



*WebSphere Application
Server V8.5 Scripting and
Automation*

(Course code WA680 / VA680)

Student Notebook

ERC 1.0

Authorized

IBM | Training

WebSphere Education

Trademarks

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AFS™	AIX®	DB™
DB2®	Express®	HACMP™
IMS™	MVS™	RACF®
Rational®	RDN®	Tivoli®
WebSphere®	WPM®	z/OS®
zSeries®		

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

May 2013 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Trademarks	xvii
Course description	xxi
Agenda	xxiii
Unit 1. Introduction to scripting	1-1
Unit objectives	1-2
The administrator role	1-3
Why automate administrative tasks? (1 of 2)	1-4
Why automate administrative tasks? (2 of 2)	1-5
Benefits of automation	1-6
WebSphere automation capabilities (1 of 2)	1-7
WebSphere automation capabilities (2 of 2)	1-8
WebSphere automation capabilities: What to use when (1 of 2)	1-9
WebSphere automation: What to use when (2 of)	1-10
What is scripting?	1-11
Types of administrative tasks that should be automated	1-12
Common administrative tasks that should be automated	1-13
When not to use scripting	1-14
Unit summary	1-15
Checkpoint questions	1-16
Checkpoint answers	1-17
Unit 2. WebSphere Application Server administrative concepts and architecture 2-1	
Unit objectives	2-2
Topics	2-3
2.1. Architecture run time	2-5
Architecture run time	2-6
Version 8.5 packaging	2-7
WebSphere Application Server basics	2-8
WebSphere architecture run time (1 of 10)	2-9
WebSphere architecture run time (2 of 10)	2-11
WebSphere architecture run time (3 of 10)	2-12
WebSphere architecture run time (4 of 10)	2-13
WebSphere architecture run time (5 of 10)	2-14
WebSphere architecture run time (6 of 10)	2-15
WebSphere architecture run time (7 of 10)	2-16
WebSphere architecture run time (8 of 10)	2-17
WebSphere architecture run time (9 of 10)	2-18
WebSphere architecture run time (10 of 10)	2-19
2.2. Architecture administration	2-21
Architecture administration	2-22
WebSphere architecture administration (1 of 4)	2-23

WebSphere architecture administration (2 of 4)	2-24
WebSphere architecture administration (3 of 4)	2-25
WebSphere architecture administration (4 of 4)	2-26
2.3. Profiles	2-27
Profiles	2-28
WebSphere profile overview	2-29
WebSphere profile benefits	2-30
Managing profiles	2-31
Profile types	2-32
2.4. Network deployment concepts	2-35
Network deployment concepts	2-36
Network deployment concepts	2-37
Managed versus unmanaged nodes	2-38
Network deployment run time flow	2-39
Administration flow	2-40
File synchronization	2-41
WebSphere Network Deployment profiles	2-42
2.5. Advanced concepts.	2-43
Advanced concepts	2-44
Flexible management	2-45
Centralized Installation Manager (CIM)	2-46
Intelligent run time provisioning	2-47
Edge Components	2-48
Intelligent Management Pack	2-49
Unit summary	2-50
Checkpoint questions (1 of 2)	2-51
Checkpoint questions (2 of 2)	2-52
Checkpoint answers (1 of 2)	2-53
Checkpoint answers (2 of 2)	2-54
Unit 3. WebSphere Application Server scripting facilities.	3-1
Unit objectives	3-2
WebSphere Application Server scripting support	3-3
Prerequisite knowledge for script development	3-4
Assembly and deployment tools	3-5
IBM Assembly and DeployTools for WebSphere Administration	3-6
IBM Assembly and DeployTools tasks	3-7
Features in IBM Assembly and DeployTools for WebSphere Administration (1 of 2)	3-8
Features in IBM Assembly and DeployTools for WebSphere Administration (2 of 2)	3-9
Create, test, and debug Jython scripts	3-10
Start the Jython debugger	3-11
Jython debugger perspective	3-12
Command assistance	3-13
Using command assistance within IBM Assembly and DeployTools	3-14
Script execution: wsadmin	3-15
Script execution: wsadmin administrative objects (1 of 2)	3-16

Script execution: wsadmin administrative objects (2 of 2)	3-17
What is Ant?	3-18
Script execution: ws_ant	3-19
Scripting languages: Jacl and Jython	3-20
WebSphere MBeans	3-21
Unit summary	3-22
Checkpoint questions	3-23
Checkpoint answers	3-24
Unit 4. Using wsadmin	4-1
Unit objectives	4-2
wsadmin versus administration console	4-3
Scripting benefits	4-4
wsadmin	4-5
wsadmin invocation options	4-6
Jython versus Jacl	4-7
Administrative functions that use wsadmin	4-8
Administrative objects in wsadmin	4-9
Starting wsadmin	4-10
Starting wsadmin with security enabled	4-11
wsadmin properties	4-12
Profile scripts	4-13
AdminConfig: Managing configurations	4-14
AdminApp: Managing applications	4-15
AdminControl: Managing running objects	4-16
AdminTask: Accessing administrative functions	4-17
Help within wsadmin	4-18
Administrative command help	4-19
Important points to remember when using wsadmin	4-20
Scripting: Simple script	4-21
Scripting: Looping script	4-22
Jython script library	4-23
How to use the Jython script library	4-24
Configuration repository: The issues	4-25
Properties file based configuration: A solution	4-26
Properties file configuration content	4-27
Properties file configuration commands	4-28
Create, test, and debug Jython scripts	4-29
Start the Jython debugger	4-30
Jython debugger perspective	4-31
Command assistance	4-32
Using command assistance within IBM Assembly and DeployTools	4-33
Unit summary	4-34
Checkpoint questions	4-35
Checkpoint answers	4-36
Exercise 1	4-37
Exercise objectives	4-38

Unit 5. Jython basics	.5-1
Unit objectives	.5-2
What is Jython?	.5-3
Jython Features	.5-4
Jython benefits	.5-5
Basic program elements	.5-6
Example python program	.5-7
Backslash is a special character	.5-8
Three types of reserved words	.5-9
The None object	.5-10
Basic built-in data type categories (1 of 2)	.5-11
Basic built-in data type categories (2 of 2)	.5-12
Order of precedence (1 of 2)	.5-13
Order of precedence (2 of 2)	.5-14
Numeric data types (1 of 2)	.5-15
Numeric data types (2 of 2)	.5-16
Sequence data type: Strings	.5-17
Common string functions and methods	.5-18
Sequence slicing	.5-19
Sequence data type: Lists	.5-20
List methods (1 of 2)	.5-21
List methods (2 of 2)	.5-22
List methods examples	.5-23
List comprehensions	.5-24
Sequence data type: Tuples	.5-25
Mappings: Dictionaries	.5-26
Boolean expressions	.5-27
Simple statements: Assignment	.5-28
The print statement	.5-29
Other simple statements (1 of 2)	.5-30
Compound statements: if elif else	.5-31
The while statement	.5-32
The for statement	.5-33
Example: The for statement	.5-34
Using the range() function in a for loop	.5-35
Exceptions	.5-36
The try statement	.5-37
Forms of the except clause	.5-38
The raise statement	.5-39
User-defined exceptions	.5-40
File input and output: Functions and methods	.5-41
Modules	.5-42
Functions	.5-43
Function argument example	.5-44
The execfile() function	.5-45
Jython standard library	.5-46
Commonly used standard modules	.5-47
The sys module	.5-48

The os module	5-49
os module variable and function examples	5-50
Using Java libraries	5-51
Classes	5-52
Class example	5-53
Unit summary	5-54
Checkpoint questions	5-55
Checkpoint answers	5-56
Unit 6. Jython development by using the IADT.....	6-1
Unit objectives	6-2
IBM Assembly and Deploy Tools for WebSphere Administration	6-3
Jython support	6-4
Selecting a workspace	6-5
Creating Jython projects	6-6
Creating Jython script files	6-7
Perspective and views	6-8
Create, test, and debug Jython scripts	6-9
Jython editor features	6-10
Color preferences	6-11
Defining a target server (1 of 3)	6-12
Defining a target server (2 of 3)	6-13
Defining a target server (3 of 3)	6-14
Script launcher	6-15
Running a script and viewing output in the Console	6-16
Jython debugger	6-17
Start the Jython debugger	6-18
Jython debugger perspective	6-19
Setting breakpoints	6-20
Stepping through breakpoints	6-21
Examining variables	6-23
Administrative console command assistance	6-24
Administrative scripting command preferences	6-25
Command assistance notifications	6-26
Unit summary	6-27
Checkpoint questions	6-28
Checkpoint answers	6-29
Exercise 2	6-30
Exercise objectives	6-31
Unit 7. Administrative object basics: Help and AdminConfig	7-1
Unit objectives	7-2
What are administrative objects?	7-3
Administrative objects: What to use when	7-4
When to use the Help object	7-5
The Help object	7-6
Methods for getting help on the administrative objects	7-7
Method for getting help on WebSphere messages	7-8

Method for getting help on starting wsadmin	7-9
When to use the AdminConfig object	7-10
The AdminConfig object	7-11
How to use the AdminConfig object	7-12
Identifying a configuration object's type	7-13
Examples: Using the configuration type help methods (1 of 4)	7-14
Examples: Using the configuration type help methods (2 of 4)	7-15
Examples: Using the configuration type help methods (3 of 4)	7-16
Examples: Using the configuration type help methods (4 of 4)	7-17
Attribute types	7-18
Using the WebSphere configuration model documentation	7-19
Getting a configuration ID	7-20
The getid method	7-21
Containment paths	7-22
Example of containment relationships in the configuration files	7-23
Containment path and getid method examples	7-24
The list method	7-25
Querying a configuration object	7-26
show method example	7-27
showall method example	7-28
Creating a configuration object	7-29
The create method	7-30
The createUsingTemplate and listTemplates methods	7-31
Example: Creating a configuration object from a template	7-32
Setting attribute values (1 of 4)	7-33
Setting attribute values (2 of 4)	7-34
Setting attribute values (3 of 4)	7-35
Setting attribute values (4 of 4)	7-36
Modifying a configuration object (1 of 3)	7-37
Modifying a configuration object (2 of 3)	7-38
Modifying a configuration object (3 of 3)	7-39
Deleting a configuration object	7-40
Saving or discarding configuration changes	7-41
AdminConfig method reference (1 of 3)	7-42
AdminConfig method reference (2 of 3)	7-43
AdminConfig method reference (3 of 3)	7-44
Unit summary	7-45
Checkpoint questions	7-46
Checkpoint answers	7-47
Exercise 3	7-48
Exercise objectives	7-49

Unit 8. Administrative object basics: AdminApp	8-1
Unit objectives	8-2
When to use the AdminApp object	8-3
The AdminApp object	8-4
Application management task support	8-5
AdminApp help methods	8-6

Querying application and module names	8-7
Example: Querying application and module names	8-8
Installing an application	8-9
Example: Installing an application in batch mode	8-10
Example: Installing an application in interactive mode (1 of 7)	8-11
Example: Installing an application in interactive mode (2 of 7)	8-12
Example: Installing an application in interactive mode (3 of 7)	8-13
Example: Installing an application in interactive mode (4 of 7)	8-14
Example: Installing an application in interactive mode (5 of 7)	8-15
Example: Installing an application in interactive mode (6 of 7)	8-16
Example: Installing an application in interactive mode (7 of 7)	8-17
AdminApp method options	8-18
Method options syntax	8-19
The options method	8-20
options method examples	8-21
Task options	8-22
taskInfo method example	8-23
Useful task option features: Pattern matching	8-24
Useful task option features: Target manipulation	8-25
Editing an application	8-26
Editing an application in batch mode	8-27
Updating an application	8-28
Examples: Updating an application in batch mode	8-29
Uninstalling an application	8-30
AdminApp method reference (1 of 2)	8-31
AdminApp method reference (2 of 2)	8-32
Unit summary	8-33
Checkpoint questions	8-34
Checkpoint answers	8-35
Exercise 4	8-36
Exercise objectives	8-37
Unit 9. Administrative object basics: AdminControl	9-1
Unit objectives	9-2
When to use the AdminControl object	9-3
The AdminControl object (1 of 2)	9-4
The AdminControl object (2 of 2)	9-5
How to use the AdminControl object	9-6
MBean object names (1 of 2)	9-7
MBean object names (2 of 2)	9-8
Object name templates	9-9
Getting the object name of an MBean (1 of 2)	9-10
Getting the object name of an MBean (2 of 2)	9-11
Examples for queryNames and completeObjectName	9-12
Determining the attributes of an MBean	9-13
Determining the operations of an MBean	9-14
Other useful Help methods for MBeans	9-15
Displaying the value of MBean attributes	9-16

Example: Displaying the value of MBean attributes	9-17
Modifying MBean attributes (1 of 2)	9-18
Modifying MBean attributes (2 of 2)	9-19
Considerations for modifying MBean attributes	9-20
Example: Modifying MBean attributes	9-21
Performing operations on MBeans (1 of 2)	9-22
Performing operations on MBeans (2 of 2)	9-23
Examples: Performing operations on MBeans (1 of 3)	9-24
Examples: Performing operations on MBeans (2 of 3)	9-25
Examples: Performing operations on MBeans (3 of 3)	9-26
Repository object name spaces	9-27
AdminControl method reference (1 of 4)	9-28
AdminControl method reference (2 of 4)	9-29
AdminControl method reference (3 of 4)	9-30
AdminControl method reference (4 of 4)	9-31
Unit summary	9-32
Checkpoint questions	9-33
Checkpoint answers	9-34
Exercise 5	9-35
Exercise objectives	9-36
 Unit 10. Administrative object basics: AdminTask.....	 10-1
Unit objectives	10-2
When to use the AdminTask object	10-3
The AdminTask object	10-4
How to use the AdminTask object	10-5
AdminTask command group reference (1 of 5)	10-6
AdminTask command group reference (2 of 5)	10-7
AdminTask command group reference (3 of 5)	10-8
AdminTask command group reference (4 of 5)	10-9
AdminTask command group reference (5 of 5)	10-10
AdminTask help commands	10-11
AdminTask help command examples	10-12
AdminTask command general syntax	10-13
AdminTask command syntax combinations (1 of 2)	10-14
AdminTask command syntax combinations (2 of 2)	10-15
AdminTask options syntax	10-16
Example: Using AdminTask in batch mode (1 of 3)	10-17
Example: Using AdminTask in batch mode (2 of 3)	10-18
Example: Using AdminTask in batch mode (3 of 3)	10-19
AdminTask interactive mode	10-20
Example: Using AdminTask in interactive mode (1 of 3)	10-21
Example: Using AdminTask in interactive mode (2 of 3)	10-22
Example: Using AdminTask in interactive mode (2 of 3)	10-23
AdminTask utility commands	10-24
Commonly used AdminTask commands	10-25
AdminTask usage considerations	10-26
Unit summary	10-27

Checkpoint questions	10-28
Checkpoint answers	10-29
Exercise 6	10-30
Exercise objectives	10-31
Unit 11. Introduction to the PlantsByWebSphere and messaging applications .	11-1
Unit objectives	11-2
Topics	11-3
11.1. PlantsByWebSphere application	11-5
PlantsByWebSphere application	11-6
PlantsByWebSphere application	11-7
PlantsByWebSphere sample	11-8
Home page: <a href="http://<hostname>/PlantsByWebSphere">http://<hostname>/PlantsByWebSphere	11-10
Login and registration	11-11
My Account	11-12
Shopping	11-13
Select an item	11-14
Shopping cart	11-15
Checking out: Billing information	11-16
Checking out: Submit	11-17
Help page	11-18
View Server Info page	11-19
Admin home page	11-20
HTML documentation for PlantsByWebSphere (1 of 2)	11-21
HTML documentation for PlantsByWebSphere (2 of 2)	11-22
11.2. WebSphere Default Messaging applications	11-25
WebSphere Default Messaging applications	11-26
Message Sender Simulator	11-27
Trade processor application	11-28
Topology diagram	11-29
Monitor web page (1 of 2)	11-30
Monitor web page (2 of 2)	11-31
Unit summary	11-32
Unit 12. Configuring and managing servers and server resources by using scripting	12-1
Unit objectives	12-2
Configuring servers	12-3
Configuring application servers, clusters, and cluster members	12-4
AdminTask: ServerManagement command group	12-5
Creating an application server template	12-6
Example: Creating an application server template	12-7
Creating an application server	12-8
Example: Creating an application server	12-9
AdminTask: ClusterConfigCommands command group	12-10
Creating a cluster	12-11
Example: Creating a cluster	12-12
Creating cluster members	12-13

Example: Creating cluster members	12-14
Modifying JVM properties	12-15
Setting JVM properties	12-16
JVM property reference (1 of 3)	12-17
JVM property reference (2 of 3)	12-18
JVM property reference (3 of 3)	12-19
Example: Setting JVM properties	12-20
Managing servers: Starting and stopping server	12-21
Managing servers: Stopping node agents	12-22
Managing clusters: Starting clusters	12-23
Managing clusters: Stopping clusters	12-24
Managing clusters: Query cluster state	12-25
Security: Determining whether security is enabled or disabled	12-26
Security: Enabling and disabling administrative security	12-27
Security: Enable and disable Java 2 security	12-28
AdminTask: AuthorizationGroupCommands command group	12-29
Unit summary	12-30
Checkpoint questions	12-31
Checkpoint answers	12-32
Exercise 7	12-33
Exercise objectives	12-34
Exercise 8	12-35
Exercise objectives	12-36
 Unit 13. Deploying and managing the PlantsByWebSphere application by using scripting	13-1
Unit objectives	13-2
JDBC resources	13-3
AdminTask: JDBCProviderManagement command group	13-4
Create JDBC providers	13-5
Example: Creating JDBC providers	13-6
Create data sources	13-7
Example: Creating data sources	13-8
Modifying WebSphere variables	13-9
Example: modifying WebSphere variables	13-10
JMS resources	13-11
Creating a service integration bus	13-12
Adding service integration bus members	13-13
Bus destinations	13-14
Example: creating an SIBus, bus member, and JMS destinations	13-15
JMS application resources	13-16
JMS connection factory	13-17
Example: creating JMS connection factory	13-18
JMS queue destination	13-19
Example: creating JMS queue	13-20
JMS activation specification	13-21
Example: creating JMS activation specification	13-22
Adding users and groups to bus connector roles	13-23

JMS resources	13-24
Managing applications	13-25
Installing and uninstalling applications	13-26
Starting and stopping applications	13-27
Other application management tasks	13-28
Unit summary	13-29
Checkpoint questions	13-30
Checkpoint answers	13-31
Exercise 9	13-32
Exercise objectives	13-33
Unit 14. Scripting methodologies and recommendations	14-1
Unit objectives	14-2
Topics	14-3
Why have such formal guidelines? (1 of 2)	14-4
Why have such formal guidelines? (2 of 2)	14-5
14.1. Programming discipline for writing scripts	14-7
Programming discipline for writing scripts	14-8
Use source control	14-9
Unit Tests	14-10
Coding standards (1 of 2)	14-11
Coding standards (2 of 2)	14-12
Peer review	14-13
Refactor the code	14-14
Keep it simple	14-15
Continuous integration	14-16
Extreme Programming	14-17
Scripting and asynchronous programming	14-18
Why asynchronous scripting?	14-19
References (1 of 2)	14-20
References (2 of 2)	14-21
Unit summary	14-22
Checkpoint questions	14-23
Checkpoint answers	14-24
Unit 15. Using ws_ant	15-1
Unit objectives	15-2
What is Ant?	15-3
Ant features	15-4
Why use Ant?	15-5
Ant buildfiles	15-6
Ant: buildfile example	15-7
Ant: projects	15-8
Ant: properties	15-9
Ant: property file example	15-10
Ant: tasks	15-11
Ant: task types (1 of 2)	15-12
Ant: task types (2 of 2)	15-14

Ant: targets	15-15
Ant: target dependencies	15-17
What is ws_ant?	15-18
ws_ant: predefined tasks (1 of 2)	15-19
ws_ant: predefined tasks (2 of 2)	15-20
ws_ant: common task attributes	15-21
ws_ant: defining tasks	15-22
ws_ant: installing and uninstalling applications	15-23
ws_ant: starting and stopping applications	15-24
ws_ant: Running the JSP precompilation tool	15-25
ws_ant: build.xml file example (1 of 2)	15-26
ws_ant: build.xml file example (2 of 2)	15-27
ws_ant: script execution	15-28
ws_ant: script execution examples (1 of 2)	15-29
ws_ant: script execution examples (2 of 2)	15-30
ws_ant: startup options (1 of 2)	15-31
ws_ant: startup options (2 of 2)	15-33
IBM Assembly and Deploy Tools: Ant support	15-34
IBM Assembly and Deploy Tools: creating buildfiles	15-35
IBM Assembly and Deploy Tools: Ant editor	15-36
IBM Assembly and Deploy Tools: Ant view	15-37
IBM Assembly and Deploy Tools: Ant preferences	15-38
IBM Assembly and Deploy Tools: Ant formatter	15-40
IBM Assembly and Deploy Tools: Ant templates	15-41
IBM Assembly and Deploy Tools: ws_ant support	15-42
IBM Assembly and Deploy Tools: defining tasks	15-43
IBM Assembly and Deploy Tools: Ant properties files	15-44
IBM Assembly and Deploy Tools: Running buildfiles (1 of 2)	15-45
IBM Assembly and Deploy Tools: Running buildfiles (2 of 2)	15-46
IBM Assembly and Deploy Tools: accessing help	15-47
Unit summary	15-48
Checkpoint questions	15-49
Checkpoint answers	15-50
Exercise 10	15-51
Exercise objectives	15-52

Unit 16. Properties file based configurations and Jython script library.....	16-1
Unit objectives	16-2
Configuration repository: The issues	16-3
Properties file based configuration: A solution	16-4
Properties file configuration content	16-5
Properties file configuration commands	16-6
Benefits	16-7
Format of a properties file	16-8
Configuration properties file content	16-9
Format of a properties file	16-10
Properties file configuration commands	16-11
Details of extractConfigProperties	16-12

Details of validateConfigProperties	16-13
Details of applyConfigProperties	16-14
Details of deleteConfigProperties	16-15
Details of createPropertiesFileTemplates	16-16
Comparing properties files and archives	16-17
Troubleshooting	16-18
Modify existing server configuration	16-19
Properties file based configuration examples	16-20
Delete a configuration object	16-21
Create a configuration object	16-22
Install an application	16-23
Jython script library	16-24
Disadvantage of Jython scripting language	16-25
Script library	16-26
Script library location	16-27
Script libraries	16-28
Application script library	16-29
Resources script library	16-30
Security script library	16-31
Servers script library	16-32
System script library	16-33
Utilities script library	16-34
Script library help	16-35
Sample script procedures	16-36
Sample script (1 of 2)	16-37
Sample script (2 of 2)	16-38
Combining script library with Jython scripts	16-39
Custom script library	16-40
Recommendations for using the script library	16-41
Unit summary	16-42
Checkpoint questions	16-43
Checkpoint answers	16-44
Unit 17. Using scripts with the job manager and CIM	17-1
Unit objectives	17-2
Topics	17-3
17.1. Job manager profile and target hosts	17-5
Job manager profile and target hosts	17-6
Flexible management	17-7
Flexible management characteristics	17-8
Job manager	17-9
Job manager topology	17-10
Creating the job manager profile	17-11
JobManagerNode command group for the AdminTask object	17-13
Registering host computers with job managers	17-14
Submitting administrative jobs by using the job manager	17-15
Testing connection to a remote target (1 of 2)	17-16
Testing connection to a remote target (1 of 2)	17-17

17.2. Centralized Installation Manager (CIM) in the job manager	17-19
Centralized Installation Manager (CIM) in the job manager	17-20
Centralized Installation Manager (CIM) in the job manager	17-21
Centralized installation manager (CIM) functions	17-22
Installing the Installation Manager on a remote host	17-23
Submitting a Manage offerings job	17-24
Install WebSphere on a remote host	17-25
Creating a profile	17-26
17.3. Creating a flexible management environment	17-27
Creating a flexible management environment	17-28
Registering deployment manager with job manager	17-29
Flexible management jobs	17-30
Asynchronous nature of jobs	17-31
Managing the server run time	17-32
Scheduling future and recurring jobs (1 of 2)	17-33
Scheduling future and recurring jobs (2 of 2)	17-35
Administrative agent	17-36
Administrative agent topology	17-37
Creating the administrative agent profile	17-38
Registering application server nodes	17-40
Register application server profiles with the administrative agent	17-41
Register application server profiles with the job manager through the administrative agent	17-42
17.4. Automating the installation of WebSphere Application Server	17-43
Automating the installation of WebSphere Application Server	17-44
Silent installation of IBM Installation Manager (IIM)	17-45
Silent installation of IBM Installation Manager (IIM)	17-46
Create a response file for installing WebSphere core product	17-47
Example: Response file for WebSphere core product (1 of 2)	17-48
Example: Response file for WebSphere core product (2 of 2)	17-49
Silent installation of WebSphere core product files	17-51
Create a profile with response file and the manageprofiles command	17-52
Unit summary	17-54
Checkpoint questions	17-55
Checkpoint answers	17-56
Exercise 11	17-57
Exercise objectives	17-58
Unit 18. Course summary	18-1
Unit objectives	18-2
Course learning objectives	18-3
Class evaluation	18-4
To learn more on the subject	18-5
References	18-6
Unit summary	18-7
Appendix 19. List of abbreviations and acronyms	19-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

AFS™	AIX®	DB™
DB2®	Express®	HACMP™
IMS™	MVS™	RACF®
Rational®	RDN®	Tivoli®
WebSphere®	WPM®	z/OS®
zSeries®		

Intel and Intel Core are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

VMware and the VMware "boxes" logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the "Marks") of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

•

•

Course description

IBM WebSphere Application Server v8.5 Scripting and Automation

Duration: 5 days

Purpose

This 5-day instructor-led course teaches you the skills that are required to automate the administration of a WebSphere Application Server V8.5 environment.

