCProvider:jdbc1/. Reviewing the string, the Node, Server and JDBCProvider show that they are WebSphere Application Server config types, while the myNode, s1, and jdbc1 are specific to the WebSphere Application Server system configuration.

Another semi-popular and easy-to-use command returning a config ID is the $AdminConfig list command. The advantage of using the list command is the knowledge that a direct parent type is not required to retrieve the config ID; that is, the getid command. The list command does wildcard searching where a scope can be input to narrow the search, starting at the scope object and looking only at the children below this object. The output from wsadmin help is shown below:

```
wsadmin>$AdminConfig help list
WASX7056I: Method: list

Arguments: type

Description: Lists all the configuration objects of the type named by "type."

Method: list

Arguments: type, scope
```

```
Description: Lists all the configuration objects of the type named by "type" within
the scope of the configuration object named by "scope."
```

The getid and list commands can return no config ID, a single config ID or multiple config IDs base on the arguments input. There are a few problems that might be seen when searching for the config IDs. If the command does not return anything, these are some possible reasons:

The type is not a correct WebSphere Application Server type. Verify the types subsection.

The name or scope is not correctly spelled. WebSphere Application Server is case sensitive.

The object does not exist. Do a wildcard search, replace the last name with nothing on getid command, for example, /Node:myNode/Server:s1/JDBCProvider:/, or in the list command remove or increase the scope.

If all fails, there might be something wrong with wsadmin. Try other commands to retrieve the config ID for a workaround on the issue; for example, showAttribute.

5.2. Techniques for Formulating Administrative Configuration Commands

This section walks wsadmin users though techniques to create or expand the number of useful configuration commands. Below are two scenarios showing step-by-step how you can use the preceding basic commands, and some creative thinking to create any configuration commands. The first scenario covers an IBM Information Center configuration example, where each command is analyzed to show how easy the IBM examples are to write. Formulating a command without any example to help guide you is shown in the second scenario.

5.2.1. Scenario 1

This scenario is an analysis of how IBM created Information Center examples for configuring WebSphere Application Server titled **Example: Configuring the Java virtual machine using wsadmin**. Following is the exact output from the Information Center:

Example: Configuring the Java virtual machine using wsadmin

Document Information: An example modifying the Java virtual machine (JVM) of a server to turn on debug follows:

Identify the server and assign it to the server1 variable.

```
set server1 [$AdminConfig getid /Cell:mycell/Node:mynode/Server:server1/]
```

Example output:

```
server1(cells/mycell/nodes/mynode/servers/server1:server.xml#Server_1)
```

Identify the JVM belonging to this server and assign it to the JVM variable.

```
set jvm [$AdminConfig list JavaVirtualMachine $server1]
```

Example output:

```
(cells/mycell/nodes/mynode/servers/server1:server.xml#JavaVirtualMachine_1)
```

Modify the JVM to turn on debug.

```
$AdminConfig modify $jvm {{debugMode true} {debugArgs "-Djava.compiler=NONE -Xdebug -
Xnoagent -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=7777"}}
```

Save the changes with the following command:

```
$AdminConfig save
```

The example is a four-step process to complete the task. The example is interested in changing the JVM properties on server1. The first two steps are focused on retrieving the config ID to the object that requires modifications. Instead of using two commands to get the required config ID, this can be done in one step by using $AdminConfig getid /Server:server1/JavaProcessDef:/JavaVirtualMachine:/.
The single command is advanced for the containment path, because you must understand that the JavaProcessDef type is the parent of a JavaVirtualMachine type. Therefore, the config IDs can be fetched in several ways; based on your skill and comfort level, you can decide your initial steps.