Most administrative tasks are multi-step and repeatedly used, and GUI-based administration tools can quickly become impractical and even a burden. WebSphere Application Server offers a collection of tools and technologies that you can use to create automated scripts, facilitating system administration.

The course describes the scripting facilities in WebSphere Application Server and teaches you how to use the Jython language with WebSphere administrative objects to develop scripts that automate common administrative tasks.

Hands-on exercises throughout the course reinforce lecture content and give you practical experience with the WebSphere Application Server V8.5 scripting environment. Exercises include tasks such as creating and configuring an application server environment, installing and configuring the IBM HTTP Server, and automating the installation of WebSphere Application Server.

Audience

This course is designed for systems integrators, administrators, architects, and developers who use WebSphere Application Server to configure, administer, or architect solutions.

Prerequisites

Before taking this course, you should have a thorough understanding of the topologies and runtime administration of WebSphere Application Server Network Deployment environments. You can gain this knowledge through experience or by completing one of the following courses:

- WU805 or VU805, *Transition to WebSphere Application Server V8.5 for Administrators*

- WA585 or VA585, *WebSphere Application Server V8.5 Administration*

Objectives

After completing this course, you should be able to:

- Describe the support in WebSphere Application Server for scripting and automation
- Use Jython and the IBM Assembly and Deploy Tools (IADT) to develop automated scripts
- Identify the administrative objects and programming APIs needed for administrative scripting
- Use the wsadmin tool to prototype and run scripts
- Write scripts to automate common WebSphere Application Server administration tasks
- Describe the use of Ant to automate tasks
- Use Jython scripting to submit jobs to the job manager

Agenda

Day 1

- Course introduction
- Unit 1 - Introduction to scripting
- Unit 2 - WebSphere Application Server administrative concepts and architecture
- Unit 3 - WebSphere Application Server scripting facilities
- Unit 4 - Using wsadmin
- Exercise 1 - Using wsadmin to explore the WebSphere Application Server environment

Day 2

- Unit 5 - Jython basics
- Unit 6 - Jython development by using the IADT
- Exercise 2 - Using the IBM Assembly and Deploy Tools (IADT) to develop Jython scripts
- Unit 7 - Administrative object basics: Help and AdminConfig
- Exercise 3 - Using the Help and AdminConfig objects

Day 3

- Unit 8 - Administrative object basics: AdminApp
- Exercise 4 - Using the AdminApp object
- Unit 9 - Administrative object basics: AdminControl
- Exercise 5 - Using the AdminControl object
- Unit 10 - Administrative object basics: AdminTask
- Exercise 6 - Using the AdminTask object

Day 4

- Unit 11 - Introduction to the PlantsByWebSphere and messaging applications
- Unit 12 - Configuring and managing servers and server resources by using scripting
- Exercise 7 - Creating and configuring the Plants server environment with scripting
- Exercise 8 - Installing and configuring the IBM HTTP Server
- Unit 13 - Deploying and managing the PlantsByWebSphere application by using scripting
- Exercise 9 - Deploying the PlantsByWebSphere application

Day 5

- Unit 14 - Scripting methodologies and recommendations
- Unit 15 - Using ws_ant
- Exercise 10 - ws_ant scripting and configuring the service integration bus
- Unit 16 - Properties file based configurations and Jython script library
- Unit 17 - Using scripts with the job manager and CIM
- Exercise 11 - Automating the installation of WebSphere Application Server
- Unit 18 - Course summary

Unit 1. Introduction to scripting

What this unit is about

This unit introduces scripting by describing what it is, why it should be used and when to use it.

What you should be able to do

After completing this unit, you should be able to:

- Define the purpose of scripting
- Describe the benefits of automation
- Identify common administrative tasks that are good candidates for scripting

Unit objectives

After completing this unit, you should be able to:

- Define the purpose of scripting
- Describe the benefits of automation
- Identify common administrative tasks that are good candidates for scripting

© Copyright IBM Corporation 2013

Figure 1-1. Unit objectives

WA680 / VA6801.0

Notes:

The administrator role

- A system administrator is responsible for configuring and administering an enterprise's computing and networking infrastructure.
 - Responsible for deploying, configuring and ensuring the runtime health of Java Enterprise applications

Some administrative tasks:

-  Product Installation
-  Product and application migration
-  Cell configuration
-  Application deployment
-  Runtime options
-  Problem determination
-  Security configuration

© Copyright IBM Corporation 2013

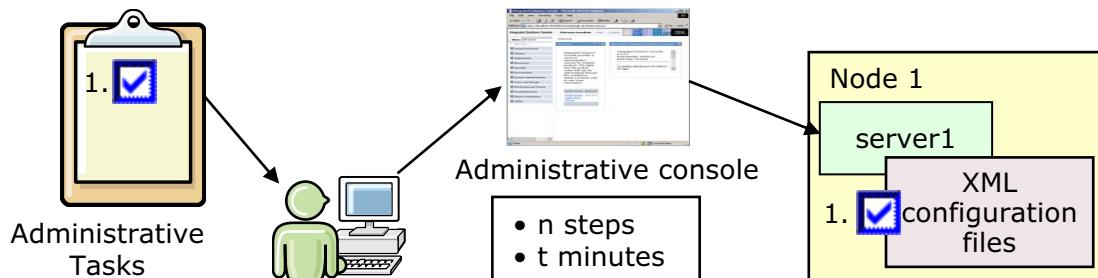
Figure 1-2. The administrator role

WA680 / VA6801.0

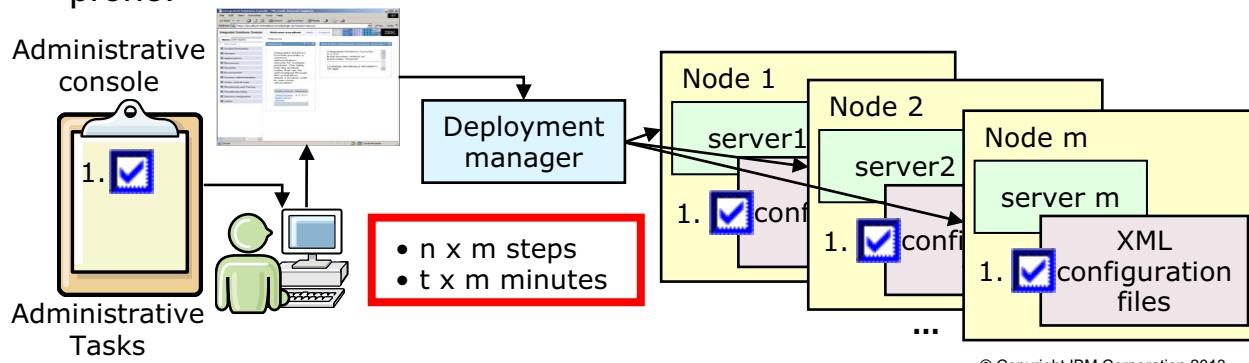
Notes:

Why automate administrative tasks? (1 of 2)

- Manual configuration is effective for simple environments.



- Manual configuration in a complex environment is tedious and error prone.



© Copyright IBM Corporation 2013

Figure 1-3. Why automate administrative tasks? (1 of 2)

WA680 / VA6801.0

Notes:

Scripts and automation make administration tasks repeatable and reliable. Some things are done infrequently. Scripts are best used for repetitious activities.

Why automate administrative tasks? (2 of 2)

- The same tasks often must be performed on multiple environments.
 - It is preferable to find a way to do them manually first and automate for subsequent requests.

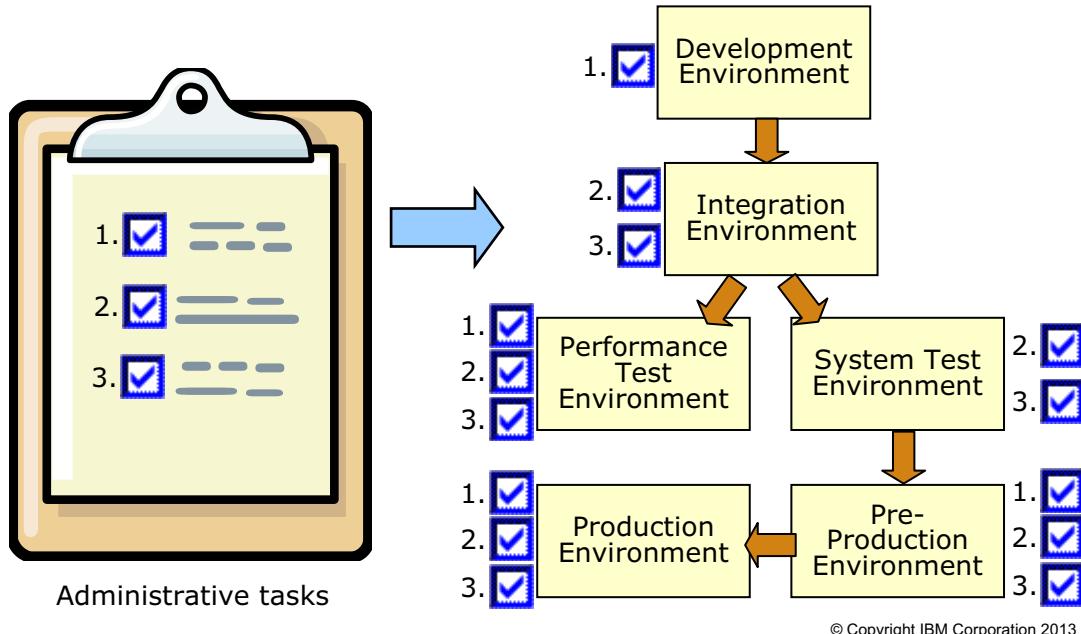


Figure 1-4. Why automate administrative tasks? (2 of 2)

WA680 / VA6801.0

Notes:

Using scripts to create and configure environments is an excellent way to create consistent environments. Test the scripts as you build the environments.

Benefits of automation

- Eliminates errors and increases reliability
 - Reduces the possibility of human error
- Ensures consistency
 - Naming conventions
 - Standard procedures and policies
- Promotes reuse
 - After they are tested, automated processes can be run repeatedly.
- Reduces manual labor and increases performance
 - Automated tasks are less labor intensive and faster to run.
- Enables scheduling flexibility
 - You can schedule unattended execution of tasks at pre-determined times or intervals.
- In short, automation makes administration easier

© Copyright IBM Corporation 2013

Figure 1-5. Benefits of automation

WA680 / VA6801.0

Notes:

WebSphere automation capabilities (1 of 2)

- Installation response files
 - Allow you to specify installation options once and use them for multiple installations of WebSphere Application Server
 - Enable silent execution mode
- Installation Manager
 - Complementary tool to combine the installation of WebSphere Application Server with optional assets (for example, Fix Packs) in a single step
- Command-line utilities
 - Shell scripts on UNIX or batch files on Windows
 - Run from standard shell or command prompt
 - Allow you to control different aspects of the WebSphere environment

© Copyright IBM Corporation 2013

Figure 1-6. WebSphere automation capabilities (1 of 2)

WA680 / VA6801.0

Notes:

There are many tools for automating WebSphere administration. Installation response files allow you to specify installation options once, and use them for multiple installations of WebSphere Application Server. The installation manager is a tool to install WebSphere Application Servers. There are many command-line utilities available as shell scripts or batch files. These command-line utilities can be run from a standard shell or command prompt, or combined as part of other shell and command scripts.

WebSphere automation capabilities (2 of 2)

- WebSphere Ant tasks
 - Facilitate build and deploy processes to WebSphere
- Java Management Extensions (JMX) framework
 - Provides standards-based capabilities to control and manage an application server
 - Allows you to create custom Java clients to access managed resources
- wsadmin scripting tool
 - Scripting interface that allows you to run administrative commands interactively or by running a script of commands

© Copyright IBM Corporation 2013

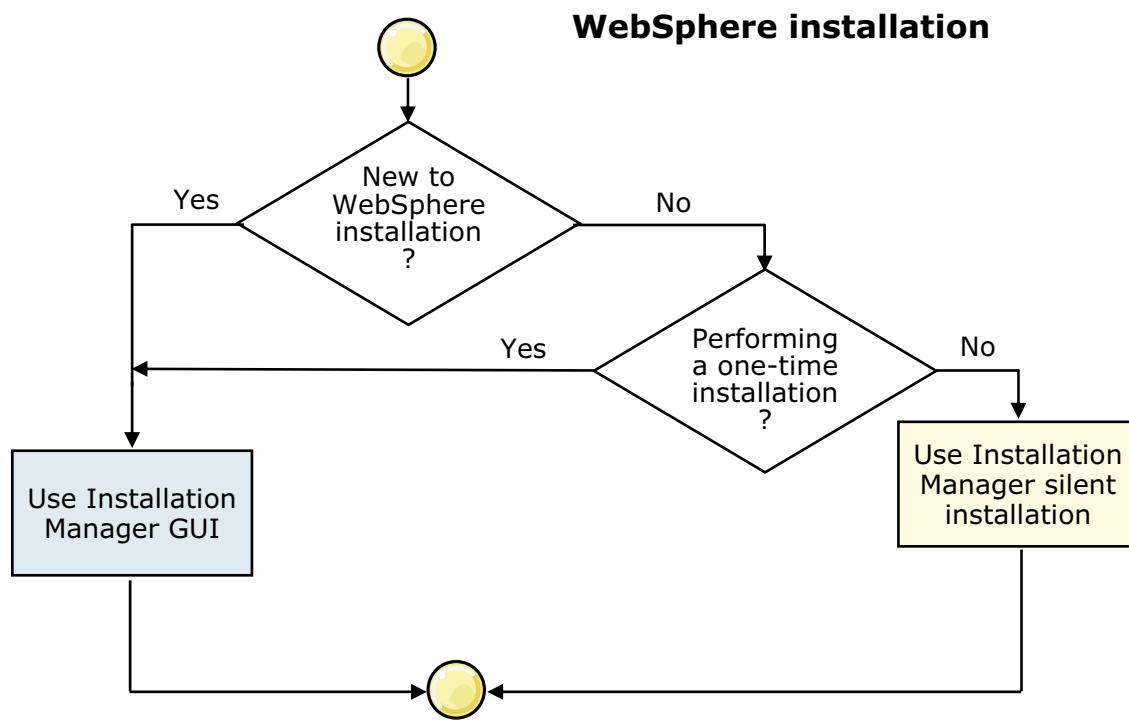
Figure 1-7. WebSphere automation capabilities (2 of 2)

WA680 / VA6801.0

Notes:

WebSphere ws_ant is an extension of Ant (Another Neat tool). WebSphere Ant can build, deploy, and configure WebSphere Application Server. The wsadmin scripting tool allows for administration of WebSphere Application Server from the command line by using either the interactive, command, or script file mode. All of these tools provide an alternative to the web-based administrative console.

WebSphere automation: What to use when (1 of 2)



© Copyright IBM Corporation 2013

Figure 1-8. WebSphere automation capabilities: What to use when (1 of 2)

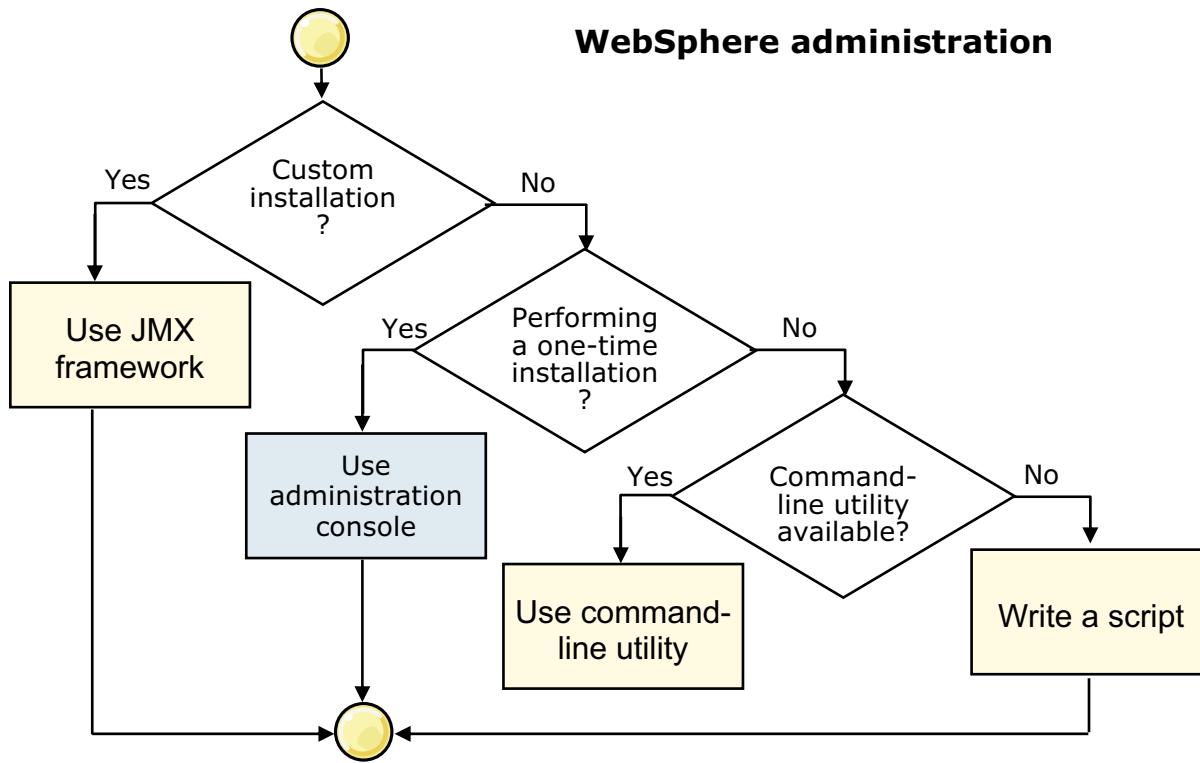
WA680 / VA6801.0

Notes:

Before you begin to automated everything within WebSphere Application Server, review this flowchart. If you are new to installing WebSphere Application Server, use the GUI installation with the installation manager. If this installation is a one time installation, you should consider the GUI installation. If this installation is repeated many times, then it might be best to perform a silent installation by using the response files and the Installation Manager.

After your WebSphere application Server infrastructure is built, you should ask “Do I have to create a custom administration client or objects?” If you need a custom or custom objects, then use the JMX framework. Much of the WebSphere administration does not require the JMX framework. If you are making a one time change or creating a simple topology, use the web-based administrative console. If there is a command-line utility available for the task such as StartServer, then use that utility. If there is no appropriate command-line utility, consider writing a script.

WebSphere automation: What to use when (2 of 2)



© Copyright IBM Corporation 2013

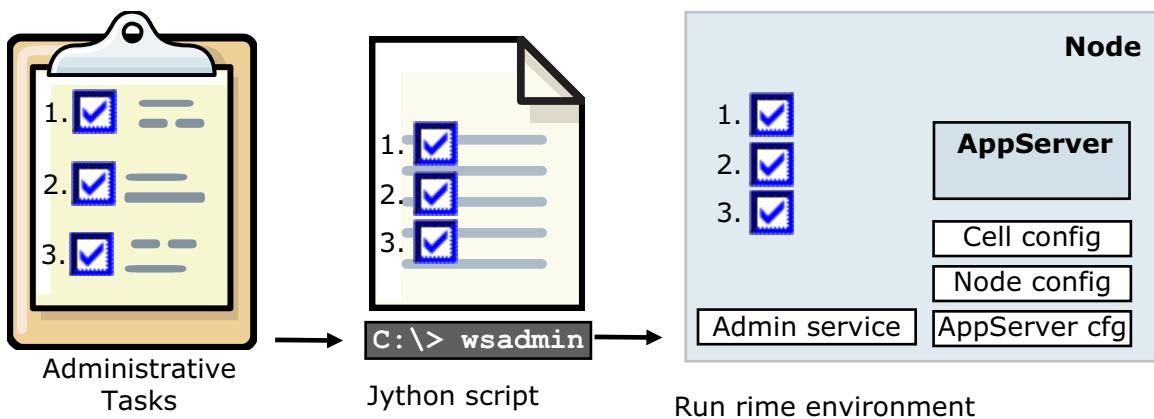
Figure 1-9. WebSphere automation: What to use when (2 of)

WA680 / VA6801.0

Notes:

What is scripting?

- Non-graphical way to configure and manage WebSphere Application Server
 - Commands are written to a script file
 - Script file is run to implement the changes
- `wsadmin` and `ws_ant` are the scripting tools that WebSphere provides.



© Copyright IBM Corporation 2013

Figure 1-10. What is scripting?

WA680 / VA6801.0

Notes:

Automation is about creating tasks for automating administrative tasks with little or no human intervention.

Types of administrative tasks that should be automated

- Product installation duplications
 - Installing WebSphere the same way multiple times
 - Easier to use silent installation with a response file
- Bulk changes
 - Change that deal with many items
 - For example, defining a list of users to the environment
- Repetitive tasks
 - Making the same change to multiple environments
 - For example, installing an enterprise application on multiple servers
- Elaborate tasks
 - Changes that require multiple dependent steps
 - For example, building a cell and creating a cluster

© Copyright IBM Corporation 2013

Figure 1-11. Types of administrative tasks that should be automated

WA680 / VA6801.0

Notes:

Common administrative tasks that should be automated

Task	Reason for automation and tools to use
Cell configuration	<ul style="list-style-type: none"> Requires defining many components. These components include servers, clusters, data sources, connectors, and variables. Often repeated across environments Use parameterized wsadmin scripts and ws_ant
Application deployment	<ul style="list-style-type: none"> Usually requires many steps Must be repeatable Use wsadmin and ws_ant
Cell services start and stop	<ul style="list-style-type: none"> Involves many components. These components include the deployment manager, node agents, application servers, and enterprise applications Use wsadmin and ws_ant
Back up and restore	<ul style="list-style-type: none"> Involves many configuration files Use backupConfig and restoreConfig command-line utilities

© Copyright IBM Corporation 2013

Figure 1-12. Common administrative tasks that should be automated

WA680 / VA6801.0

Notes:

When not to use scripting

- Ad hoc or one-time configuration or operational change
 - For example:
 - Building a playpen environment
 - Viewing real-time activity
 - Examining product information
 - Examining logs
 - Administrative console is better suited for these types of tasks
- Task is too difficult or expensive to script
- Someone new to WebSphere Application Server uses the administrative console

© Copyright IBM Corporation 2013

Figure 1-13. When not to use scripting

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Define the purpose of scripting
- Describe the benefits of automation
- Identify common administrative tasks that are good candidates for scripting

© Copyright IBM Corporation 2013

Figure 1-14. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. Which of the following is NOT a benefit gained from automating administrative tasks?
 - a) Scheduling flexibility
 - b) Server availability
 - c) Error reduction
 - d) Reusability
2. Name three of the five automation capabilities that WebSphere Application Server provides.
3. True or false: When making a configuration change that requires a list of values to be specified, it is best to use the administrative console.

© Copyright IBM Corporation 2013

Figure 1-15. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

Checkpoint answers

1. Which of the following is NOT a benefit gained from automating administrative tasks?
 - a) Scheduling flexibility
 - b) Server availability
 - c) Error reduction
 - d) Reusability

Answer: **b) Server availability**

2. Name three of the five automation capabilities that WebSphere Application Server provides.

Any three of the following are correct answers:

- **Silent installation**
- **Command-line utilities**
- **wsadmin scripting**
- **ws_ant scripting**
- **Java Management Extensions (JMX) framework**

3. True or false: When making a configuration change that requires a list of values to be specified, it is best to use the administrative console.

Answer: **False**

© Copyright IBM Corporation 2013

Figure 1-16. Checkpoint answers

WA680 / VA6801.0

Notes:

Unit 2. WebSphere Application Server administrative concepts and architecture

What this unit is about

This unit provides an overview of the WebSphere Application Server architectural components important to system administration.

What you should be able to do

After completing this unit, you should be able to:

- Describe the overall architecture of WebSphere Application Server V8.5
- Explain the administration model
- Identify basic command-line commands that are used for administration

How you will check your progress

- Checkpoint questions

References

WebSphere Application Server V8.5 Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp>

Unit objectives

After completing this unit, you should be able to:

- Describe the overall architecture of WebSphere Application Server V8.5
- Explain the administration model
- Identify basic command-line commands that are used for administration

© Copyright IBM Corporation 2013

Figure 2-1. Unit objectives

WA680 / VA6801.0

Notes:



Topics

- Architecture run time
- Architecture administration
- Profiles
- Network deployment concepts
- Advanced concepts



© Copyright IBM Corporation 2013

Figure 2-2. Topics

WA680 / VA6801.0

Notes:

2.1. Architecture run time

These topics provide information about using the IBM Installation Manager.

Architecture run time



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

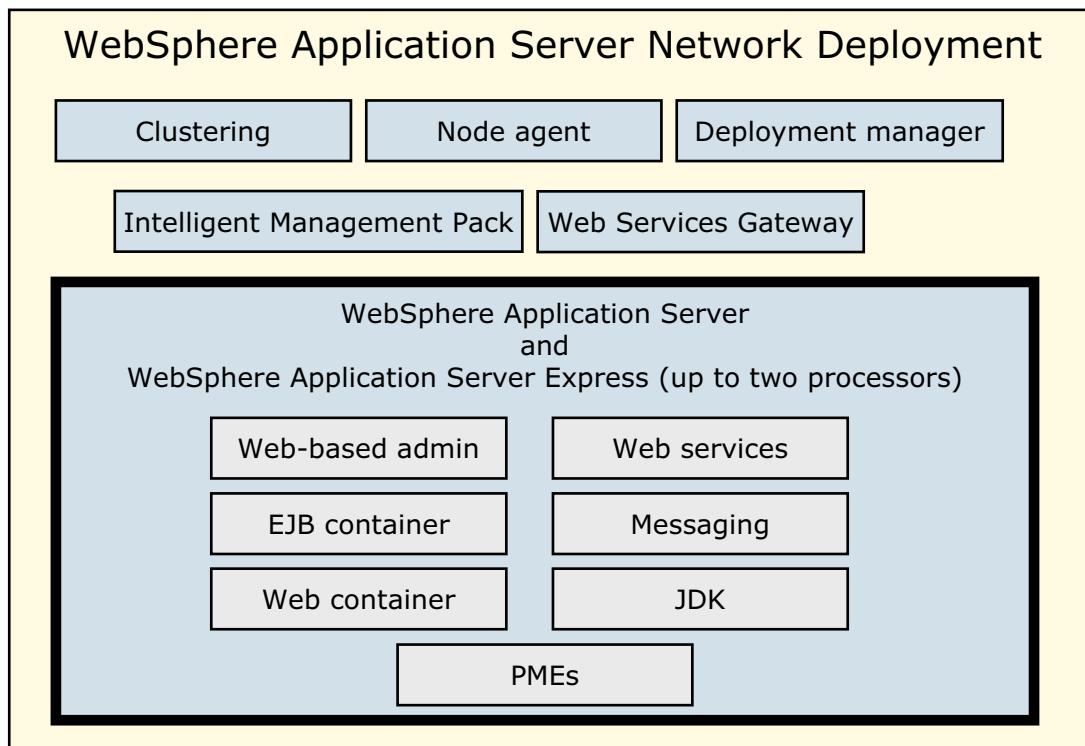
7.0

Figure 2-3. Architecture run time

WA680 / VA6801.0

Notes:

Version 8.5 packaging



© Copyright IBM Corporation 2013

Figure 2-4. Version 8.5 packaging

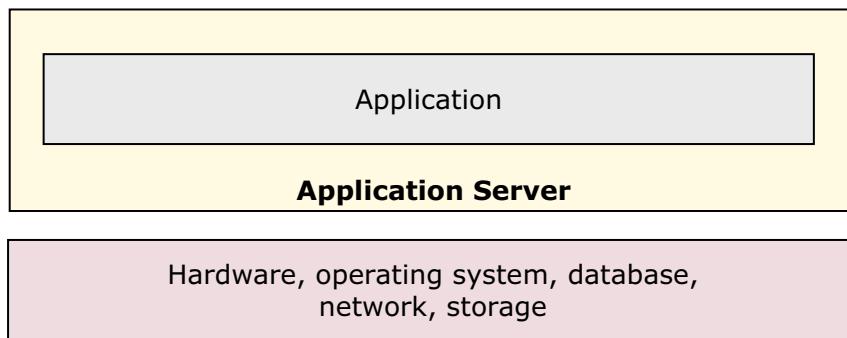
WA680 / VA6801.0

Notes:

In this unit, the focus is on the Express and Base versions of WebSphere Application Server. All of the concepts that this unit presents are applicable to all three versions: Express, Base, and Network Deployment.

WebSphere Application Server basics

- WebSphere Application Server
 - Is a platform on which Java based business applications run
 - Is an implementation of the Java Platform, Enterprise Edition (Java EE) specification
 - Provides services (database connectivity, threading, workload management) that the business applications can use



© Copyright IBM Corporation 2013

Figure 2-5. WebSphere Application Server basics

WA680 / VA6801.0

Notes:

This diagram illustrates the differences between the application, the application server, and the hardware and operating system layers. WebSphere Application Server is a platform on which Java based business applications run and is an implementation of the Java Platform, Enterprise Edition specification. It provides services (database connectivity, threading, workload management) that the business applications can use.

WebSphere architecture run time (1 of 10)



Browser



© Copyright IBM Corporation 2013

Figure 2-6. WebSphere architecture run time (1 of 10)

WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

The main element is the application server, a Java process that encapsulates many services, including the containers, where business logic runs. If you are familiar with Java EE, you recognize the web container and the EJB container. The web container runs servlets and JavaServer Pages (JSPs), both of which are Java classes that generate markup that a web browser views. Traffic into and out of the web container travels through the embedded IBM HTTP Server. While servlets and JSPs can act independently, they most commonly make calls to EJBs to run business logic or access data. EJBs, which run in the EJB container, are easily reusable Java classes. They most commonly communicate with a relational database or other external source of application data, either returning that data to the web container or changing the data on behalf of the servlet or JSP.

The JMS messaging engine is built into the application server. This messaging engine is pure Java. JMS destinations, which are known as queues and topics, provide

asynchronous messaging services to the code that runs inside the containers. JMS is covered in more depth later in this course.

The web services engine provides the ability for application components to be exposed as web services, which can be accessed by using SOAP. This process is shown in greater detail later on.

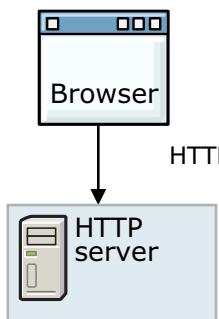
Several other services are run within the application server, including the dynamic cache, data replication, security, and others. These topics are presented later in the course.

In addition, there are some important components outside of the application server process.

WebSphere Application Server also provides a plug-in for HTTP servers that determines the HTTP traffic that WebSphere intends to handle, and routes the requests to the appropriate server. The plug-in is also a critical player in workload management of HTTP requests, as it can distribute the load to multiple application servers, and steer traffic away from unavailable servers. It also reads its configuration from a special XML file.

1 of 10: The browser is the main interaction mechanism for users.

WebSphere architecture run time (2 of 10)



© Copyright IBM Corporation 2013

Figure 2-7. WebSphere architecture run time (2 of 10)

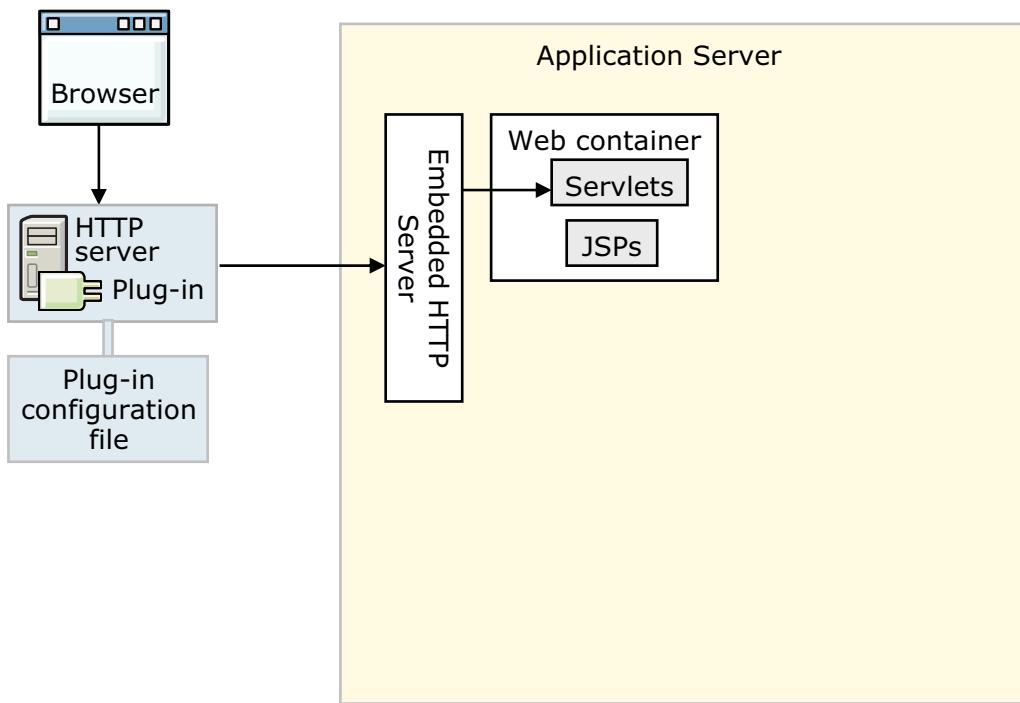
WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

2 of 10: A browser communicates with a web server (HTTP server).

WebSphere architecture run time (3 of 10)



© Copyright IBM Corporation 2013

Figure 2-8. WebSphere architecture run time (3 of 10)

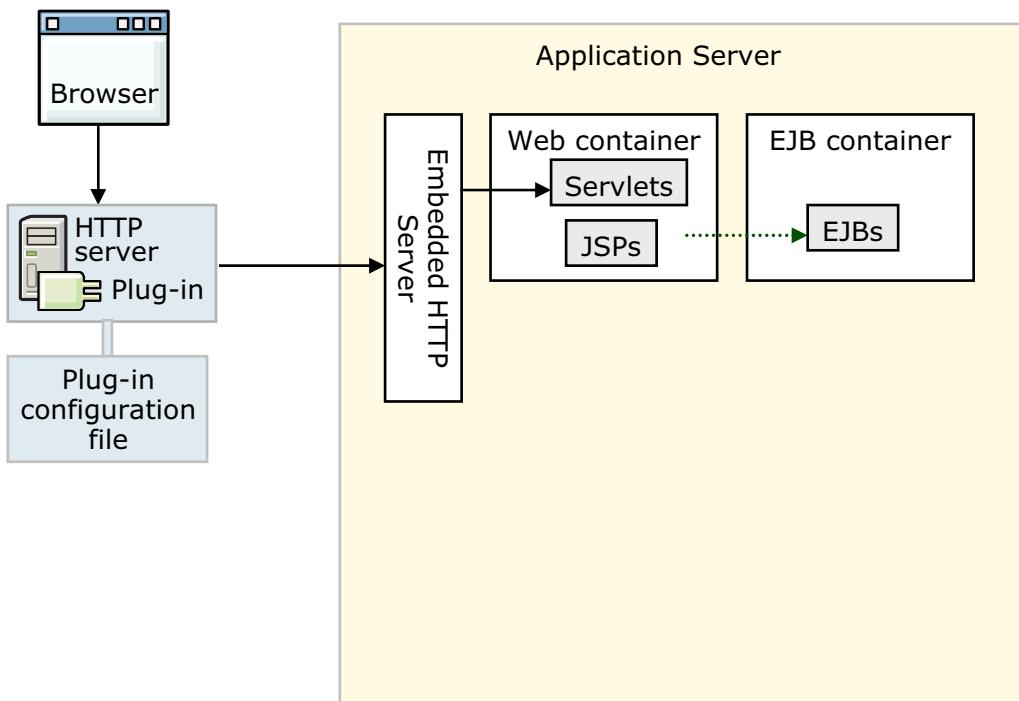
WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

3 of 10: The way the request gets into the WebSphere Application Server is from the HTTP server plug-in that is loaded with the HTTP server. This request is forwarded to the embedded HTTP server within the application server. The embedded server forwards the request into the web container to either a servlet or a JSP.

WebSphere architecture run time (4 of 10)



© Copyright IBM Corporation 2013

Figure 2-9. WebSphere architecture run time (4 of 10)

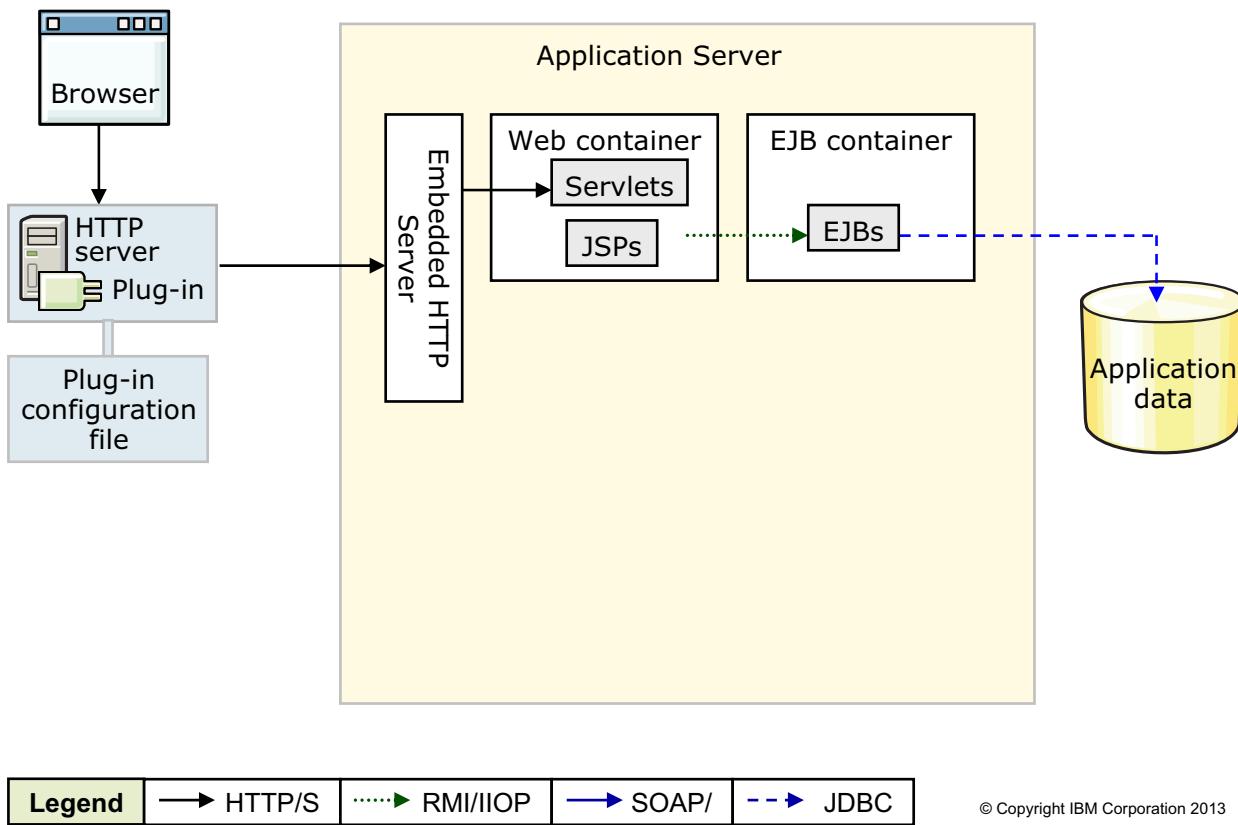
WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

4 of 10: If these servlets or JSPs access distributed business logic or a database, the Java EE way to accomplish it is through EJBs within the EJB container.

WebSphere architecture run time (5 of 10)


Legend

→ HTTP/S ▶ RMI/IOP → SOAP/ - - ▶ JDBC

© Copyright IBM Corporation 2013

Figure 2-10. WebSphere architecture run time (5 of 10)

WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

5 of 10: EJBs (entity in this case) can communicate with the database to store, retrieve, query, and delete data. JDBC is one way that this communication can occur.

WebSphere architecture run time (6 of 10)

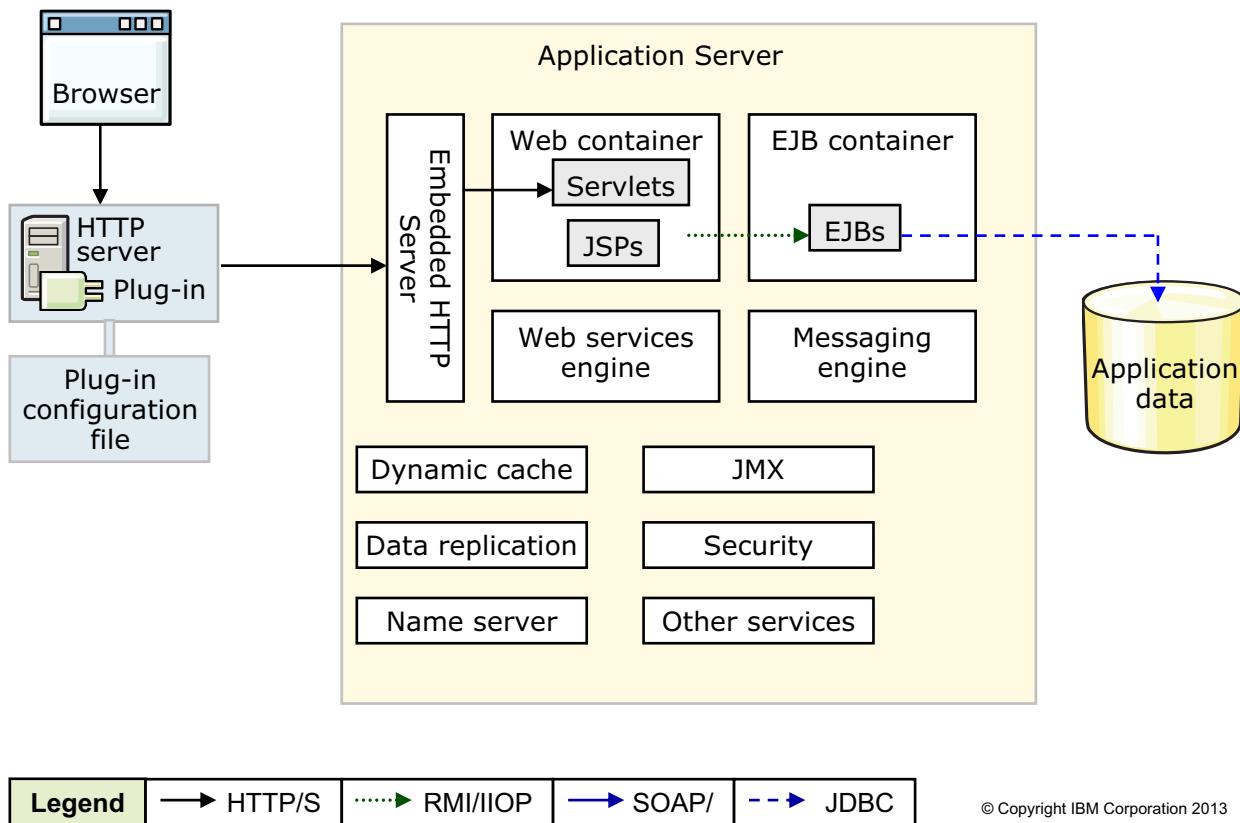


Figure 2-11. WebSphere architecture run time (6 of 10)

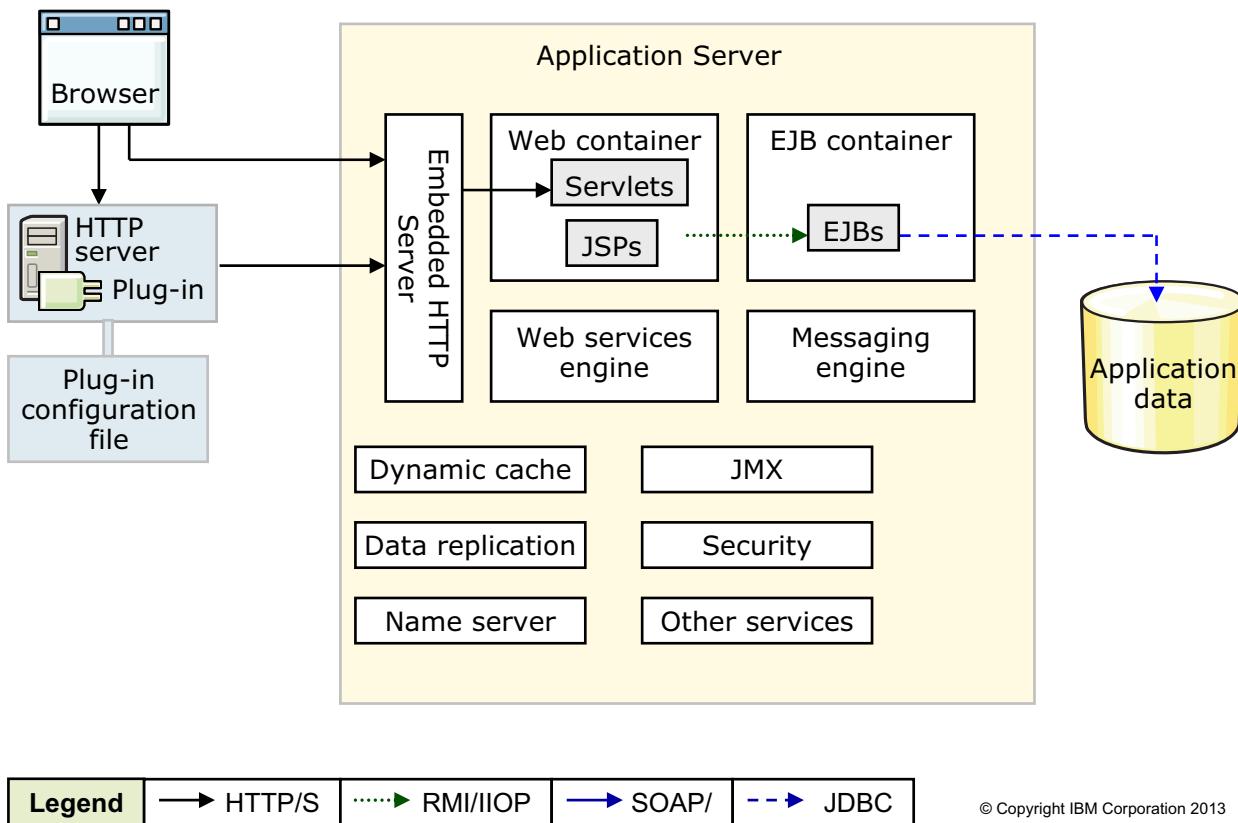
WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

6 of 10: There are many other services that are provided within WebSphere Application Server. Some of those services are listed here.

WebSphere architecture run time (7 of 10)



Legend

→ HTTP/S ► RMI/IOP —► SOAP/ -► JDBC

© Copyright IBM Corporation 2013

Figure 2-12. WebSphere architecture run time (7 of 10)

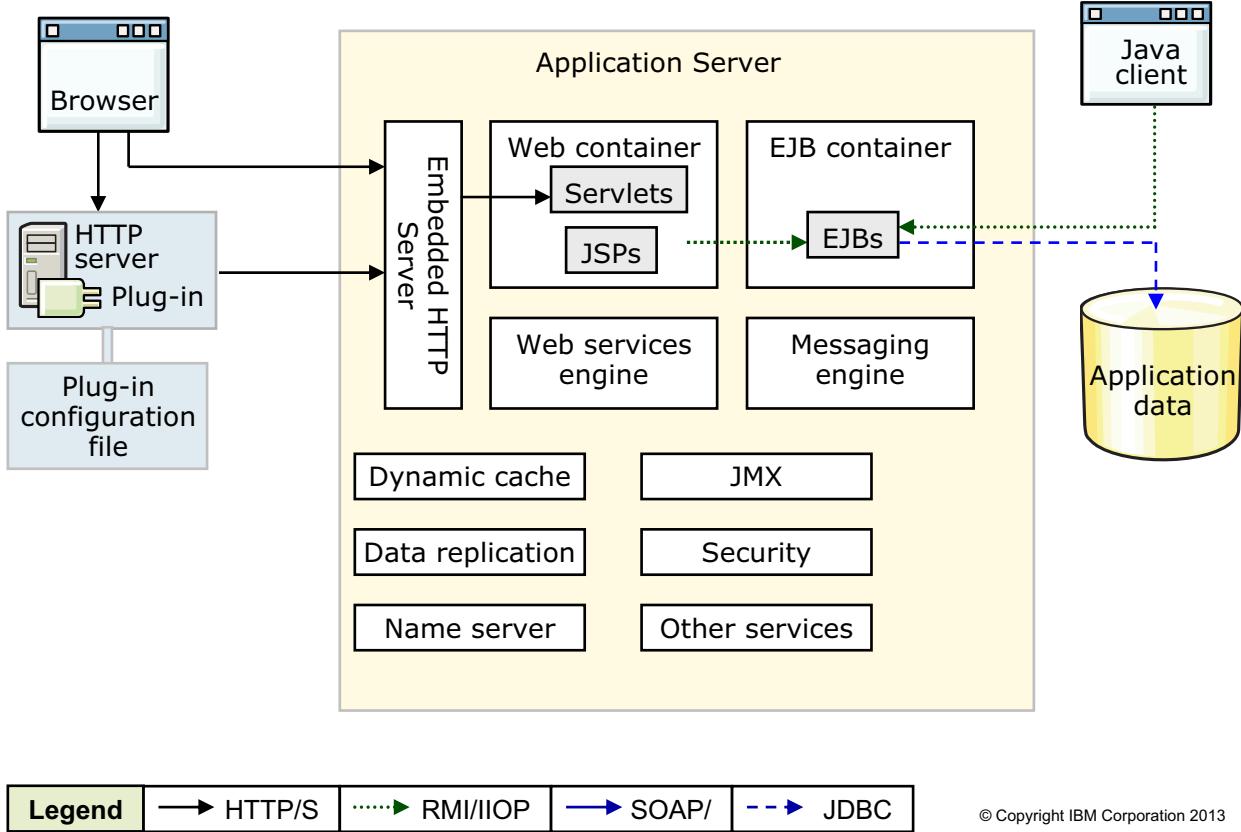
WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

7 of 10: The browser can communicate directly with the embedded HTTP server (bypassing the external web server); use this direct communication only for testing and development purposes. In this way, you access your application servers in many of the lab exercises.

WebSphere architecture run time (8 of 10)



© Copyright IBM Corporation 2013

Figure 2-13. WebSphere architecture run time (8 of 10)

WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

8 of 10: Browsers are not the only clients; a pure Java client can access EJBs directly through RMI/IOP.