The second phase of this example is modifying the JVM debug options. Because the example is modifying only the JVM debug options, the attributes command executed on this type shows all the parameters possible for a JavaVirtualMachine type. The output from wsadmin is below:

```
wsadmin>$AdminConfig attributes JavaVirtualMachine
"bootClasspath String*"
"classpath String*"
"debugArgs String"
"debugMode boolean"
"disableJIT boolean"
"executableJarFileName String"
"genericJvmArguments String"
"hprofArguments String"
"initialHeapSize int"
"maximumHeapSize int"
"osName String"
"runHProf boolean"
"systemProperties Property(TypedProperty)*"
"verboseModeClass boolean"
"verboseModeGarbageCollection boolean"
"verboseModeJNI boolean"
```

In the example, the interest is only in editing the debugMode and debugArgs options for the JavaVirtualMachine. Now the command can be properly constructed using the modify command.

```
wsadmin>$AdminConfig help modify
WASX7058I: Method: modify

Arguments: config id, attributes

Description: Changes the attributes specified by "attributes" for the configuration
object named by "config id."
```

Preceding is the output from wsadmin where the modify command accepts only two arguments, config ID and attributes. Remember, wsadmin using the JACL language understands that each argument ends when a space is found, and the use of brackets around an object that requires a space bypasses the logic. Because each attribute requires spaces, the use of brackets around the object pair are required, and then around the collection of attributes to represent the single attributes argument for the modify command.

The last step to all wsadmin configuration commands is a save to the WebSphere Application Server master configuration.

5.2.2. Scenario 2

Scenario 2 is the creation of wsadmin commands where there is no example showing the exact or similar commands. The most popular wsadmin commands are administrative configuration done in the administrative

console. When there are no examples to follow, you must use the basic wsadmin commands to collect the information necessary to formulate the commands, and you must do some educated researching.

The researching requires that you understand how the WebSphere Application Server types map to objects that need to be created, modified or deleted. Or the research can be as easy as configuring using the administrative console, and reviewing the configuration changes in the *install_root*/config directory. This scenario demonstrates how to use wsadmin to add a console user.

The administrative console can add console users for system administration without difficulty because you can see where to click, and you can see all the parameters that are required for. Expand System Administration, then choose Console Users and click Add to create a new user. A new page displays places for the User name and the Role(s) to map to this user's administration level.

Following is a technique used to determine how to add a Console User with wsadmin. The technique is broken down to three steps: discovering the type, understanding the final command and forming the final command.

5.2.2.1. Discovering the Type

When we use the $AdminConfig types command, we can make an educated guess at which type will create a new Console User. There is no type that has the word console in the name, but two types contain the word user: UserExt and UserRegistry. The $AdminConfig attributes *type* command displays all the attributes for both types. Following is the output from wsadmin:

```
wsadmin>$AdminConfig attributes UserRegistry
"ignoreCase boolean"
"limit int"
"properties Property(TypedProperty)*"
"realm String"
"serverId String"
"serverPassword String"

wsadmin>$AdminConfig attributes UserExt
"accessId String"
"name String"
```

If you understand that a Console User is a global setting, and if you review all the attributes for each type, you can determine that the UserExt is the correct type. Why would a global object, Console User, need attributes such as serverId or serverPassword? Note that the administrative console does not show these attributes.

Use the administrative console to create a Console User, then save the new settings and see what is modified in the *install_root*/config directory. This research shows that the admin-authz.xml file, located in *install_root*/config/cells/<cell name> directory was modified by adding the following xml tag:

```
<users xmi:id="UserExt_1" name="IBM"/>
```

The xmi:id= shows the type of object that was created: UserExt.

Both educated and direct research points to the same conclusion: UserExt is the type to create a new Console User in wsadmin.

5.2.2.2. Understanding the Final Command

This section explains how to create a new UserExt in wsadmin. $AdminConfig help has a method called create; running the help command on this method sheds light on how to operate. Following is example wsadmin output:

```
wsadmin>$AdminConfig help create
```

```
WASX7054I: Method: create

Arguments: type, parent, attributes

Description: Create a configuration object of the type named by "type," the parent
named by "parent," using the attributes supplied by "attributes."

Method: create

Arguments: type, parent, attributes, parent attribute name,

Description: Create a configuration object of the type named by "type," the parent
named by "parent," using the attributes supplied by "attributes" and the attribute
name in the parent given by "parent attribute name"
```

We know the type is UserExt, but we might not know the qualified attributes or direct parent of this type:

The attributes and required commands can help determine the proper attributes for this object. These show all the possible attributes for UserExt and what is required when creating a new UserExt.