WebSphere architecture run time (9 of 10)

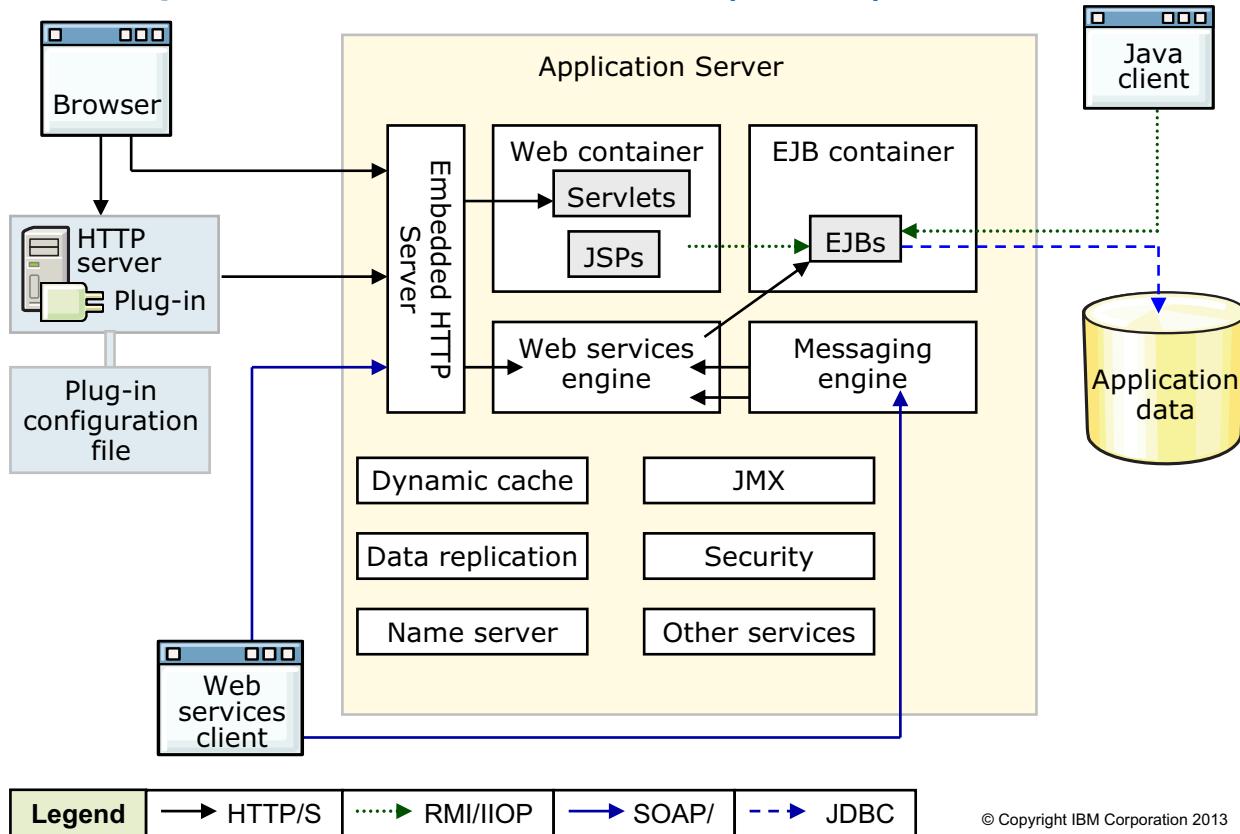


Figure 2-14. WebSphere architecture run time (9 of 10)

WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

9 of 10: Web services clients can also access your application server. This communication occurs in two ways:

- Through SOAP over HTTP and passing through the embedded HTTP server
- Through SOAP over JMS Communicating directly to the messaging engine within the application server

WebSphere architecture run time (10 of 10)

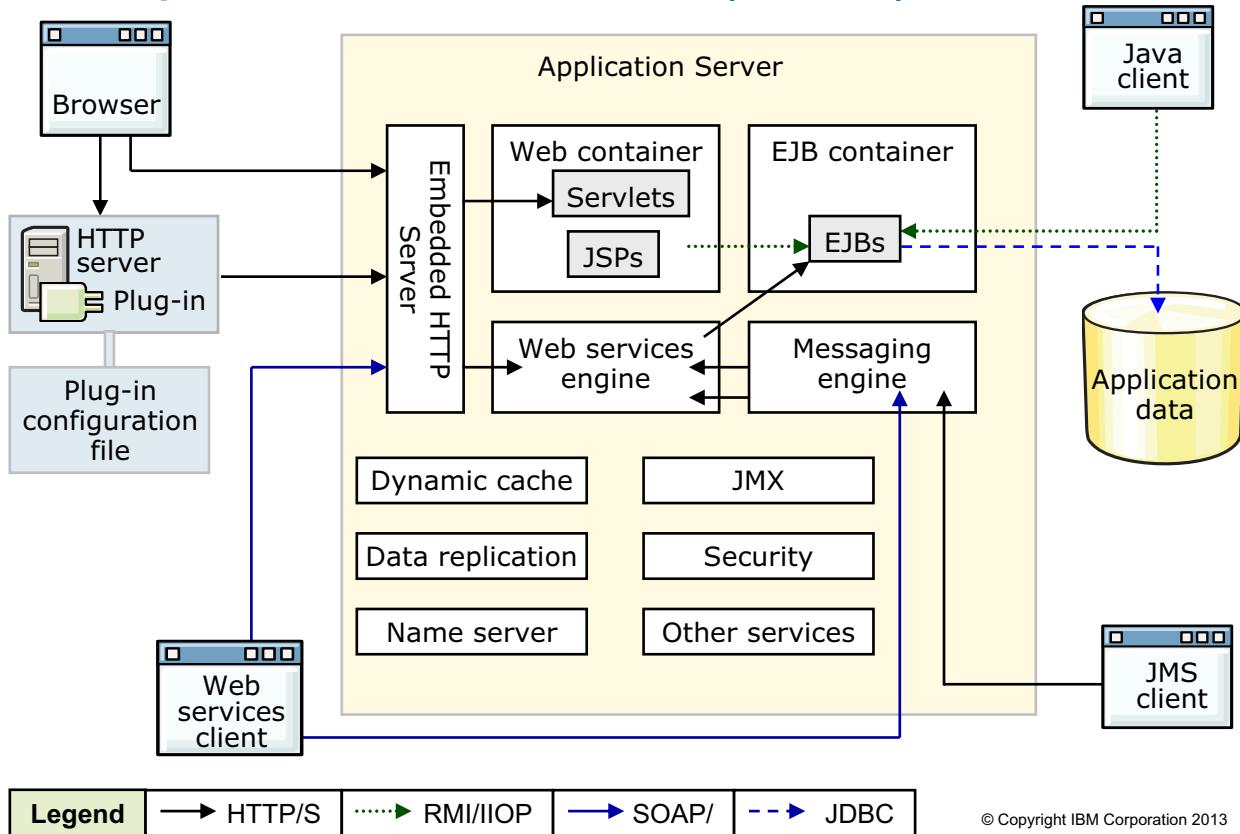


Figure 2-15. WebSphere architecture run time (10 of 10)

WA680 / VA6801.0

Notes:

This diagram illustrates the basic architecture of WebSphere Application Server, including several of the larger components.

10 of 10: Finally, you can use a JMS client to directly communicate with the messaging engine.

2.2. Architecture administration

These topics provide information about using the IBM Installation Manager.

Architecture administration



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

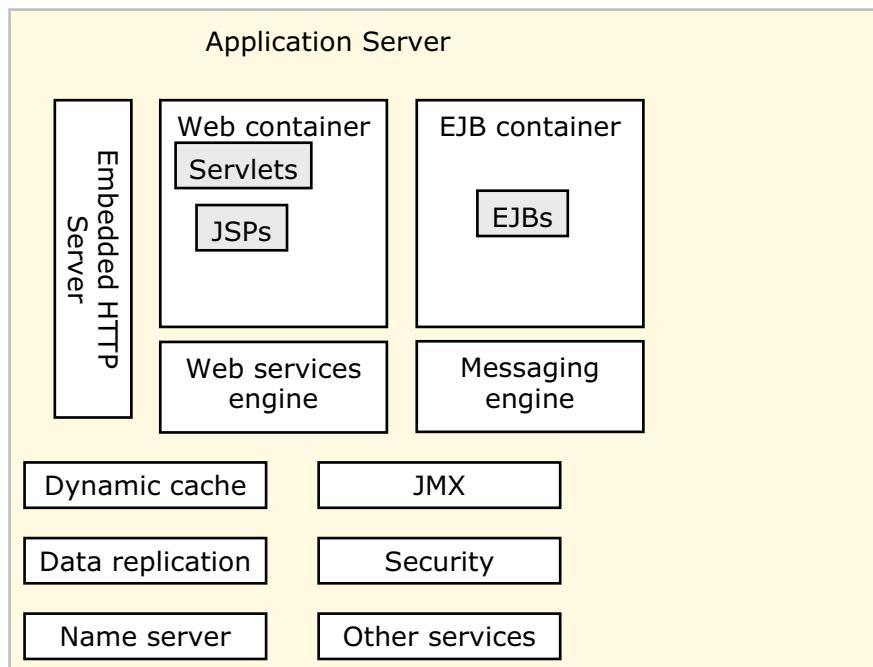
7.0

Figure 2-16. Architecture administration

WA680 / VA6801.0

Notes:

WebSphere architecture administration (1 of 4)



© Copyright IBM Corporation 2013

Figure 2-17. WebSphere architecture administration (1 of 4)

WA680 / VA6801.0

Notes:

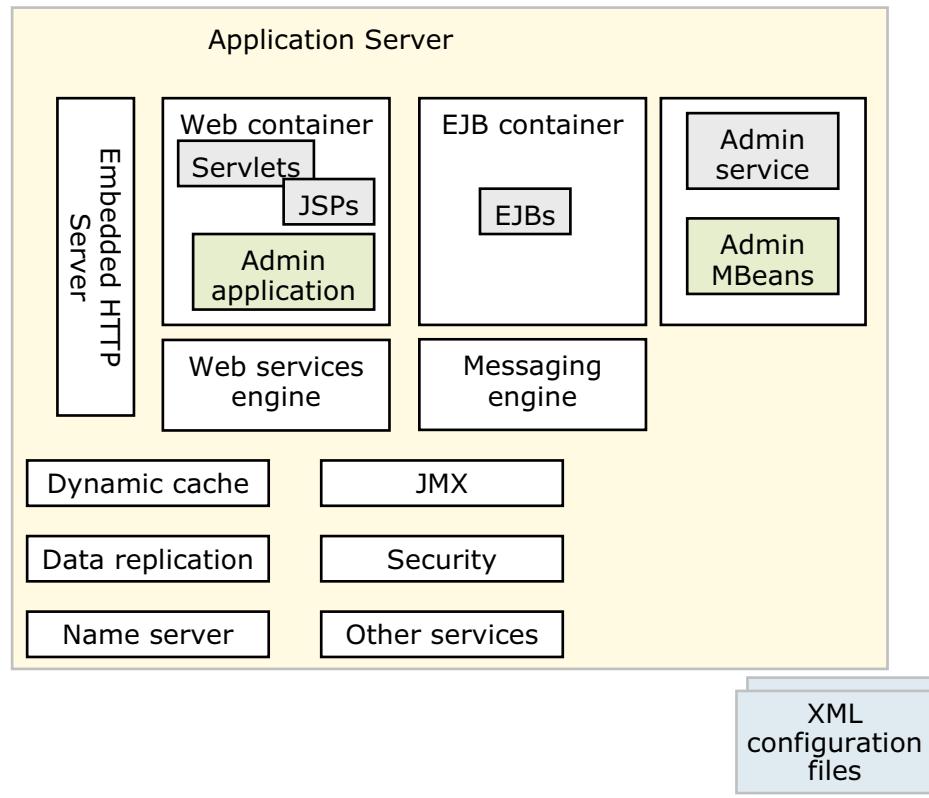
Earlier, you saw the run time depiction of a WebSphere Application Server. This diagram illustrates the basic architecture of administering WebSphere Application Server.

There are two main tools that are used to administer WebSphere Application Server: the administrative console and the wsadmin command-line tool.

The configuration of the server is stored in a set of XML files, often referred to as the configuration repository. These files define the server itself, and resources and services that it provides.

1 of 4: This diagram is a standard application server, as previously mentioned.

WebSphere architecture administration (2 of 4)



© Copyright IBM Corporation 2013

Figure 2-18. WebSphere architecture administration (2 of 4)

WA680 / VA6801.0

Notes:

2 of 4: One of the services available within the application server is the administrative service. This service allows for configuration of the application server. The files necessary for configuration are stored outside of the actual application server in a set of XML configuration files. An application that runs within the web container provides users the ability to administer the application server through a web application: the administrative console.

WebSphere architecture administration (3 of 4)

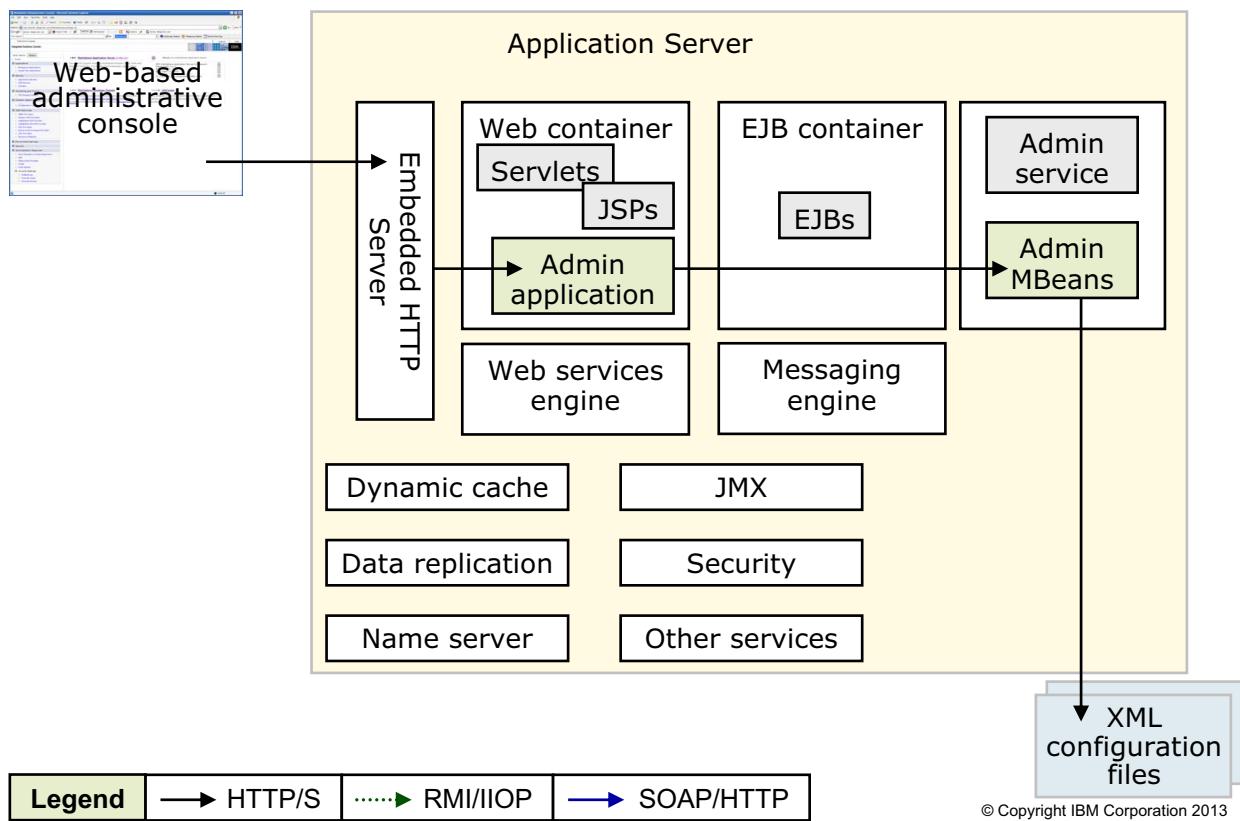


Figure 2-19. WebSphere architecture administration (3 of 4)

WA680 / VA6801.0

Notes:

3 of 4: This diagram illustrates communication from the browser to the XML configuration files.

WebSphere architecture administration (4 of 4)

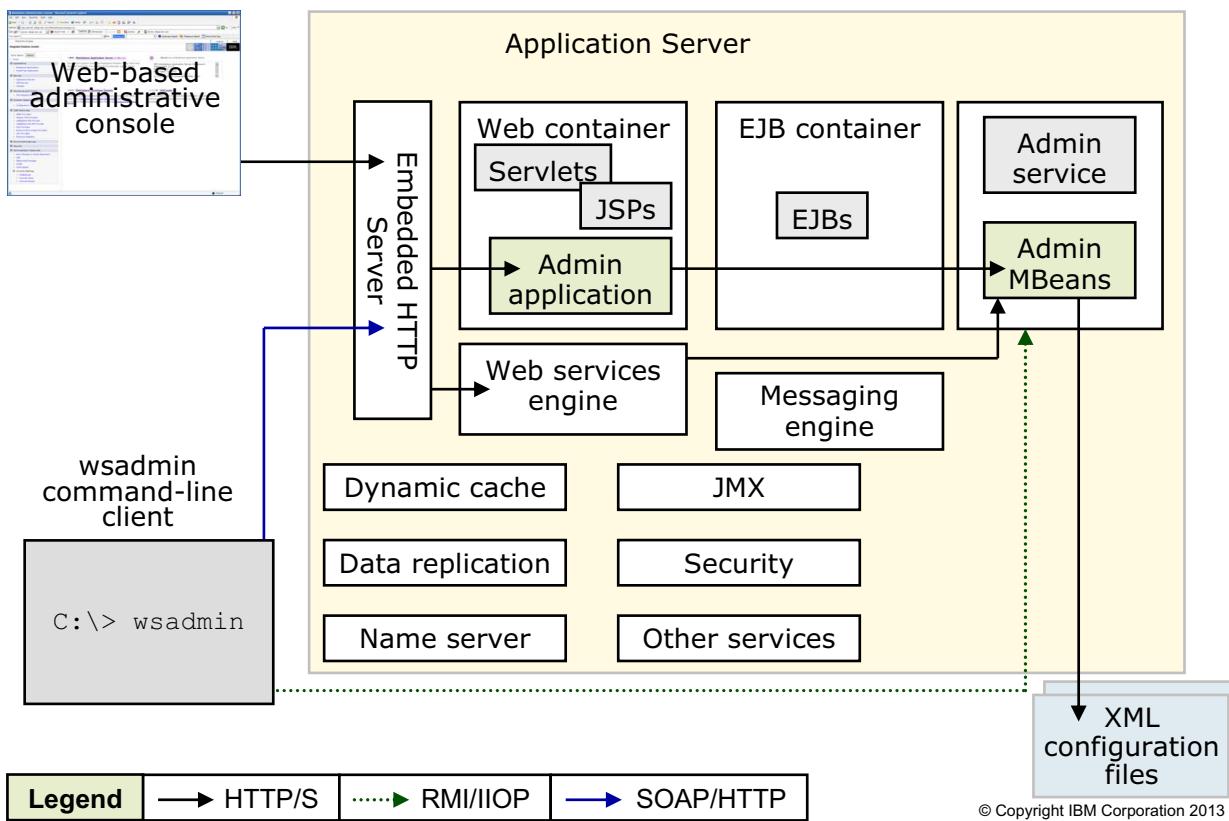


Figure 2-20. WebSphere architecture administration (4 of 4)

WA680 / VA6801.0

Notes:

4 of 4: The wsadmin command-line client is used to administer the application server through SOAP, by communicating with the embedded HTTP server, or by using RMI (the default) to communicate directly with the administrative service.

2.3. Profiles

This topic provides information about using the IBM Installation Manager.

Profiles



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 2-21. Profiles

WA680 / VA6801.0

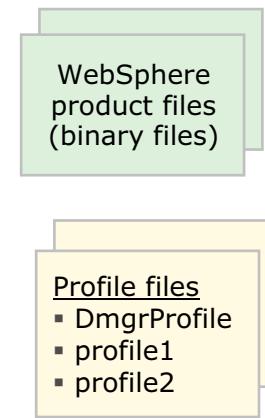
Notes:

WebSphere profile overview

Profiles are sets of files that represent a WebSphere Application Server configuration

WebSphere Application Server files are split into two categories:

- Product files
 - Set of shared read-only static files or product binary files
 - Shared among any instances of the WebSphere Application Server product
- Profiles (configuration files)
 - Set of user-customizable data files
 - Files include WebSphere configuration, installed applications, resource adapters, properties, and log files



© Copyright IBM Corporation 2013

Figure 2-22. WebSphere profile overview

WA680 / VA6801.0

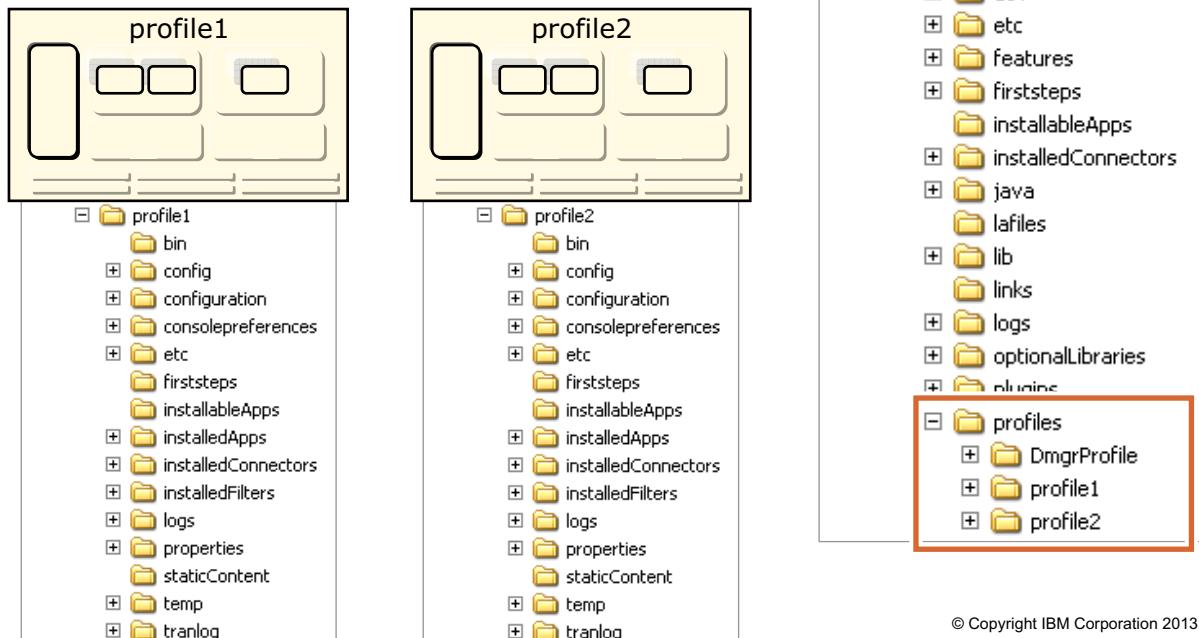
Notes:

Profiles are the configuration mechanism that allows you to run more than one application server on a single installation of WebSphere product files.

For a stand-alone server, the dmgr profile would not exist yet. The dmgr profile is presented in the next unit.

WebSphere profile benefits

- Benefits of profiles:
 - Each profile uses the same product files
 - Simpler than multiple WebSphere installations
 - Less disk space
 - Simplifies application of product updates



© Copyright IBM Corporation 2013

Figure 2-23. WebSphere profile benefits

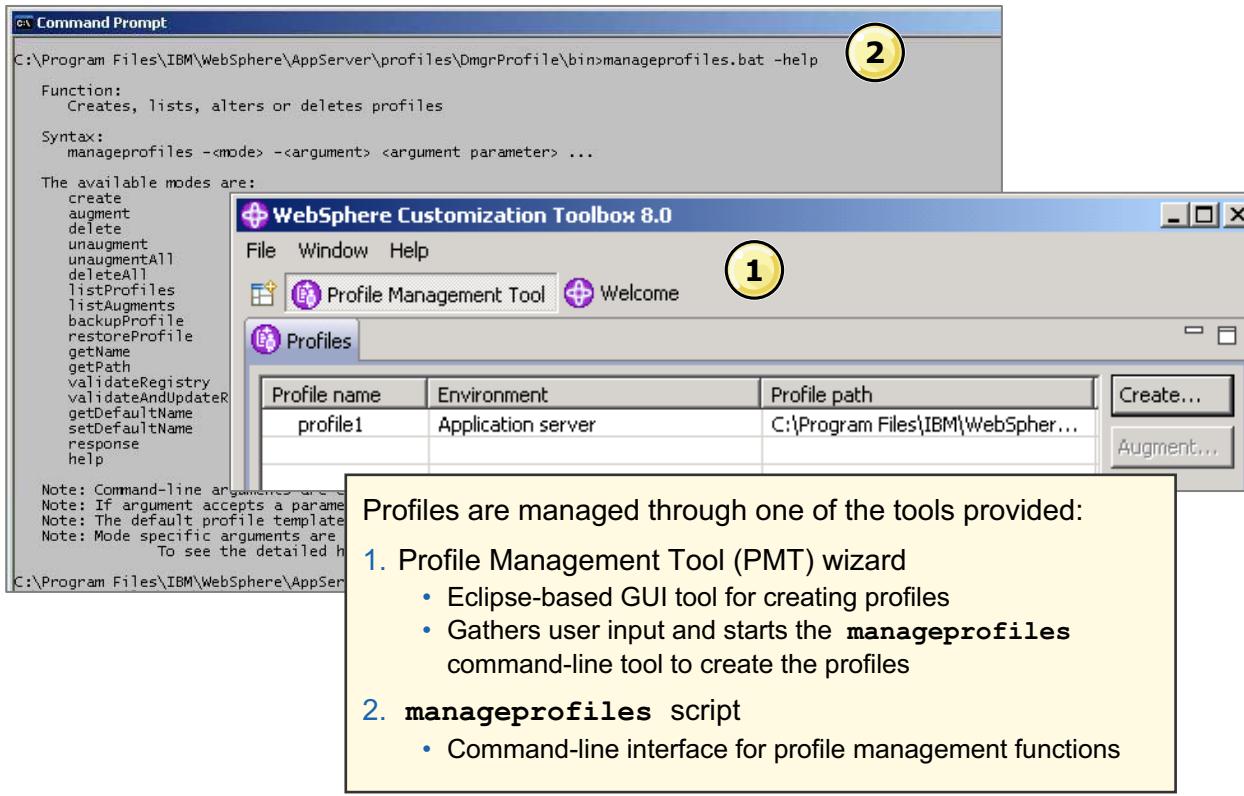
WA680 / VA6801.0

Notes:

Notice that under the WebSphere installation directory (`<was_root>`) there are subdirectories for each profile. In the example that is shown, there are two application servers that are each configured according to the files that exist within their own profile directory.



Managing profiles



© Copyright IBM Corporation 2013

Figure 2-24. Managing profiles

WA680 / VA6801.0

Notes:

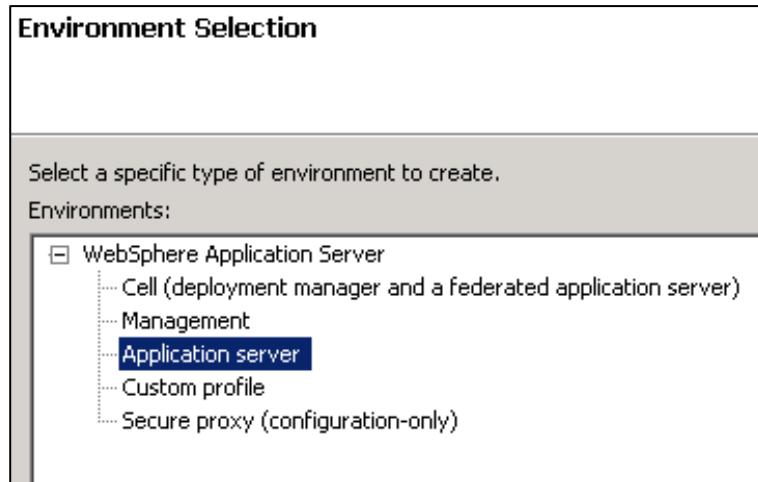
These two tools are available for creating and managing profiles. Profiles are managed through one of the tools provided:

1. Profile Management Tool (PMT) wizard is an Eclipse-based GUI tool for creating profiles. The wizard gathers user input and starts the `manageprofiles` command-line tool to create the profiles.
2. The `manageprofiles` script is run from a command-line interface for profile management functions.



Profile types

- Cell
 - Deployment manager with a federated application server
- Management
 - Administrative agent
 - Deployment manager
 - Job manager
- Application server
 - Stand-alone
- Custom profile
 - Federated node
(no application server)
- Secure proxy



© Copyright IBM Corporation 2013

Figure 2-25. Profile types

WA680 / VA6801.0

Notes:

There are numerous profile types:

- Cell
 - Deployment manager with a federated application server
- Management
 - Administrative agent
 - Deployment manager
 - Job manager
- Application server
 - Stand-alone
- Custom profile
 - Federated node
 - (No application server)

- Secure proxy

2.4. Network deployment concepts

This topic provides information about using the IBM Installation Manager.

Network deployment concepts



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

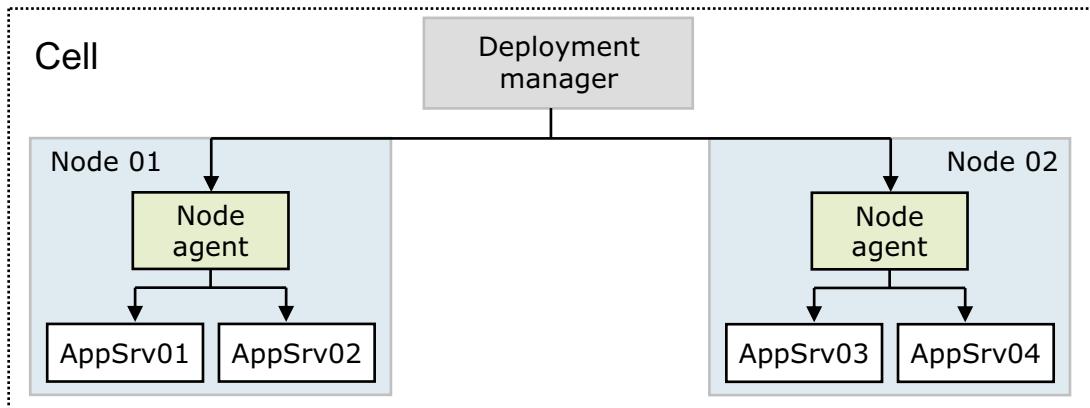
Figure 2-26. Network deployment concepts

WA680 / VA6801.0

Notes:

Network deployment concepts

- A **deployment manager** (DMgr) process manages the node agents
 - Holds the configuration repository for the entire management domain, called a cell
 - Within a cell, the administrative console runs inside the DMgr
- A **node** is a logical grouping of application servers
 - A single **node agent** process manages each node
 - Multiple nodes can exist on a single machine by using profiles



© Copyright IBM Corporation 2013

Figure 2-27. Network deployment concepts

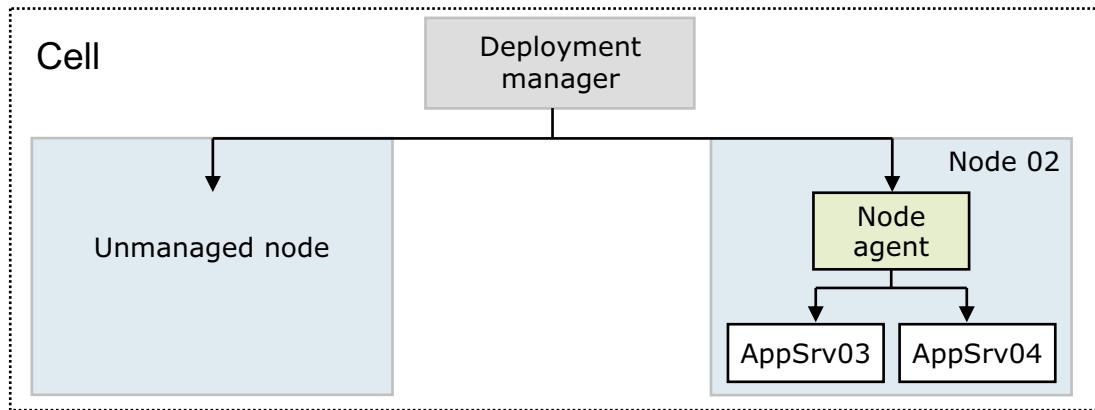
WA680 / VA6801.0

Notes:

The deployment manager is an application server that manages the administrative environment within a cell. As you see later in this unit, a node is represented as a profile. The node agent is an important process that allows for communication of administrative information, such as commands and configuration files, to reach the applications servers.

Managed versus unmanaged nodes

- A managed node is a node that contains a node agent
- An unmanaged node is a node in the cell without a node agent
 - The rest of the environment can be aware of the node
 - Useful for defining HTTP servers as part of the topology
 - Allows creation of different plug-in configurations for different HTTP servers



© Copyright IBM Corporation 2013

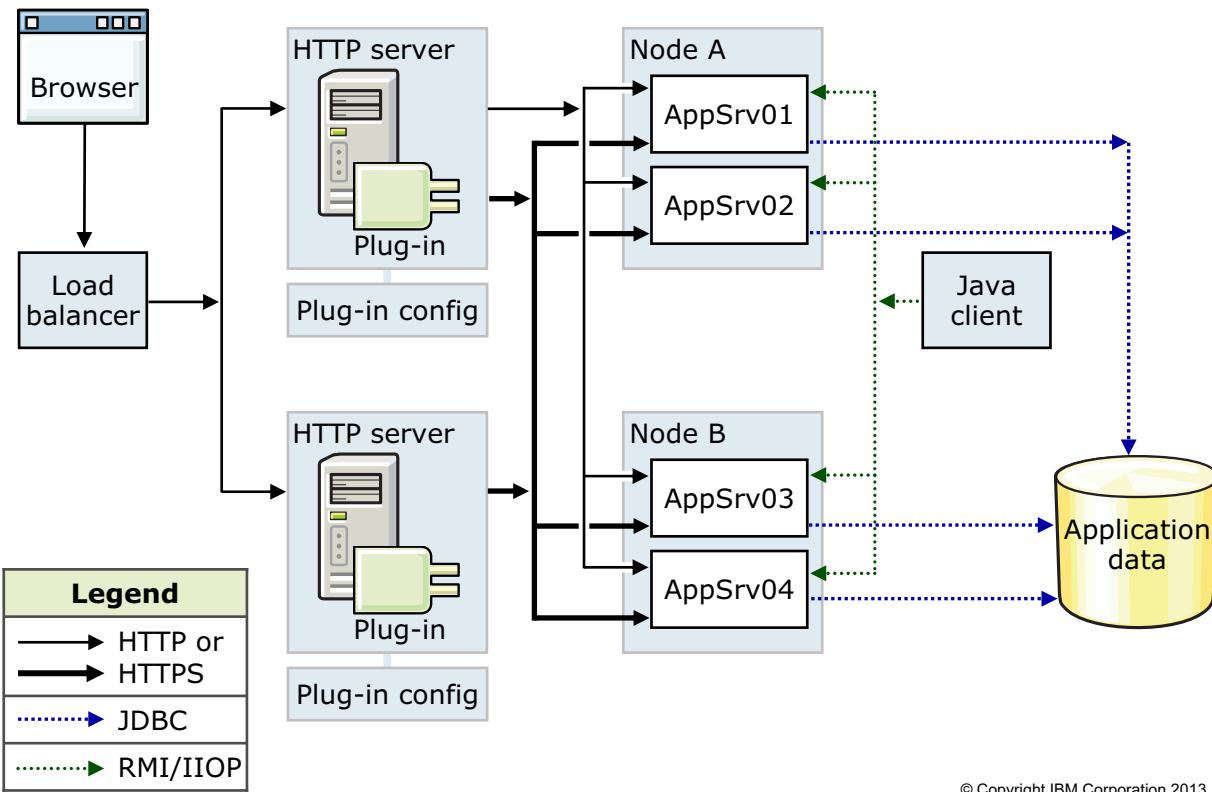
Figure 2-28. Managed versus unmanaged nodes

WA680 / VA6801.0

Notes:

A node agent is a process that handles communications with the resources within the node. An example of an unmanaged node is IBM HTTP Server.

Network deployment run time flow



© Copyright IBM Corporation 2013

Figure 2-29. Network deployment run time flow

WA680 / VA6801.0

Notes:

The main theme with Network Deployment is distributed applications. While the "flow" of an application remains the same, there are significant additions to the run time of an application. Note the "load balancer": it allows for multiple HTTP servers. Users point their browsers to the load balancer and their requests are workload managed to an HTTP server. When a request reaches one of these HTTP servers, the HTTP server plug-in load balances the request between the application servers that it is configured to serve. When the request enters the application server, the flow is identical to how it was in Express and Base.

The Java client requests to EJBs can also be workload managed so that the requests do not all hit one application server.

Administration flow

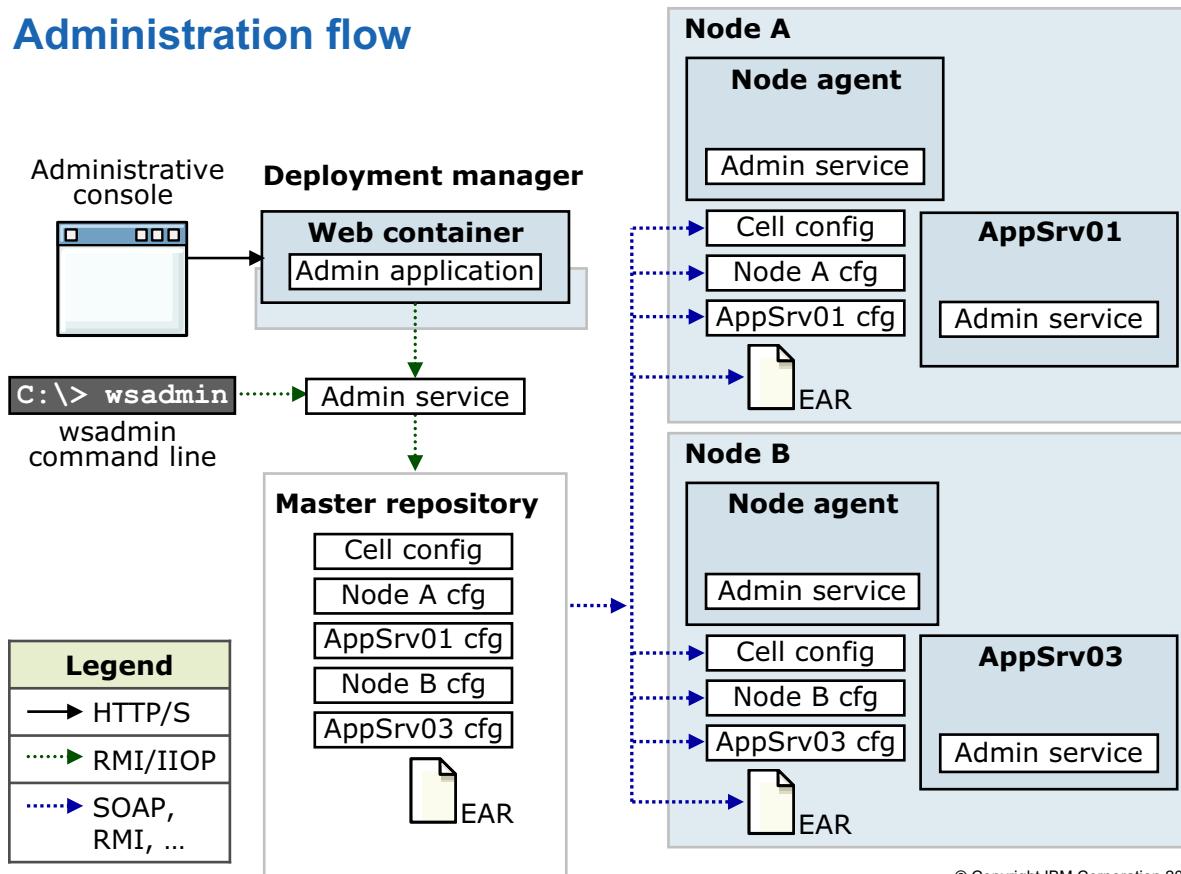


Figure 2-30. Administration flow

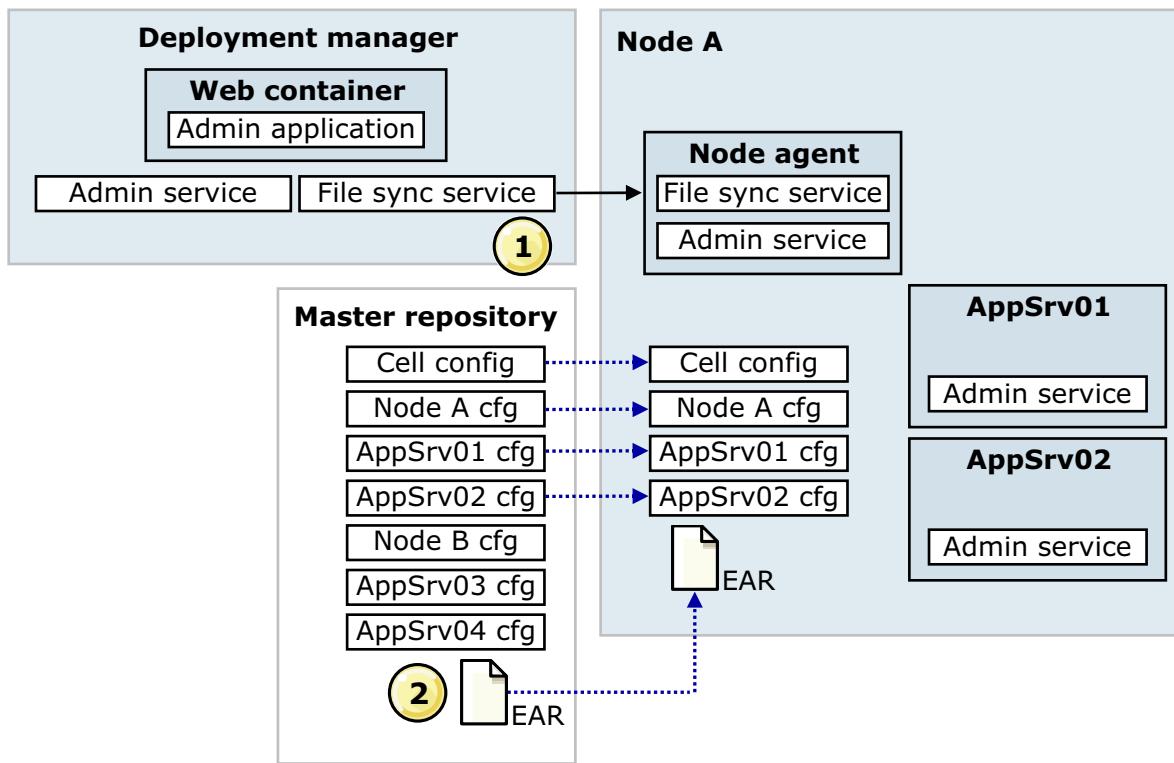
WA680 / VA6801.0

Notes:

The administrative console and wsadmin are the two ways that the environment is administered. However, these tools communicate with the deployment manager and **not** with the application servers directly. The communication of these commands flows from the tools to the deployment manager, then to the node agents, and then to the application servers. This flow allows for the administration of multiple nodes from a single focal point (the deployment manager). Each node can possibly contain multiple application servers.

There is **one** main (master) repository for the configuration files within a cell, and those files are associated with the deployment manager. All updates to the configuration files go through the deployment manager. Be careful about directly connecting to an application server with wsadmin or the administrative console. Any changes to the configuration files are only temporary and are overwritten with the configuration files from the master files (repository).

File synchronization



© Copyright IBM Corporation 2013

Figure 2-31. File synchronization

WA680 / VA6801.0

Notes:

Each managed process, node agent, and deployment manager starts with its own set of configuration files. The deployment manager contains the master configuration. Any changes at the node agent or server level are local, and the MASTER configuration overrides them at the next synchronization (update).

1. Node agents synchronize their files with the master copy either automatically or manually. Automatic synchronization can be done at startup or scheduled periodically. Manual synchronization is done with the administrative console or from the command line.
2. During synchronization, the node agent asks for changes to master configuration. Any new or updated files are copied to the node.

WebSphere Network Deployment profiles

Benefits of profiles in network deployment

- Think of profiles as representing a node
- Can install multiple profiles on a single host

All profiles use the same product files

- **Application server** profile (stand-alone)
 - Equivalent to Base or Express application server
 - Has a node name and a cell name property, and corresponding directories
 - Cell directory is overwritten upon federation
- **Deployment manager** profile
 - Creates a deployment manager
- **Custom** profile (managed)
 - Creates a managed node, which, by default, is federated into a cell
 - Creates a node agent, but no application servers
- **Cell** profile
 - Creates both a deployment manager and a federated node
- Others

© Copyright IBM Corporation 2013

Figure 2-32. WebSphere Network Deployment profiles

WA680 / VA6801.0

Notes:

The addition of Network Deployment to this discussion does not change the definition of a profile. The WebSphere configuration is still built by creating profiles, which consist of product binary files and configuration files. The profile that was previously listed as a "stand-alone node" is listed here as an "application server". The deployment manager profile is added, which is a special type of node that manages the administrative domain of a cell.

2.5. Advanced concepts

This topic provides information about using the IBM Installation Manager.

Advanced concepts



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 2-33. Advanced concepts

WA680 / VA6801.0

Notes:

Flexible management

- Loose management coupling
- Coordinates management across a group of endpoints
 - One job to install application across a number of nodes
- Can manage through administrative agent or deployment manager

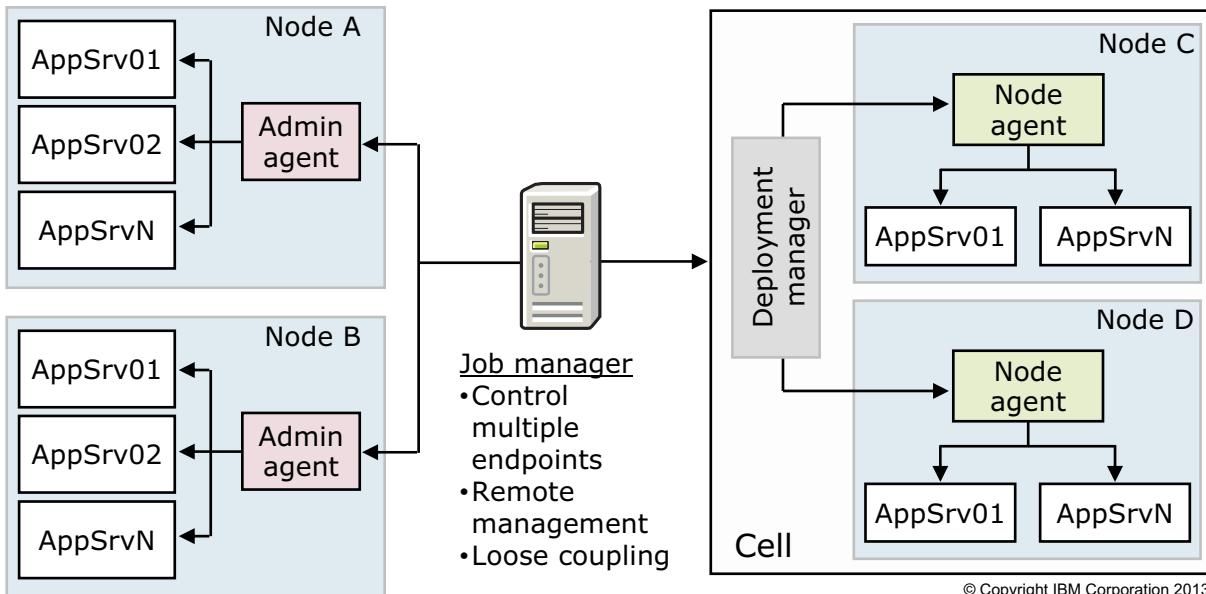


Figure 2-34. Flexible management

WA680 / VA6801.0

Notes:

Flexible management is an approach that allows an administrator to manage multiple application servers or cells through a loose asynchronous interface. Flexible management is covered later in this course.



Centralized Installation Manager (CIM)

- Simplifies the installation and maintenance of application servers within a Network Deployment cell
- Pushes remote binary files or maintenance to remote targets
- Starts the standard or update installer to complete the installation of the update
- Allows you to:
 - Download interim fixes and fix packs from IBM support directly to the CIM repository
 - Install interim fixes and fix packs on target nodes with the Network Deployment cell
 - Monitor download and installation status of packages through the administrative console

© Copyright IBM Corporation 2013

Figure 2-35. Centralized Installation Manager (CIM)

WA680 / VA6801.0

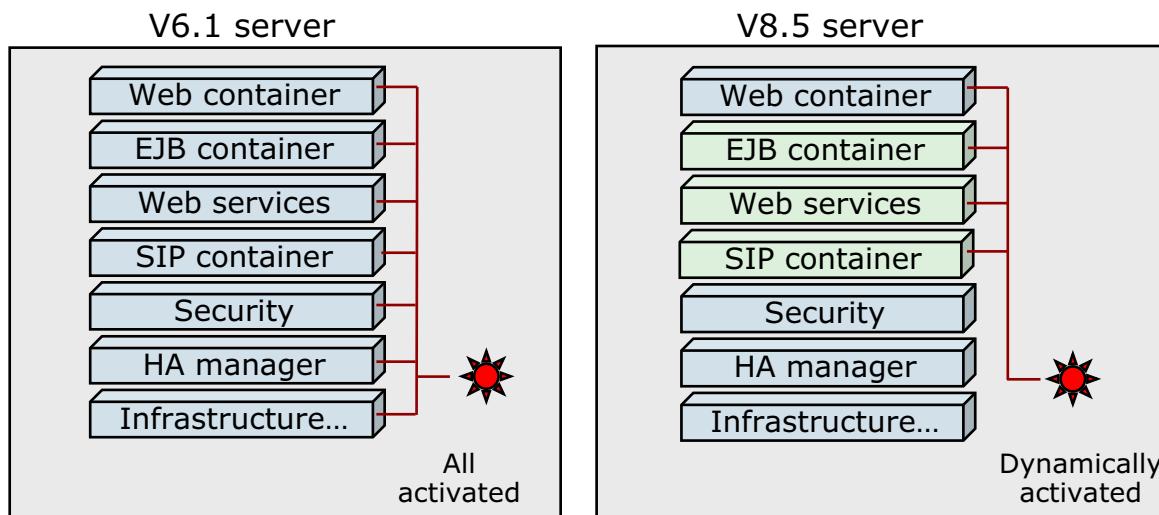
Notes:

The CIM pushes the product binary files or maintenance to the remote targets and starts the standard installer or update installer tool to install or update the targets, allowing you to:

- Download interim fixes and fix packs from IBM support directly to the CIM repository
- Install interim fixes and fix packs on target nodes with the Network Deployment cell
- Monitor download and installation status of packages through the administrative console

Intelligent run time provisioning

- Dynamic start of application server components that are based on application needs
- Reduces the run time footprint; less memory required
- Can significantly reduce startup times



© Copyright IBM Corporation 2013

Figure 2-36. Intelligent run time provisioning

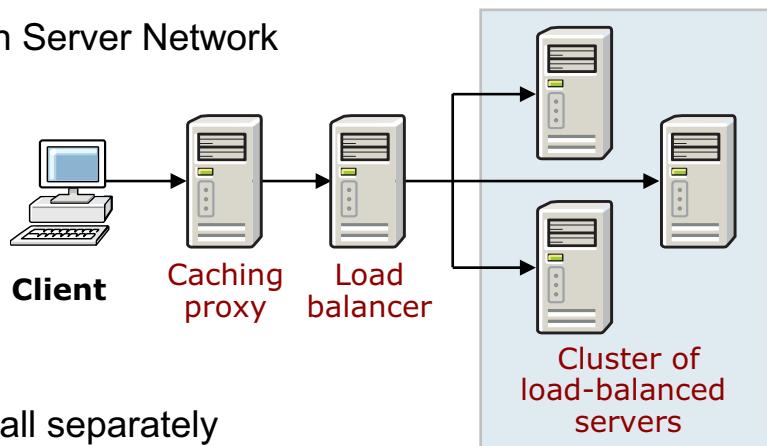
WA680 / VA6801.0

Notes:

Intelligent run time provisioning allows the application server to start faster and with less memory because it loads only those components that are required. As other services are needed, they are loaded on demand.