To determine the parent is not as easy. We have to take an educated guess or review the actual configuration files to determine the parent of the UserExt object. Below is an example of how this can be done:

Review admin-authz.xml. This file contains the Console user information; the following xml tag is located at the top:

```
<rolebasedauthz:AuthorizationTableExt xmi:version="2.0" ...
```

In the types list, we see that AuthorizationTableExt is a WebSphere Application Server type. Examine the attributes on this object to determine who is the direct parent of the UserExt object. Below is the attributes output:

```
wsadmin>$AdminConfig attributes AuthorizationTableExt
"authorizations RoleAssignmentExt*"
"context String"
"fileName String"
"roles SecurityRoleExt*"
```

The asterisk (*) at the end of an object means that the object is a collection of objects, and contains multiple attributes to create this object. Following is the RoleAssignmentExt type:

```
wsadmin>$AdminConfig attributes RoleAssignmentExt
"groups GroupExt*"
"role SecurityRoleExt@"
"specialSubjects SpecialSubjectExt(EveryoneExt, AllAuthenticatedUsersExt,
ServerExt)*"
"users UserExt*"
```

In the preceding output, the UserExt is an attribute of the RoleAssignmentExt; therefore, the direct parent of UserExt is RoleAssignmentExt. All the mandatory arguments for create are retrieved.

5.2.2.3. Forming the Final Command

This subsection focuses on assembling the commands to create the new console user in wsadmin.

STEP 1
Getting the parent config ID for the UserExt is done with the following command; a wildcard search on the RoleAssignmentExt type is used because the display name is unknown:

```
wsadmin>set c [$AdminConfig getid /AuthorizationTableExt:admin-
authz.xml/RoleAssignmentExt:/]
(cells/IBMNetwork:admin-authz.xml#RoleAssignmentExt_1)
(cells/IBMNetwork:admin-authz.xml#RoleAssignmentExt_2)
(cells/IBMNetwork:admin-authz.xml#RoleAssignmentExt_3)
(cells/IBMNetwork:admin-authz.xml#RoleAssignmentExt_4)
```

STEP 2
Multiple config IDs are returned for this example because the user needs to pick the level of security requested:

administrator - RoleAssignmentExt_1 (index 0)
operator - RoleAssignmentExt_2 (index 1)
configurator - RoleAssignmentExt_3 (index 2)
monitor - RoleAssignmentExt_4 (index 3)

Use a JACL command to select the correct parent config ID and map the security level for the new UserExt.

```
wsadmin>set d [lindex $c 0]
(cells/FatherNetwork:admin-authz.xml#RoleAssignmentExt_1)
```

Step two is required because the config IDs do not each have an associated display name that allows a devoted containment path.

STEP 3
The create command is formed for the new UserExt.