Edge Components

- WebSphere Application Server Network Deployment package contains the following Edge Components functions:
 - Load balancer
 - Caching proxy
- Edge Components install separately from WebSphere Application Server
- Load balancer is responsible for balancing the load across multiple servers that can be within either local area networks or wide area networks
- Purpose of caching proxy is to reduce network congestion within an enterprise by offloading security and content delivery from web servers and application servers



© Copyright IBM Corporation 2013

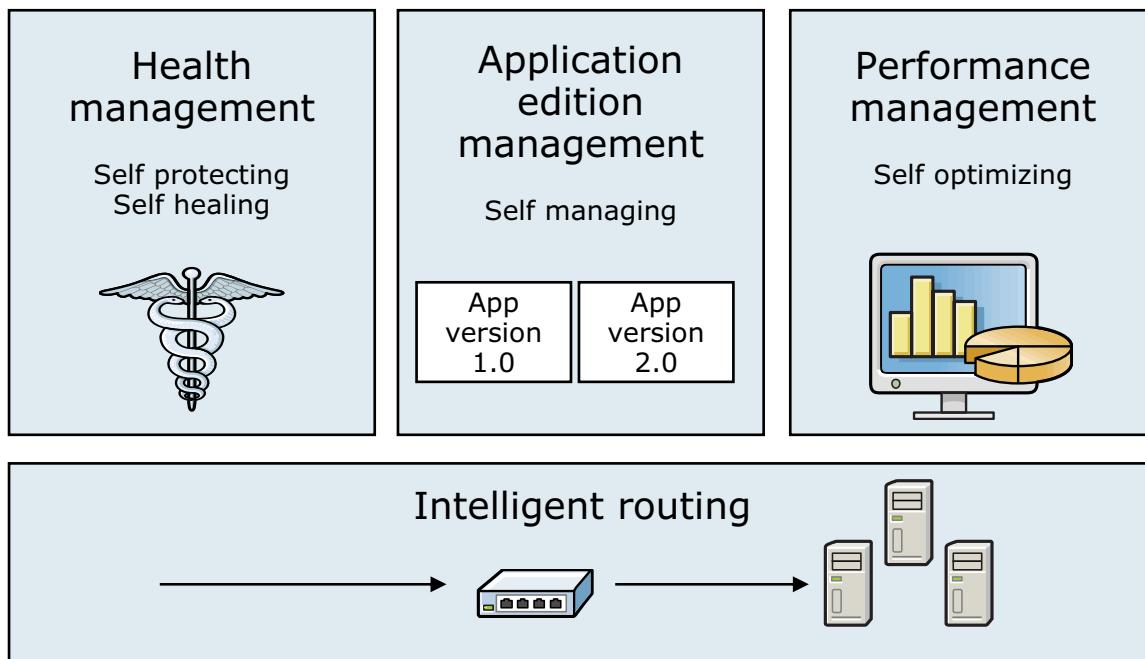
Figure 2-37. Edge Components

WA680 / VA6801.0

Notes:

Edge Components are included the Network Deployment package. The Edge Components include a load balancer and a caching proxy. The load balancer distributes incoming client requests across servers, balancing workload and providing high availability by routing around unavailable servers. The caching proxy can satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the content host. Cacheable content includes static web pages and JSP files with dynamically generated but infrequently changed fragments.

Intelligent Management Pack



© Copyright IBM Corporation 2013

Figure 2-38. Intelligent Management Pack

WA680 / VA6801.0

Notes:

Intelligent Management provides a virtualized infrastructure that redefines the traditional concepts of Java Platform, Enterprise Edition (Java EE) resources and applications and their relationships with one another. This application infrastructure virtualization facilitates the product's ability to automate operations in an optimal manner, increasing the quality of service. By introducing an automated operating environment with workload management, you can reduce total cost of ownership by performing more work that requires less hardware.



Unit summary

Having completed this unit, you should be able to:

- Describe the overall architecture of WebSphere Application Server V8.5
- Explain the administration model
- Identify basic command-line commands that are used for administration

© Copyright IBM Corporation 2013

Figure 2-39. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions (1 of 2)

1. Which of the following provides an environment for running servlets?
 - A. Client module
 - B. Web container
 - C. EJB module

2. Which of the following are components contained within the JVM of the application server?
 - A. HTTP Server plug-in
 - B. Embedded HTTP Server
 - C. DB2 database

© Copyright IBM Corporation 2013

Figure 2-40. Checkpoint questions (1 of 2)

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

Checkpoint questions (2 of 2)

3. A process that handles communications with the resources within the node is _____.
4. What is the process when the node agent checks for changes to the master configuration?
5. What defines the run time environment for either the deployment manager or the application server?

© Copyright IBM Corporation 2013

Figure 2-41. Checkpoint questions (2 of 2)

WA680 / VA6801.0

Notes:

Write your answers here:

3.

4.

5.



Checkpoint answers (1 of 2)

1. Which of the following provides an environment for running servlets?
B. Web container

2. Which of the following are components contained within the JVM of the application?
B. Embedded HTTP Server

© Copyright IBM Corporation 2013

Figure 2-42. Checkpoint answers (1 of 2)

WA680 / VA6801.0

Notes:



Checkpoint answers (2 of 2)

3. A process that handles communications with the resources within the node is the _____.

Node agent

4. What is the process when the node agent checks for changes to the master configuration?

File synchronization

5. What defines the run time environment for either the deployment manager or the application server?

Profiles

© Copyright IBM Corporation 2013

Figure 2-43. Checkpoint answers (2 of 2)

WA680 / VA6801.0

Notes:

Unit 3. WebSphere Application Server scripting facilities

What this unit is about

This unit introduces the tools, languages, Administrative objects, and APIs that comprise the scripting facilities in WebSphere Application Server.

What you should be able to do

After completing this unit, you should be able to:

- Describe the scripting tools that are available in WebSphere Application Server V8.5
- List the supported scripting languages
- Identify the administrative objects
- Explain the WebSphere Application Server configuration model

Unit objectives

After completing this unit, you should be able to:

- Introduce the scripting tools available in WebSphere Application Server V8.5
- List the supported scripting languages
- Identify the Administrative objects
- Explain the WebSphere Application Server configuration model

© Copyright IBM Corporation 2008,2013

Figure 3-1. Unit objectives

WA680 / VA6801.0

Notes:

WebSphere Application Server scripting support

Script Development

Script Execution

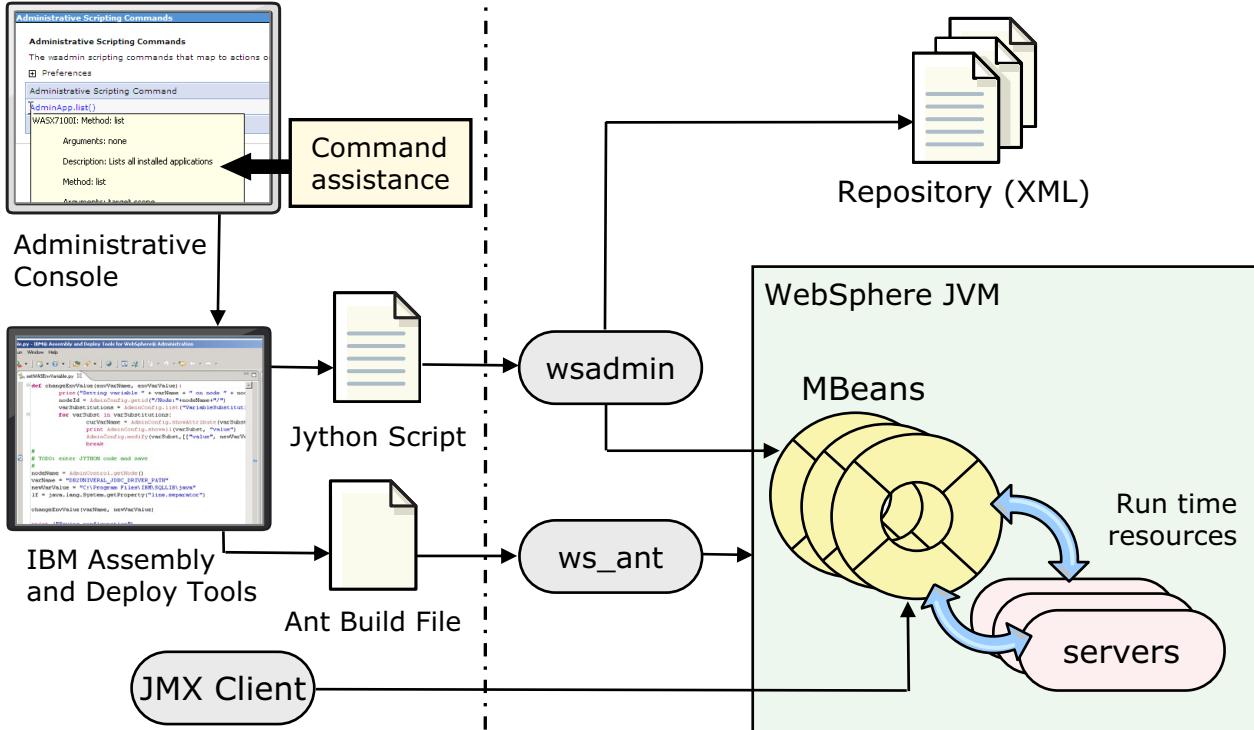


Figure 3-2. WebSphere Application Server scripting support

WA680 / VA6801.0

Notes:

This diagram displays the tools, application programming interfaces (APIs), and software components that WebSphere Application Server provides to support scripting. They can be grouped in two general categories: script development and script execution. Script development can be done with The IBM Assembly and Deploy Tools (IADT). wsadmin is the central tool for script execution. wsadmin has a collection of administrative objects. These administrative objects allow scripts to access the JMX MBeans. The JMX beans manage runtime resources inside the application server and the configuration repository files. ws_ant provides the means to group and run individual scripted or command-line driven tasks for added automation. Each of these facilities is in more detail in the subsequent slides.



Prerequisite knowledge for script development

- WebSphere Application Server configuration model
 - Defines the available configuration types and their organization
- wsadmin tool
 - Provides the interface for running scripts
- Jython syntax
 - Represents the language for writing scripts
- Scripting (administrative) objects
 - Represent the management objects that are used in scripts to initiate changes
- wsadmin is a powerful but fairly complex scripting interface. A good understanding of these prerequisites is required to use wsadmin effectively.

© Copyright IBM Corporation 2008,2013

Figure 3-3. Prerequisite knowledge for script development

WA680 / VA6801.0

Notes:

Assembly and deployment tools

- IBM Assembly and Deploy Tools for WebSphere Administration is included with WebSphere Application Server V8.5
 - A subset of Rational Application Developer for the rapid assembly and deployment of modules for WebSphere
 - License for IBM Assembly and DeployTools does not expire
 - Runs on Windows, Linux, AIX, and Oracle (Solaris)
 - Replaces the V7 Assembly and Deploy toolkit
 - WebSphere Application Server license fully licenses and supports assembly and deployment functions
- Rational Application Developer for WebSphere Software is available on a trial basis for a limited time
 - Available at:
<http://www.ibm.com/developerworks/downloads/r/rad/>

© Copyright IBM Corporation 2008,2013

Figure 3-4. Assembly and deployment tools

WA680 / VA6801.0

Notes:

The IBM Assembly and Deploy Tools (IADT) is included with WebSphere Application Server 8.5. IADT is a subset of tools that are available in the Rational Application Developer.



IBM Assembly and DeployTools for WebSphere Administration

- Tool to assemble and configure enterprise applications
 - Based on Eclipse
 - Subset of IBM Rational Application Developer for WebSphere
- Opening IBM Assembly and Deploy Tools requires pointing to a workspace folder
 - Files and metadata are kept in the workspace
 - Create a workspace by pointing to an empty folder
 - Thereafter open the workspace folder
 - Guideline: one workspace per enterprise application

© Copyright IBM Corporation 2008,2013

Figure 3-5. IBM Assembly and DeployTools for WebSphere Administration

WA680 / VA6801.0

Notes:

IBM Assembly and DeployTools tasks

- Create and configure Java EE enterprise applications (EAR files):
 - Build from scratch
 - Java EE modules
- Generate and modify deployment descriptor information
- Generate and modify binding information attributes
- Generate and modify the IBM extension attributes
- Deploy applications to a remote server
- Create, debug, and run Jython scripts
- Import command assistance logs from the console in to Jython scripts
- View, analyze, and correlate log files

© Copyright IBM Corporation 2008,2013

Figure 3-6. IBM Assembly and DeployTools tasks

WA680 / VA6801.0

Notes:

There are many activities that can be done with the IADT. This course focuses primarily on tools for working with Jython scripts.

Features in IBM Assembly and DeployTools for WebSphere Administration (1 of 2)

- Tools for publishing server-side code on WebSphere V8.5
- Automated deployment descriptor generation and visual editors
- Jython scripting editor and debugger
- Java Platform, Enterprise Edition and XML form-based deployment descriptor and binding editors
- Tools for assembling and deploying OSGi applications, bundles, fragments, and composites.
- Tools for JAX-RPC web service import, discovery, and deployment
- Tools for JAX-WS web service deployment; policy set association and binding

© Copyright IBM Corporation 2008,2013

Figure 3-7. Features in IBM Assembly and DeployTools for WebSphere Administration (1 of 2)

WA680 / VA6801.0

Notes:

Features in IBM Assembly and DeployTools for WebSphere Administration (2 of 2)

- XML deployment descriptor, binding editors, and code validators
- Tools for importing, exporting, assembling, and deploying Session Initiation Protocol (SIP) applications
- Tools for assembling and deploying portlet applications
- Tools for adding database access to your applications, including built-in support and JDBC providers for many supported databases
- EJB deployment
- Enhanced EAR editor

© Copyright IBM Corporation 2008,2013

Figure 3-8. Features in IBM Assembly and DeployTools for WebSphere Administration (2 of 2)

WA680 / VA6801.0

Notes:



Create, test, and debug Jython scripts

Rational Application Developer Assembly and Deploy provides:

- Full support for Jython projects and source files
- Ability to create, test, and debug Jython projects and files
- Syntax colored editor and command completion tools
- A source outline view
- Source level debugging

```

def changeEnvValue(envVarName, envVarValue):
    print("Setting variable " + varName + " on node " + nodeName)
    nodeId = AdminConfig.getId("/Node:" + nodeName + "/")
    varSubstitutions = AdminConfig.list("VariableSubstitution")
    for varSubst in varSubstitutions:
        curVarName = AdminConfig.showAttribute(varSubst, "name")
        print AdminConfig.showAll(varSubst, "value")
        AdminConfig.modify(varSubst, [{"value": newVarValue}])
        break
    #
    # TODO: enter JYTHON code and save
    #
    nodeName = AdminControl.getNode()
    varName = "DB2UNIVERSAL_JDBC_DRIVER_PATH"
    newValue = "C:\Program Files\IBM\SQLLIB\java"
    if = java.lang.System.getProperty("line.separator")
    changeEnvValue(varName, newValue)

    print ("Saving configuration")
    AdminConfig.save()

```

Figure 3-9. Create, test, and debug Jython scripts

WA680 / VA6801.0

Notes:



Start the Jython debugger

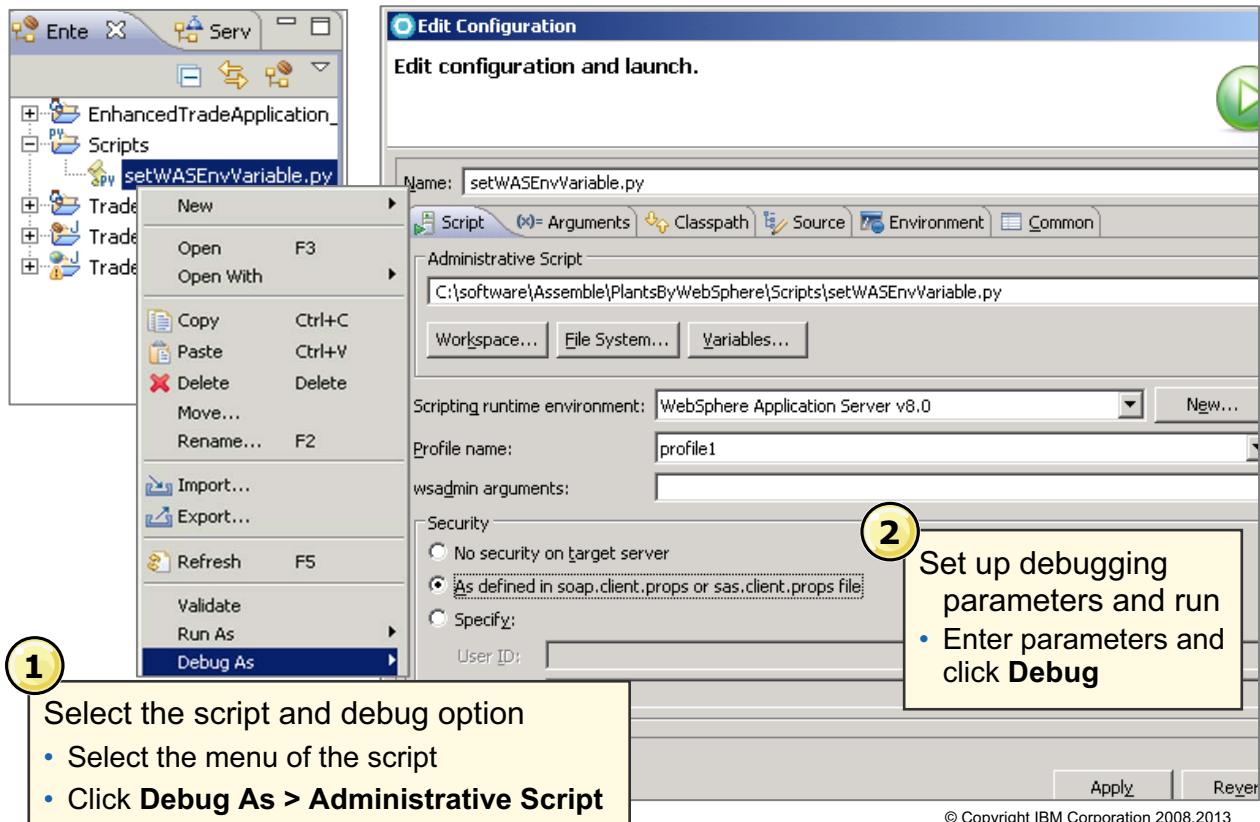


Figure 3-10. Start the Jython debugger

WA680 / VA6801.0

Notes:

The IADT includes a robust environment for running and debugging Jython scripts for WebSphere Administration.



Jython debugger perspective

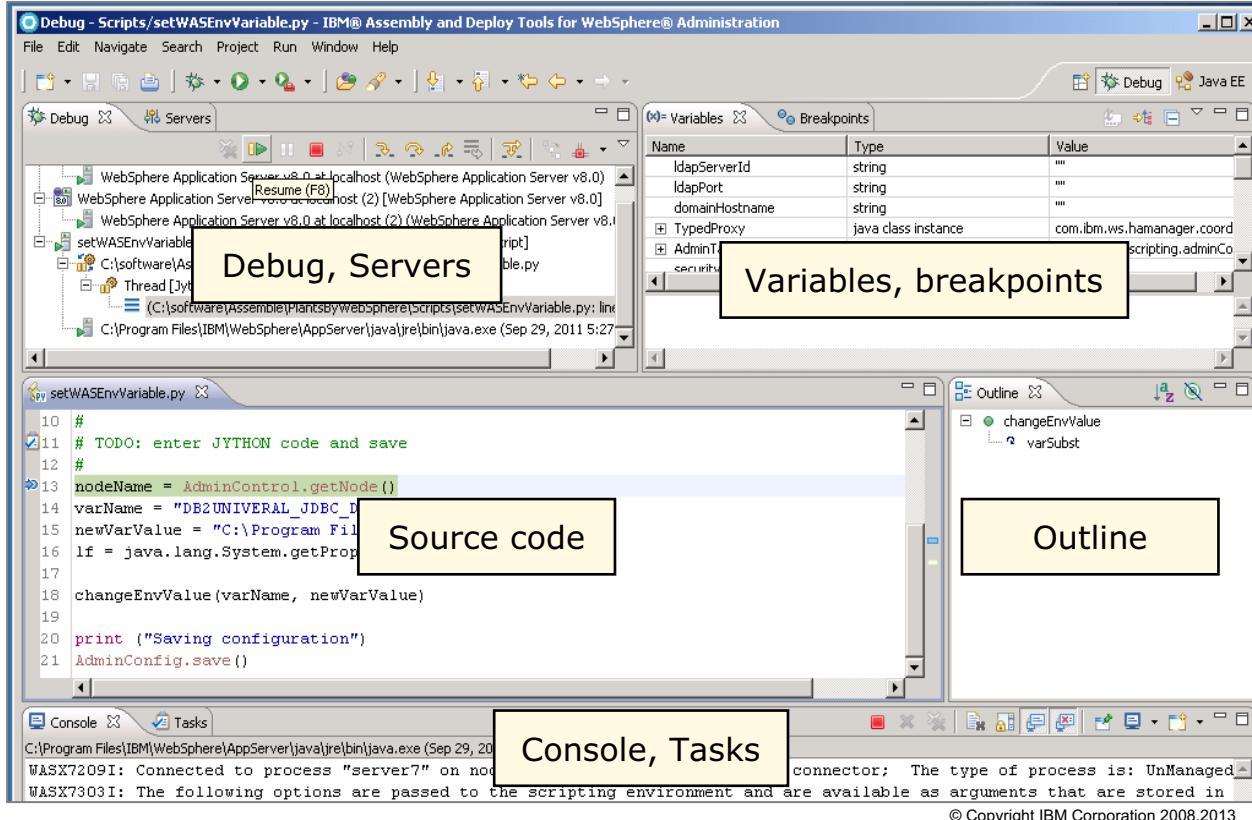


Figure 3-11. Jython debugger perspective

WA680 / VA6801.0

Notes:

The screenshot shows the WebSphere Education interface. At the top left is the "WebSphere Education" logo. At the top right is the IBM logo. Below the header, the title "Command assistance" is displayed. To the right of the title is a detailed view of administrative scripting commands. A specific command, "adminApp.list()", is highlighted in yellow. This command is described as "WASX7100I: Method: list" with "Arguments: none". It is also described as "Method: list" with "Arguments: target scope". A tooltip for this command provides additional information: "Description: List installed applications on a given target scope".

- Works in concert with the administrative console
 - Last run commands are made available to Rational Application Developer
 - Commands can be pasted directly to Jython scripts
- Administrative console access
 - Under **Help**, click **View administrative scripting command for last action**
 - The last command run is displayed
 - Place the cursor over the command to get command information
 - Command can be copied into a Jython script



Figure 3-12. Command assistance

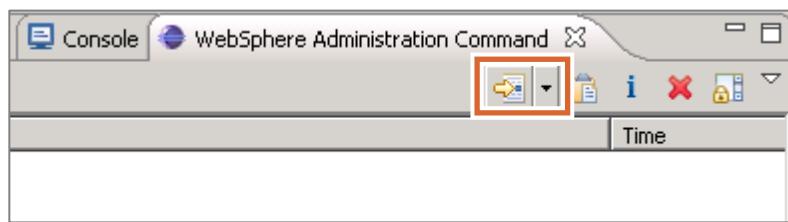
WA680 / VA6801.0

Notes:

A useful feature is the ability to do tasks in the administrative console and then export those tasks as commands. These commands can be refined into a script.

Using command assistance within IBM Assembly and DeployTools

- Command assistance setup includes the following steps:
 - Add the WebSphere Administration Command view:
Click **Window > Show View > Other > Server > WebSphere Administration Command**
 - Open the WebSphere Administration Command view
 - Use **Select Server to Monitor** to connect to the server



- Select the command that you want
- Use **Insert into Editor** to copy the command to a Jython script file

© Copyright IBM Corporation 2008,2013

Figure 3-13. Using command assistance within IBM Assembly and DeployTools

WA680 / VA6801.0

Notes:

Script execution: wsadmin

- wsadmin runs scripts and commands to configure and manage WebSphere Application Server
 - Command-line tool
 - Can be used interactively, one or more commands at a time or with script files
 - Can operate with or without a connection to a server process

- Supports two scripting languages:
 - Jacl (Java Application Control Language)
 - Deprecated in V6.1
 - Jython
 - Object-oriented, similar to Java
 - Strategic language of choice for wsadmin

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin
WASX7209I: Connected to process "server1" on node waschost00Node01 using SOAP connector; The type of process is: UnManagedProcess
WASX7029I: For help, enter: "$Help help"
wsadmin>
```

© Copyright IBM Corporation 2008,2013

Figure 3-14. Script execution: wsadmin

WA680 / VA6801.0

Notes:

You should use the Jython language for writing new administrative scripts.

Script execution: wsadmin administrative objects (1 of 2)

- AdminTask
 - Provides task-oriented commands to conduct common configuration and operational changes
- AdminConfig
 - Use to create or change static configuration objects
- AdminApp
 - Used to install, modify, or administer applications
- AdminControl
 - Used to govern, and trace objects that are running.
- Help
 - Provides help

© Copyright IBM Corporation 2008,2013

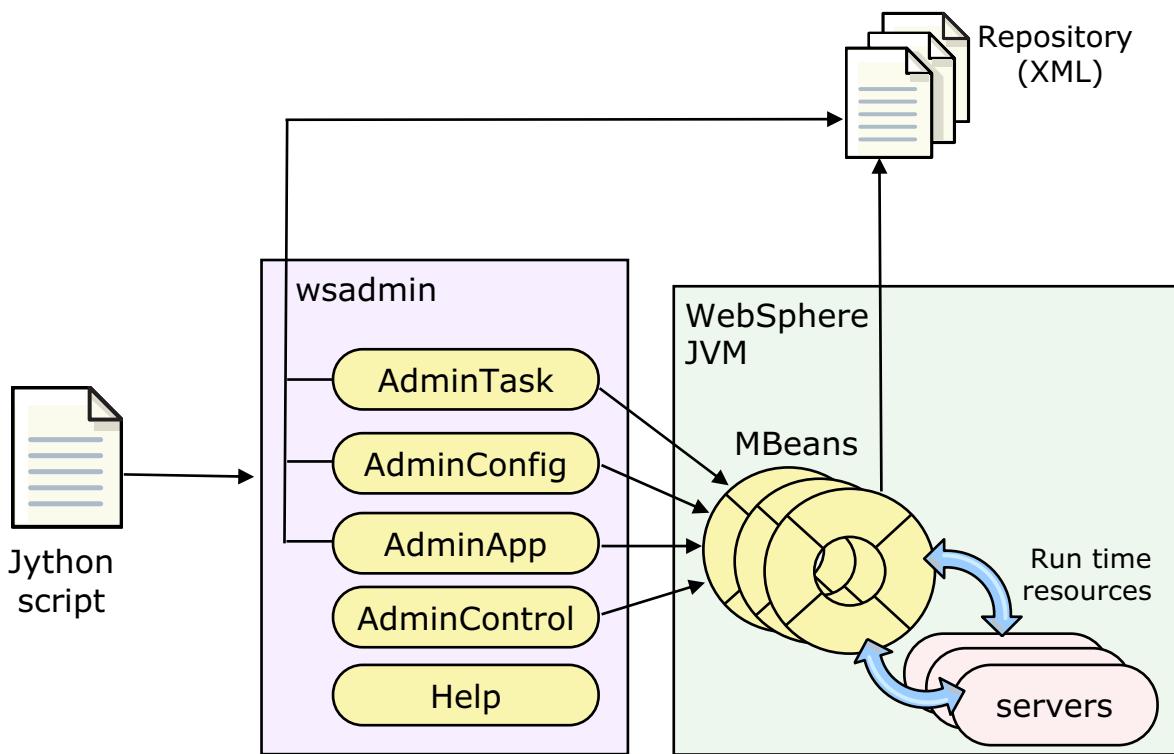
Figure 3-15. Script execution: wsadmin administrative objects (1 of 2)

WA680 / VA6801.0

Notes:

Each of these administrative objects is covered in greater detail in the later units in this course.

Script execution: wsadmin administrative objects (2 of 2)



© Copyright IBM Corporation 2008,2013

Figure 3-16. Script execution: wsadmin administrative objects (2 of 2)

WA680 / VA6801.0

Notes:

Each of these administrative objects is covered in greater detail in the later units in this course.

What is Ant?

- Apache Ant is a Java based build tool
- Provides a simple open source scripting engine that can run scripts that are written in XML format
- WebSphere Application Server provides a set of Ant-based tasks that you can use to perform common administrative operations
- Allows you to automate the task of compiling, packaging, installing, and testing applications on the application server

© Copyright IBM Corporation 2008,2013

Figure 3-17. What is Ant?

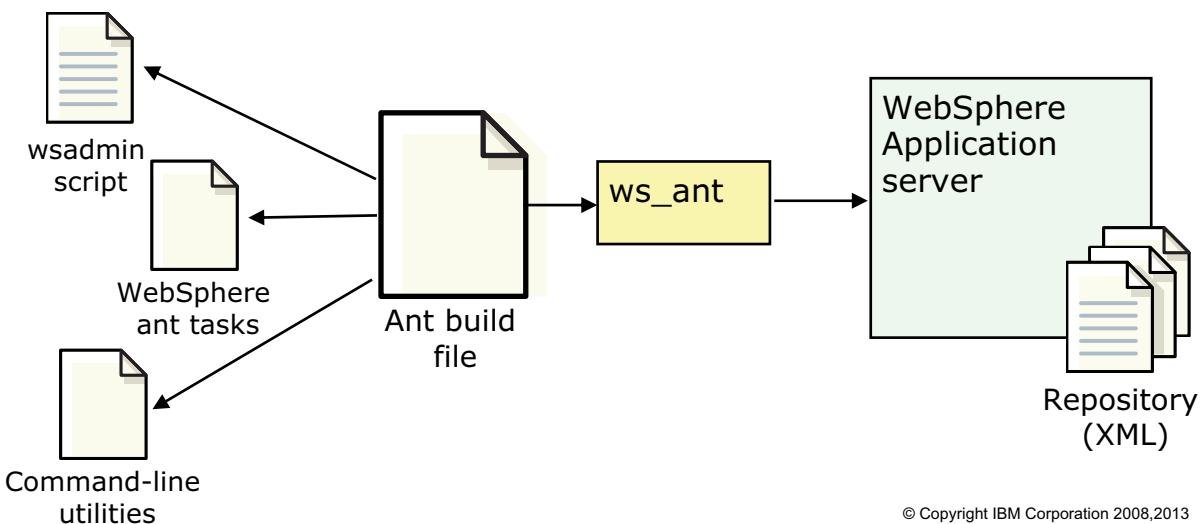
WA680 / VA6801.0

Notes:

In theory, Ant it is similar to Make, but Ant is different. Instead of a model that extends it is with shell-based commands. You can extend Ant by using Java classes. Instead of writing shell commands, XML-based configuration files are used. These files reference a target tree in which various tasks are run. There is an object that runs each task that implements a particular task interface.

Script execution: ws_ant

- A copy of the Apache Ant. The Java build tool
- WebSphere specific tasks include:
 - Install and uninstall applications
 - Start and stop servers
 - Run administrative scripts or commands



© Copyright IBM Corporation 2008,2013

Figure 3-18. Script execution: ws_ant

WA680 / VA6801.0

Notes:

Scripting languages: Jacl and Jython

- wsadmin supports both the Jacl and Jython scripting languages
- Jacl
 - Java implementation of Tcl
 - No new enhancements or features
 - Conversion assistant tool available to convert from Jacl to Jython (Jacl2Jython)
- Jython
 - Java implementation of Python
 - Syntax seems more natural to programmers used to Java or C
 - Supported by tools
 - AST (Jython editor, command completion, debugger)
 - Administrative console command assistance feature

© Copyright IBM Corporation 2008,2013

Figure 3-19. Scripting languages: Jacl and Jython

WA680 / VA6801.0

Notes:

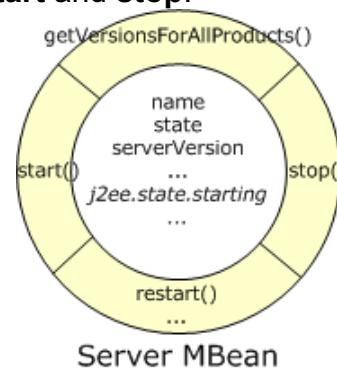
You should use the Jython language for writing new administrative scripts.

WebSphere MBeans

- JMX MBeans are used to expose managed resources to administrative clients.
 - The administrative console and wsadmin scripting client use the WebSphere MBeans to administer resources

- WebSphere provides a number of MBean types that represent the different resources that can be managed.
 - For example, the **Server** MBean type represents a server. It exposes a server's process control operations, such as **start** and **stop**.

- Each MBean can have:
 - Attributes
 - Operations
 - Notifications



© Copyright IBM Corporation 2008,2013

Figure 3-20. WebSphere MBeans

WA680 / VA6801.0

Notes:

MBeans represent the live objects that run in a WebSphere server process. The number and type of MBeans vary depending on the server process. If a scripting client is connected to a deployment manager, then all MBeans in all server processes are visible. If a scripting client is connected to a node agent, all MBeans in all server processes on that node are accessible. When connected to an application server, only MBeans running in that application server are visible. MBeans represent the underlying administrative architecture of the WebSphere Application Server.

Unit summary

Having completed this unit, you should be able to:

- Introduce the scripting tools available in WebSphere Application Server V8.5
- List the supported scripting languages
- Identify the Administrative objects
- Explain the WebSphere Application Server configuration model

© Copyright IBM Corporation 2008,2013

Figure 3-21. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint

1. True or false: Jython is the recommended language for wsadmin scripting with WebSphere Application Server V8.5.
2. Name three prerequisites that you should become familiar with before writing a wsadmin script.
3. When writing a wsadmin script to manage a run time component, you most likely must use:
 - a) A Stateful session bean
 - b) An MBean
 - c) An Ant task
 - d) The AdminRunTime object

© Copyright IBM Corporation 2008,2013

Figure 3-22. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

Checkpoint solutions

1. True or false: Jython is the recommended language for wsadmin scripting with WebSphere Application Server V8.5.
Answer: True
2. Name three prerequisites that you should become familiar with before writing a wsadmin script.
 1. wsadmin tool
 2. Administrative objects
 3. Jython syntax
 4. WebSphere Application Server configuration model
 5. WebSphere Application Server MBean types
3. When writing a wsadmin script to manage a run time component, you most likely must use:
 - a) A Stateful session bean
 - b) An MBean
 - c) An Ant task
 - d) The AdminRunTime object

Answer: An MBean

© Copyright IBM Corporation 2008,2013

Figure 3-23. Checkpoint answers

WA680 / VA6801.0

Notes:

Unit 4. Using wsadmin

What this unit is about

This unit describes the wsadmin scripting tool and explains the usage of the Help Administrative object.

What you should be able to do

After completing this unit, you should be able to:

- Describe the features and usage of wsadmin
- Explain the usage of the Help Administrative object

How you will check your progress

- Checkpoint questions
- Exercises

References

WebSphere Application Server V8.5 Information Center
<http://publib.boulder.ibm.com/infocenter/wasinfo/v8r5/index.jsp>

Unit objectives

After completing this unit, you should be able to:

- Describe the features and usage of wsadmin
- Explain the usage of the Help Administrative object

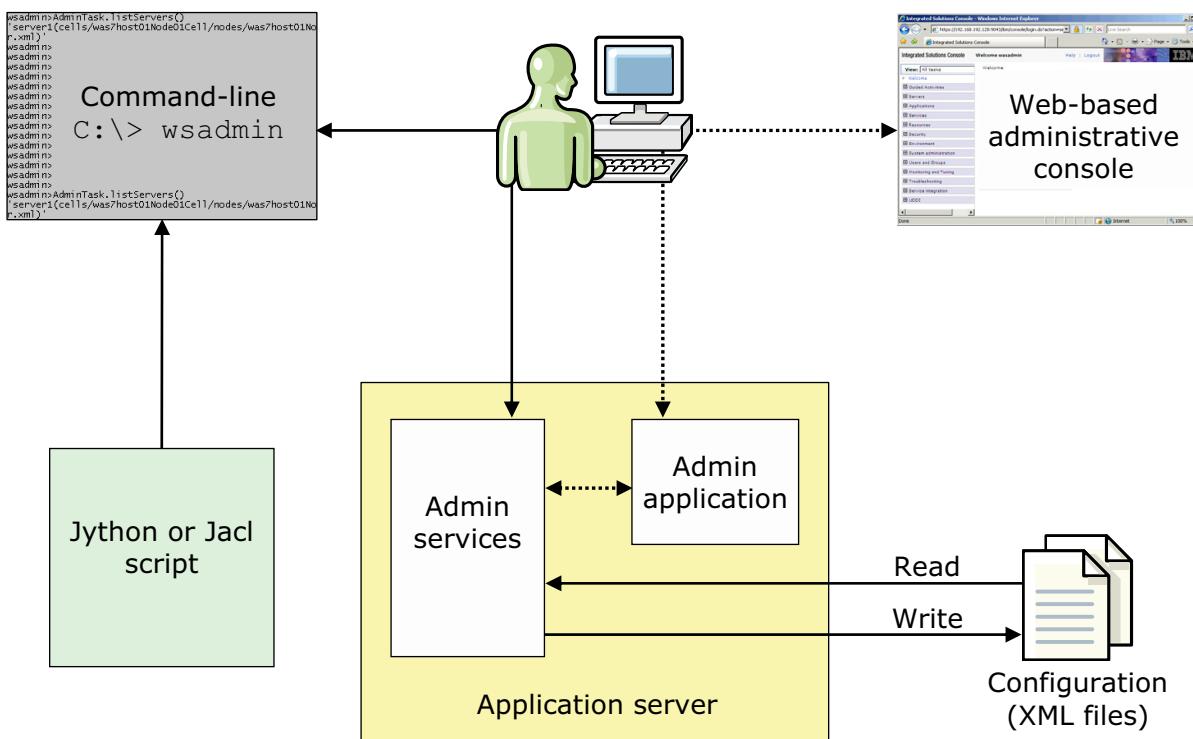
© Copyright IBM Corporation 2013

Figure 4-1. Unit objectives

WA680 / VA6801.0

Notes:

wsadmin versus administration console



© Copyright IBM Corporation 2013

Figure 4-2. wsadmin versus administration console

WA680 / VA6801.0

Notes:

Running commands from wsadmin is the same as doing the same tasks from the administrative console.

Scripting benefits

- Automation of routine administration tasks
- Schedule administration tasks
- Support changes in production environments, especially with multiple nodes
- Apply configuration changes or updates to targeted resources in an efficient and consistent manner
- Safer than the administrative console in production environments

© Copyright IBM Corporation 2013

Figure 4-3. Scripting benefits

WA680 / VA6801.0

Notes:



wsadmin

- Provides:
 - Scripting capabilities
 - Command-line administration
- Support for:
 - Python commands through Jython
 - Tcl commands through Java Command Language (Jacl)
- Modes:
 - Interactive
 - Command-line
 - Script file
- Examples:
 - Start and stop deployment manager, nodes, application servers, enterprise applications, and clusters
 - Configure virtual hosts, JDBC providers, JMS resources
 - Create application servers
 - Create clusters and add members to a cluster

© Copyright IBM Corporation 2013

Figure 4-4. wsadmin

WA680 / VA6801.0

Notes:

The WebSphere Application Server wsadmin tool runs scripts. You can use the wsadmin tool to manage a WebSphere Application Server V8.5. The wsadmin environment supports a two scripting languages to configure and control WebSphere Application Server. The wsadmin launcher makes Java objects available through language-specific interfaces. Scripts use these objects for application management, configuration, operational control, and for communication with MBeans running in WebSphere server processes. Examples of operations that can be done include: stop and start of the application server, cluster, or enterprise application. You can use wsadmin to create configurations such as virtual hosts, JDBC providers, application server, clusters, and cluster members to name a few.

wsadmin invocation options

- wsadmin invocation options include:

```
wsadmin
-h (help) or -?
-c <command>
-f <script_file_name>
-p <properties_file_name>
-profile <profile_script_name>
-profileName <profile_name>
-lang
-conntype [SOAP | RMI | JSR160RMI | IPC | NONE ]
-host
-user or username
-password
-port
```

© Copyright IBM Corporation 2013

Figure 4-5. wsadmin invocation options

WA680 / VA6801.0

Notes:

wsadmin starts a new Java virtual machine process when it is run. The parameter “-help” is used to get a brief description about wsadmin. Use the parameter “-lang” to specify either Jacl or Jython, the parameter “-conntype” for selecting the connection type. There are other parameters for host, port, user, and password.

Jython versus Jacl

- Jython syntax is more natural to Java or C programmers
- Jython provides much better support in tools
 - Rational Application Developer (Jython editor, command completion, debugger)
 - Administrative console (console command assistance)
- Jacl syntax is more familiar to administrators familiar with Tcl
- Jacl is supported in version 8
 - No future development or enhancements for Jacl
 - Jacl-to-Jython (Jacl2Jython) conversion assist tool is available

© Copyright IBM Corporation 2013

Figure 4-6. Jython versus Jacl

WA680 / VA6801.0

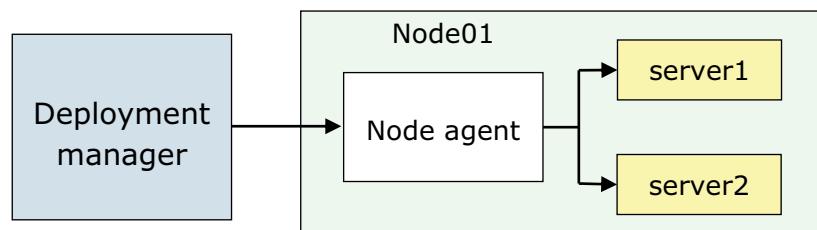
Notes:

Administrative functions that use wsadmin

- WebSphere Application Server system management separates administrative functions into two categories:

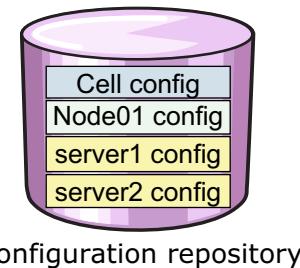
1

Running objects in WebSphere Application Server installations



2

Configuration of WebSphere Application Server installations (repository)



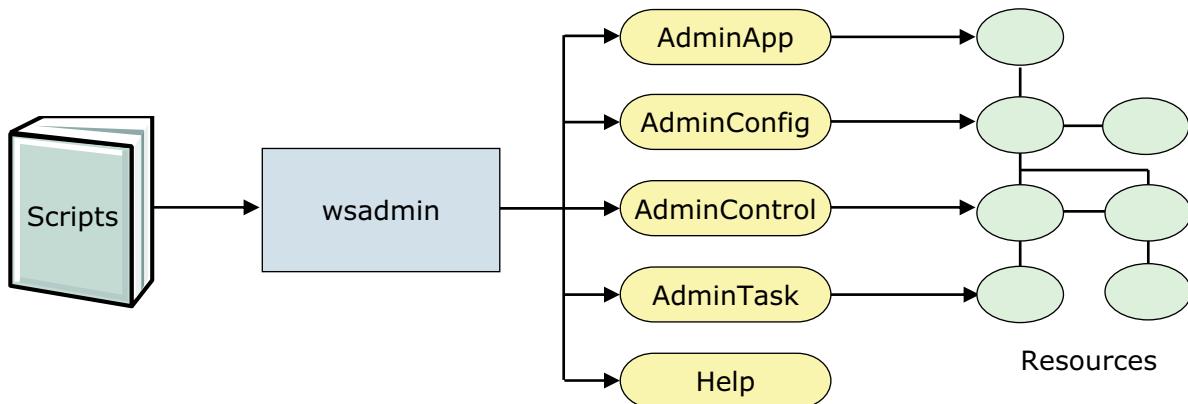
© Copyright IBM Corporation 2013

Figure 4-7. Administrative functions that use wsadmin

WA680 / VA6801.0

Notes:

Administrative objects in wsadmin



- Five administrative objects perform different operations:
 - **AdminControl**: work with “live” running objects
 - **AdminConfig**: create or modify WebSphere Application Server “static” configuration
 - **AdminApp**: install, modify, or administer applications
 - **AdminTask**: administrative commands that are more usable and task-oriented
 - **Help**: obtain general help

© Copyright IBM Corporation 2013

Figure 4-8. Administrative objects in wsadmin

WA680 / VA6801.0

Notes:



Starting wsadmin

- Interactively

wsadmin

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin
WASX7209I: Connected to process "server1" on node was7host01Node01 using SOAP connector; The type of process is: UnManagedProcess
WASX7029I: For help, enter: "$Help help"
wsadmin>
```

- Command option

wsadmin -c <command>

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -c print(AdminApp.list()) -lang jython
WASX7209I: Connected to process "server1" on node was7host01Node01 using SOAP connector; The type of process is: UnManagedProcess
DefaultApplication
ivtApp
query
```

- Script file

wsadmin -f <script_file>

```
C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -f "C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin\initialSIBSetup.py"
```

© Copyright IBM Corporation 2013

Figure 4-9. Starting wsadmin

WA680 / VA6801.0

Notes:

There are three ways to run wsadmin.

1. Interactive mode
2. Run a command or commands that are passed as arguments
3. Run a script.

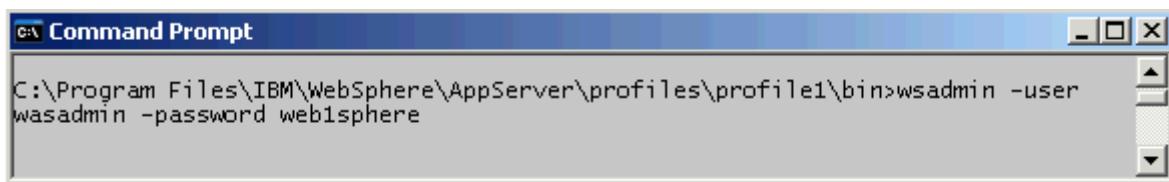


Starting wsadmin with security enabled

- If security is enabled, authentication information must be supplied
- There are several ways to provide authentication information:
 - Prompted



- Command-line parameters



- RMI connections: `sas.client.props` file
- SOAP connections: `soap.client.props` file

© Copyright IBM Corporation 2013

Figure 4-10. Starting wsadmin with security enabled

WA680 / VA6801.0

Notes:

When WebSphere has security that is enabled for administration, a user and password must be supplied. Several mechanisms can be used for the user and password. User and password can be passed as arguments when wsadmin is run. There are two property files that can contain the user and password. A dialog is displayed if no user and password are specified.

wsadmin properties

- Certain default behaviors for wsadmin can be changed by editing:
`<profile_root>\<profile_name>\properties\wsadmin.properties`
- Examples of 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=jython  
- 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=
```

© Copyright IBM Corporation 2013

Figure 4-11. wsadmin properties

WA680 / VA6801.0

Notes:

Profile scripts

- Profile scripts can be used to load wsadmin with predefined settings and functions
- Run during wsadmin startup
- Either of the following can call a profile script:
 - Using the `-profile` option on the command line
 - Defined in `wsadmin.properties com.ibm.ws.scripting.profiles=`

```

#-----
# Print whereAMI
#-----
def whereAMI():
    #Print 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"
print "here. It would also be possible to extend wsadmin by defining new"
print "customized commands and procedures."
print ""

whereAMI()
print ""

```

© Copyright IBM Corporation 2013

Figure 4-12. Profile scripts

WA680 / VA6801.0

Notes:

In this example, a Jython script is passed as the profile property argument to wsadmin. When wsadmin starts, it runs the script before starting the interactive mode. Profile scripts can be used to preload wsadmin with predefined settings and functions. These profiles can be used to create a standard wsadmin environment for everyone.