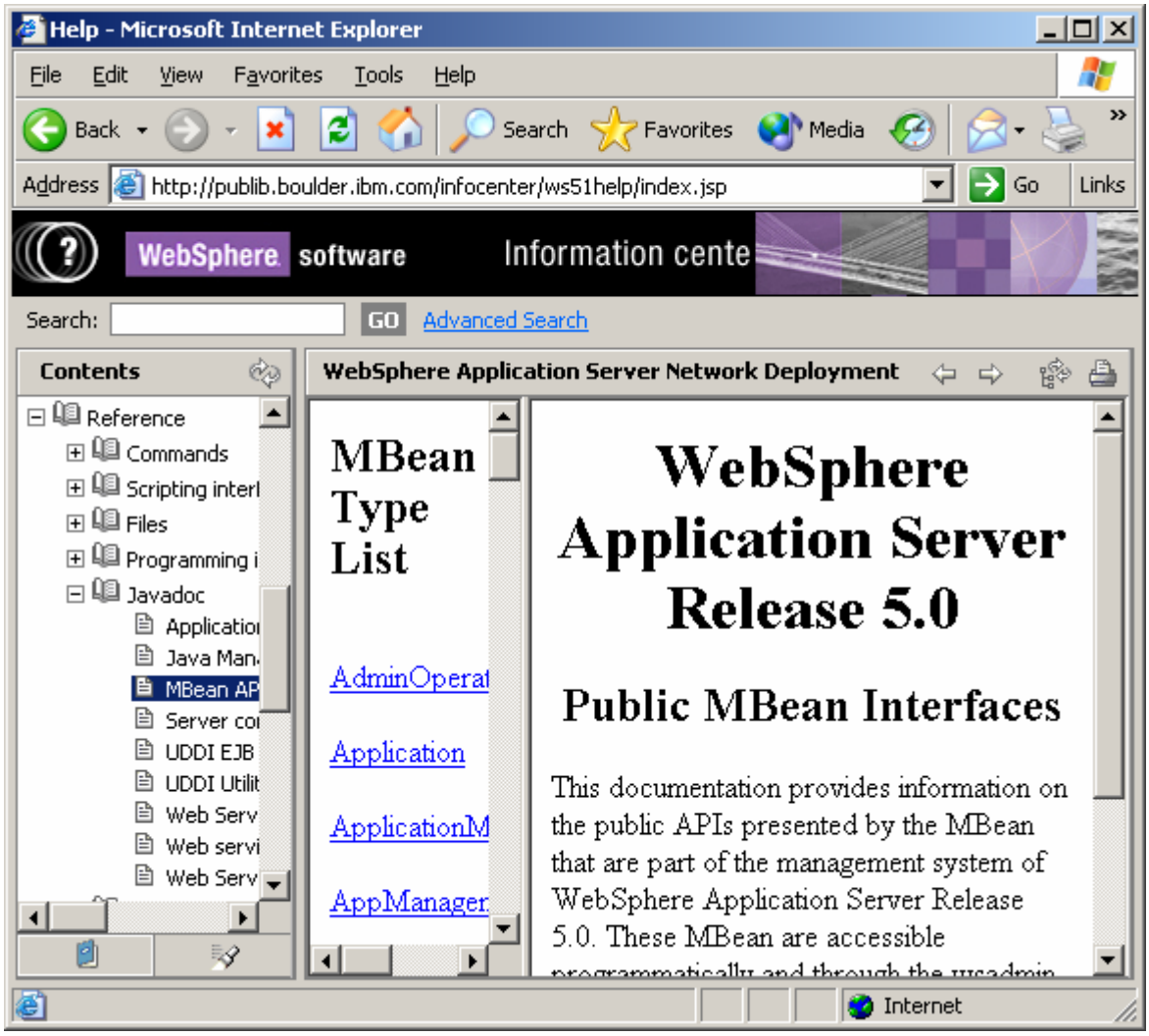
```
wsadmin>$AdminConfig create UserExt $d {{name James}}
James(cells/IBMNetwork:admin-authz.xml#UserExt_1074722017464)
```

STEP 4
Save all changes done to the configuration to the master WebSphere Application Server configuration.

```
wsadmin>$AdminConfig save
```

6. Wsadmin Control Commands, AdminControl
The AdminControl category enables the manipulation of MBeans running in a WebSphere Application Server process. If the MBean is not reachable to the MBean server to which wsadmin is connected, wsadmin cannot manipulate that MBean.

Built into wsadmin is a useful command, **$Help operation *$objectname***, where *$objectname* is the ObjectName of an mbean. This command provides a list of available operations that the specific mbean provides. The Information Center describes all the functions that be executed on all MBeans in WebSphere Application Server. Search for **MBean JavaDoc** to display the following page:



There are many examples in the information center and in various white papers that explain this category in detail.

7. Conclusion

After reading this information, you should be able to create or research any wsadmin command. If you have any problems with a command that is not working, make sure you have the answers to the following questions before contacting IBM:

a. Did you review the WebSphere Application Server Support Web site to see if a defect or technote discusses this issue?

b. Is wsadmin running the most current code version? If not, upgrade to the current fix pack and cumulative fix, then test again to make sure this is not already amended.

c. Follow directions in MustGather: Readme First to capture data for problem determination, then open a PMR.