AdminConfig: Managing configurations

- Management configuration scripts use the AdminConfig object to access the repository where configuration information is stored
- Example:

```
wsadmin> AdminConfig.list ('Server')
'server(cells/was8host01Node01Cell/nodes/was8host01Node01/servers/server1
|server.xml#Server_1183122130078
wsadmin>
```

- Use the AdminConfig object to:
 - List configuration objects and their attributes
 - Create configuration objects
 - Modify configuration objects
 - Remove configuration objects
 - Obtain help

© Copyright IBM Corporation 2013

Figure 4-13. AdminConfig: Managing configurations

WA680 / VA6801.0

Notes:

AdminApp: Managing applications

- Application management scripts use the AdminApp object to manage applications in the application server configuration
- Example:

```
wsadmin>print AdminApp.view('ivtApp')

Specifying application options

Specify the various options that are available to prepare and install your application.

Directory to install application: ${APP_INSTALL_ROOT}/${CELL}
Distribute application: Yes
Use Binary Configuration: No
Create MBeans for resources: No
Override class reloading settings for Web and EJB modules: No
Reload interval in seconds:
Validate Input off/warn/fail: off
File Permission: .*\*.dll=755#.*\*.so=755#.*\*.a=755#.*\*.sl=755
```

- Use the AdminApp object to:
 - Install and uninstall applications
 - List installed applications
 - Edit application configurations
 - Obtain help

© Copyright IBM Corporation 2013

Figure 4-14. AdminApp: Managing applications

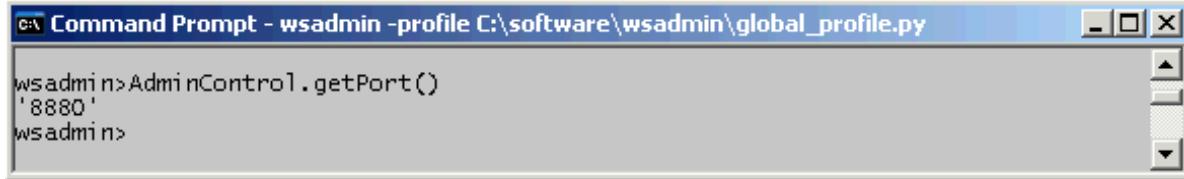
WA680 / VA6801.0

Notes:



AdminControl: Managing running objects

- Operation management scripts use the AdminControl object to communicate with the MBeans that represent running objects
- Example:



The screenshot shows a Windows Command Prompt window titled "Command Prompt - wsadmin -profile C:\software\wsadmin\global_profile.py". The window contains the following text:
wsadmin> AdminControl.getPort()
'8880'
wsadmin>

- Use the AdminControl object to:
 - List running objects and their attributes
 - Start actions on running objects
 - Obtain dynamic information about MBeans that represent running objects
 - Obtain help

© Copyright IBM Corporation 2013

Figure 4-15. AdminControl: Managing running objects

WA680 / VA6801.0

Notes:

AdminTask: Accessing administrative functions

- AdminTask object is used to access a set of administrative commands to provide an alternative way to access configuration commands
- Example:

```
wsadmin> AdminTask.binaryAuditLogReader('-interactive')
Binary Audit Log Reader

Binary Audit Log Reader Command

*File name of the Binary Audit log (filename): C:\Program
 Files\IBM\WebSphere\AppServer\profiles\dmgrProfile\logs\dmgr\
BinaryAudit_was8host01Cell01_was8host01CellManager01_dmgr.log
```

- Benefits of using AdminTask:
 - Provides more usable and task-oriented commands
 - Runs simple and complex commands
 - Commands grouped based on function
 - Can be run in batch or interactive mode
 - Can be run in connected or local mode

© Copyright IBM Corporation 2013

Figure 4-16. AdminTask: Accessing administrative functions

WA680 / VA6801.0

Notes:

Help within wsadmin

Jython

- print Help.help()
- print Help.AdminConfig()
- print Help.AdminTask()
- print Help.AdminControl()
- print Help.AdminApp()

```

wsadmin>print 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.

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
operations     given an MBean, returns help for operations
constructors   given an MBean, returns help for constructors
description    given an MBean, returns help for description
notifications  given an MBean, returns help for notifications

```

© Copyright IBM Corporation 2013

Figure 4-17. Help within wsadmin

WA680 / VA6801.0

Notes:

You can obtain help on any of the administrative tasks. Use the help commands as shown. The help command provides you with an overview of each of the administrative objects and the required parameters. Put the print statement in front of the command to have the output that is nicely formatted.

Administrative command help

- You can select from three levels of online help for administrative commands
- Example of command-level help:



The screenshot shows a Windows Command Prompt window titled "Command Prompt - wsadmin -lang jython". The user has run the command "wsadmin>print AdminTask.help('listServers')". The output provides detailed help for the "listServers" command, including its description, target object (None), and arguments (serverType and nodeName).

```
wsadmin>print AdminTask.help('listServers')
WASX8006I: Detailed help for command: listServers

Description: list servers of specified server type and node name. If node name is
not specified, whole cell will be searched. If the server type is not specified
servers of all types are returned.

Target object: None

Arguments:
  serverType - The ServerType ie: (APPLICATION_SERVER)
  nodeName - The Node Name
```

© Copyright IBM Corporation 2013

Figure 4-18. Administrative command help

WA680 / VA6801.0

Notes:

Important points to remember when using wsadmin

- Commands are case-sensitive
- Running multiple commands in a script file is faster than running individual commands
- Saving configuration changes is a two-step process:
 - The first step validates the changes
 - The second step completes the save
- Save periodically:

```
wsadmin> AdminConfig.save()  
''  
wsadmin>
```

© Copyright IBM Corporation 2013

Figure 4-19. Important points to remember when using wsadmin

WA680 / VA6801.0

Notes:

You must use AdminConfig.save() to save your work. updates to the configuration through a scripting client are kept in a private temporary area that is called a workspace and are not copied to the master configuration repository until you run a save command. The workspace is a temporary repository of configuration information that administrative clients use. The workspace is kept in the wstemp subdirectory of your WebSphere Application Server installation. The use of the workspace allows multiple clients to access the master configuration. If the more than one client makes an update, it is possible that these updates can cause a conflict.

Scripting: Simple script

```

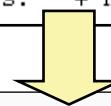
print "Simple wsadmin scripting example."

cell = AdminConfig.list("Cell")
node = AdminConfig.list("Node")

cellName = AdminConfig.showAttribute(cell, "name")
nodeName = AdminConfig.showAttribute(node, "name")

print ""
print "Cell name is: " + cellName
print "Node name is: " + nodeName

```



```

C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -f
"C:\software\wsadmin\simple_script.py" -username wasadmin -password
*****
WASX7209I: Connected to process "server1" on node was8host01Node01 using
SOAP connector: The type of process is: UnManagedProcess
Simple wsadmin scripting example

Cell name is: was8host01Node01Cell
Node name is: was8host01Node01

```

© Copyright IBM Corporation 2013

Figure 4-20. Scripting: Simple script

WA680 / VA6801.0

Notes:

Scripting: Looping script

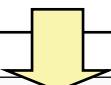
```

print "Simple loop scripting example"

appNames = AdminApp.list()
AppNamesArray = appNames.split('\r\n')

for appName in AppNamesArray:
    print "app Name: " + appName

```



```

C:\Program Files\IBM\WebSphere\AppServer\profiles\profile1\bin>wsadmin -f
"C:\software\wsadmin\simple_script.py" -username wasadmin -password
*****
WASX7209I: Connected to process "server1" on node was8host01Node01 using
  SOAP connector: The type of process is: UnManagedProcess
Simple loop scripting example

App Name: DefaultApplication
App Name: PlantsByWebSphere
App Name: ivtApp
App Name: query

```

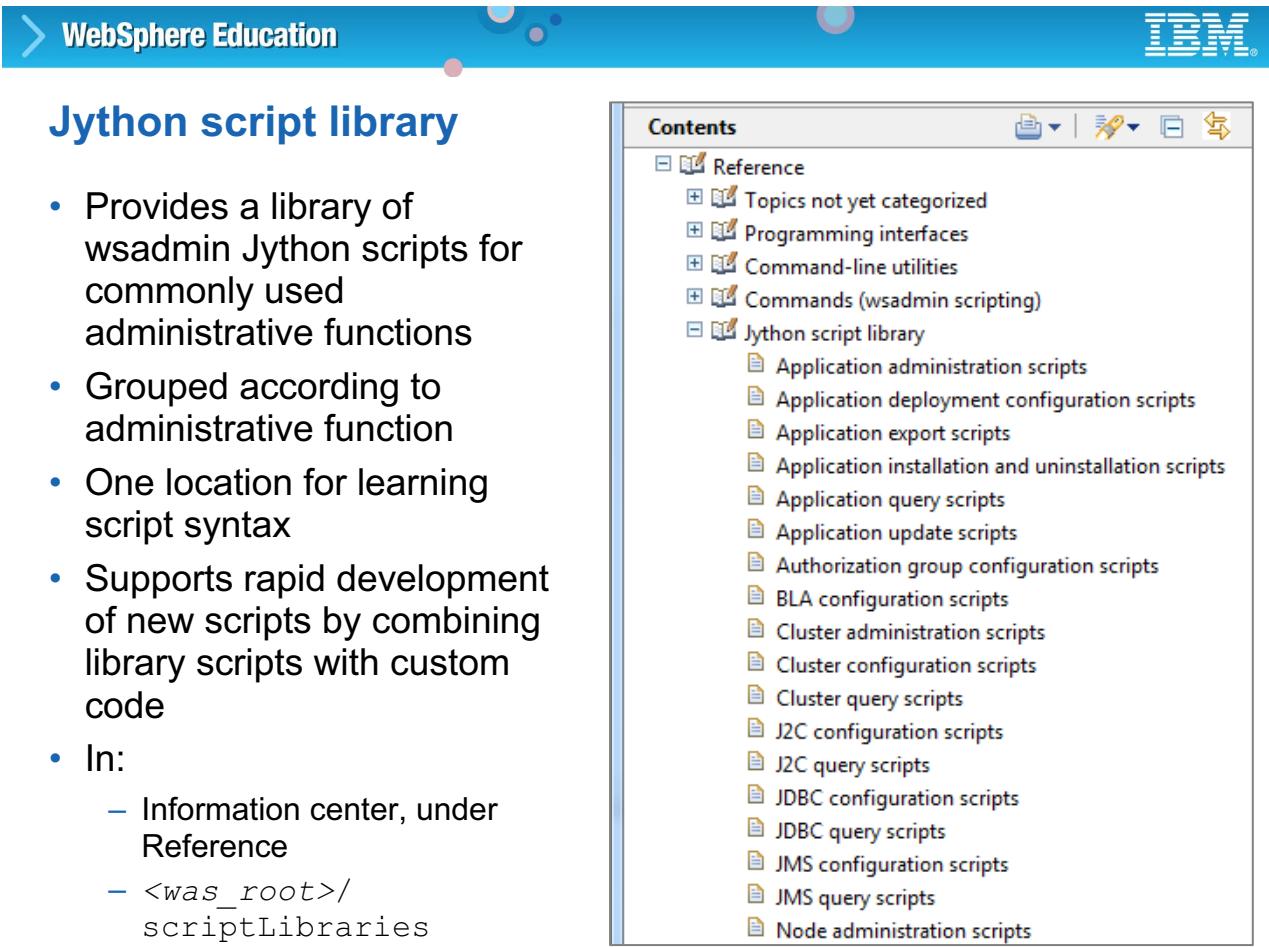
© Copyright IBM Corporation 2013

Figure 4-21. Scripting: Looping script

WA680 / VA6801.0

Notes:

Here is an example that shows a unique feature of the Jython language. White space indicates a block of code. The loop ends because there are no more indented lines.



The screenshot shows the WebSphere Education interface with the title 'WebSphere Education' and the IBM logo. The main content area is titled 'Jython script library'. To the right is a 'Contents' sidebar with a tree view of available scripts. The tree structure includes 'Reference' (Topics not yet categorized, Programming interfaces, Command-line utilities, Commands (wsadmin scripting)), 'Jython script library' (Application administration scripts, Application deployment configuration scripts, Application export scripts, Application installation and uninstallation scripts, Application query scripts, Application update scripts, Authorization group configuration scripts, BLA configuration scripts, Cluster administration scripts, Cluster configuration scripts, Cluster query scripts, J2C configuration scripts, J2C query scripts, JDBC configuration scripts, JDBC query scripts, JMS configuration scripts, JMS query scripts, Node administration scripts), and a toolbar with icons for search, refresh, and navigation.

- Provides a library of wsadmin Jython scripts for commonly used administrative functions
- Grouped according to administrative function
- One location for learning script syntax
- Supports rapid development of new scripts by combining library scripts with custom code
- In:
 - Information center, under Reference
 - <was_root>/scriptLibraries

Figure 4-22. Jython script library

WA680 / VA6801.0

Notes:

How to use the Jython script library

- There are three ways to use the Jython script library:

- Run scripts in interactive mode with the wsadmin tool

```
wsadmin>AdminServerManagement.createApplicationServer("profile1",
    "server1", "default")
```

- Use a text editor to combine several scripts

```
# My Custom Jython Script - filepy
AdminServerManagement.createApplicationServer("profile1",
    "server1", "default")
AdminServerManagement.createApplicationServer("profile2",
    "server2", "default")

# Use one of them as the first member of a cluster
AdminClusterManagement.createClusterWithFirstMember("cluster1",
    "APPLICATION_SERVER", "profile1", "server1")

# Install an application
AdminApplication.installAppWithClusterOption("DefaultApplication",
    "..\installableApps\DefaultApplication.ear", "cluster1")

# Start all servers and applications on the node
AdminServerManagement.startAllServers("profile1")
```

- Use the Jython scripting library code as sample syntax to write custom scripts

© Copyright IBM Corporation 2013

Figure 4-23. How to use the Jython script library

WA680 / VA6801.0

Notes:

Configuration repository: The issues

- The repository consists of multiple files in XML and other formats
- The configuration files are spread across many directories
- Configuration objects are complex
- Some configuration objects repeatedly stored in multiple files
- Example: properties for a JDBC provider

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1183122153343" name="Derby JDBC Provider">
  <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</classpath>
  <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1183122153625" name="D
    <propertySet xmi:id="J2EEResourcePropertySet_1183122153625">
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153625" name="databaseName" ty
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153626" name="shutdownDatabase"
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153627" name="dataSourceName"
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153628" name="description" typ
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153629" name="connectionAttribut
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153630" name="createDatabase"
    </propertySet>
    <connectionPool xmi:id="ConnectionPool_1183122153631">
  </factories>
</resources.jdbc:JDBCProvider>
```

© Copyright IBM Corporation 2013

Figure 4-24. Configuration repository: The issues

WA680 / VA6801.0

Notes:

Properties file based configuration: A solution

- Properties files are more human readable
 - Properties files consist of name and value pairs
 - Decouples configuration data from changes in the underlying configuration model between releases
 - Can be used with configuration archives
 - Differences between configuration environments are easier to identify

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName  
    jdbcprovider.props -configData Server=server1 filterMechanism  
• SELECTED_SUBTYPES -selectedSubTypes [JDBCProvider]')
```

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_118
  <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</cla
  <factories xmi:type="resources.jdbc:DataSource" xmi
    <propertySet xmi:id="J2EEResourcePropertySet_1183
      <resourceProperties xmi:id="J2EEResourcePropert
      <resourceProperties xmi:id="J2EEResourcePropert
    </propertySet>
    <connectionPool xmi:id="ConnectionPool_1183122153
  </factories>
</resources.jdbc:JDBCProvider>
```

ResourceType=JDBCProvider
ImplementingResourceType=JDBCProvider
ResourceId=Cell!=!(cellName):Node!=!(nodeName):Server#

#Properties

classpath=\${DERBY_JDBC_DRIVER_PATH}/derby.jar
name=Derby JDBC Provider (XA)
implementationClassName=org.apache.derby.jdbc.EmbeddedDriver
nativepath={}
description=Built-in Derby JDBC Provider (XA)
providerType=Derby JDBC Provider (XA) #readonly
xa=true #boolean

© Copyright IBM Corporation 2013

Figure 4-25. Properties file based configuration: A solution

WA680 / VA6801.0

Notes:

Properties file configuration content

- Each object is defined in a separate section:
 - Resource type and identifier
 - Configuration information
- Example: properties for a JDBC provider

```
# SubSection 1.0 # JDBCProvider attributes
#
# ResourceType=JDBCProvider
# ImplementingResourceType=JDBCProvider
# ResourceId=Cell!=!{cellName}:Node!=!{nodeName}:Server!=!
#   {serverName}:JDBCProvider=ID#JDBCProvider_1183122153343
#
# Properties
#
#   classpath={$DERBY_JDBC_DRIVER_PATH}/derby.jar}
# name=Derby JDBC Provider
# implementationClassName=
#   org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource
# nativepath={}
# description=Derby embedded non-XA JDBC Provider
# providerType=Derby JDBC Provider #readonly
# xa=false #boolean
```

1

Resource type and identifier

2

Configuration information

© Copyright IBM Corporation 2013

Figure 4-26. Properties file configuration content

WA680 / VA6801.0

Notes:

Properties file configuration commands

- extractConfigProperties: extracts configuration data into a properties file

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName  
server1.props -configData Server=server1')
```

- validateConfigProperties: verifies that the properties in the properties file are valid

```
wsadmin>AdminTask.validateConfigProperties('-propertiesFileName  
server1.props')
```

- applyConfigProperties: applies properties in a specific properties file

```
wsadmin>AdminTask.applyConfigProperties('-propertiesFileName  
app.props')
```

- deleteConfigProperties: deletes objects in your configuration

```
wsadmin>AdminTask.deleteConfigProperties('-propertiesFileName  
thread.props')
```

- createPropertiesFileTemplates: creates template properties files

```
wsadmin>AdminTask.createPropertiesFileTemplates('-propertiesFileName  
app.props -configType Application')
```

© Copyright IBM Corporation 2013

Figure 4-27. Properties file configuration commands

WA680 / VA6801.0

Notes:



Create, test, and debug Jython scripts

Rational Application Developer Assembly and Deploy provides:

- Full support for Jython projects and source files
- Ability to create, test, and debug Jython projects and files
- Syntax colored editor and command completion tools
- A source outline view
- Source level debugging

```

def changeEnvValue(envVarName, envVarValue):
    print("Setting variable " + varName + " on node " + nodeName)
    nodeId = AdminConfig.getid("/Node:" + nodeName + "/")
    varSubstitutions = AdminConfig.list("VariableSubstitution")
    for varSubst in varSubstitutions:
        curVarName = AdminConfig.showAttribute(varSubst, "name")
        print AdminConfig.showAttribute(varSubst, "value")
        AdminConfig.modify(varSubst, [{"name": "value", "value": newVarValue}])
        break
#
# TODO: enter JYTHON code and save
#
nodeName = AdminControl.getNode()
varName = "DB2UNIVERSAL_JDBC_DRIVER_PATH"
newValue = "C:\Program Files\IBM\SQLLIB\java"
lf = java.lang.System.getProperty("line.separator")

changeEnvValue(varName, newValue)

print ("Saving configuration")
AdminConfig.save()

```

Figure 4-28. Create, test, and debug Jython scripts

WA680 / VA6801.0

Notes:

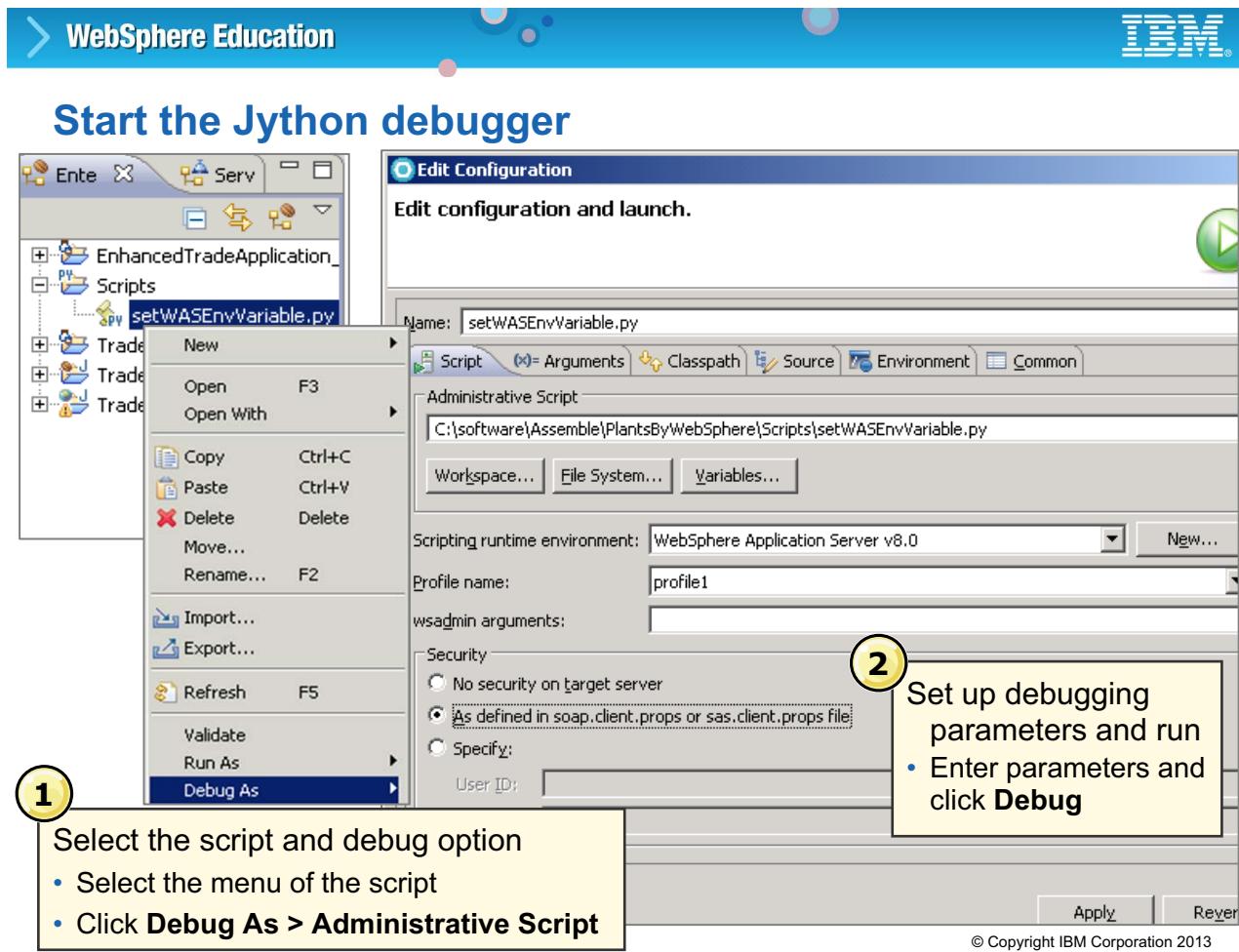


Figure 4-29. Start the Jython debugger

WA680 / VA6801.0

Notes:



Jython debugger perspective

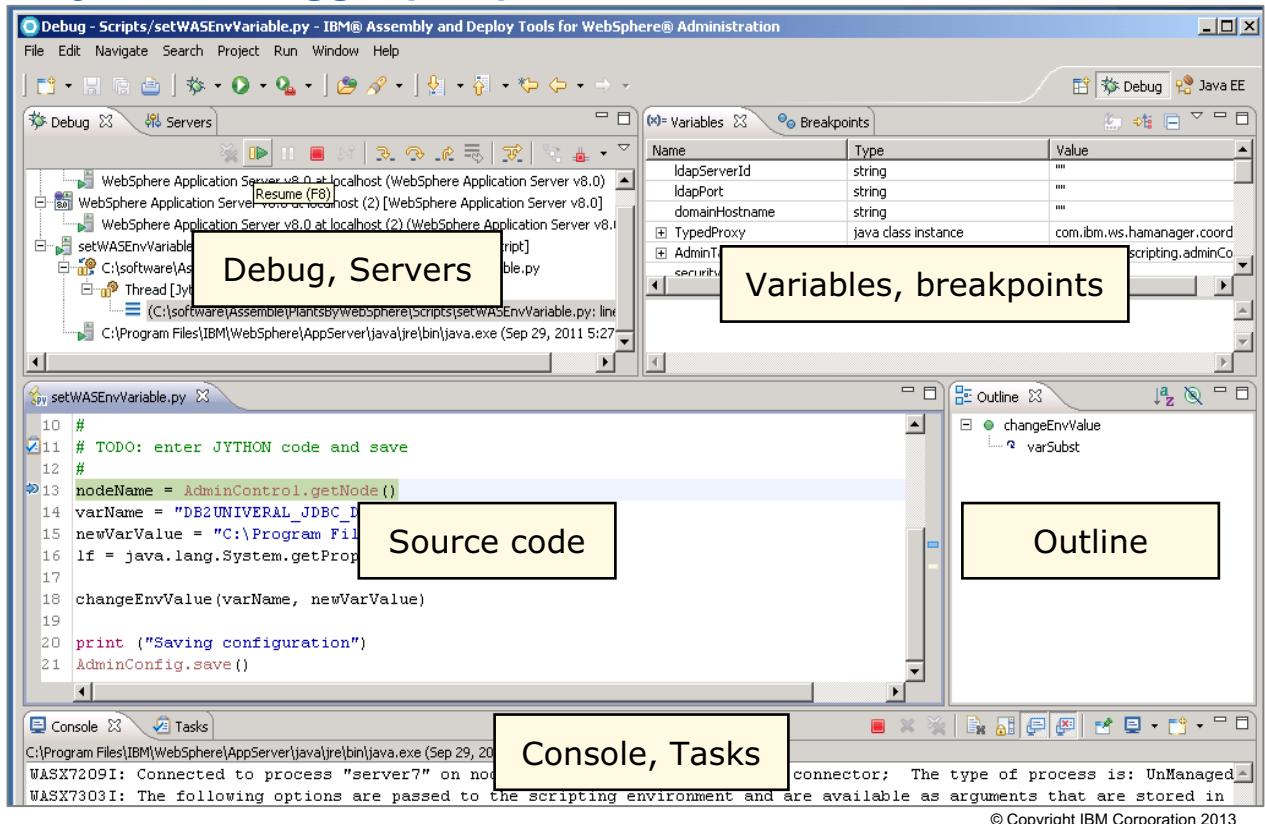


Figure 4-30. Jython debugger perspective

WA680 / VA6801.0

Notes:

The screenshot shows the "Administrative Scripting Commands" interface. At the top, there's a navigation bar with a blue arrow pointing right labeled "WebSphere Education" and the IBM logo. Below the navigation bar, the title "Command assistance" is displayed in a large blue font. To the left, a sidebar lists "Administrative Scripting Commands" and "Preferences". The main content area displays a table for the "AdminApp.list()" command. The table includes columns for "Method", "Arguments", and "Description". The "Method" column shows "WASX7100I: Method: list", "Arguments" shows "none", and "Description" shows "Lists all installed applications". A yellow box highlights the "Method" and "Arguments" rows. Below this, another row for "AdminApp.list(target_scope)" is shown with "Method: list", "Arguments: target scope", and "Description: List installed applications on a given target scope". To the right, a "Help" panel is open, showing sections for "Field help", "Page help", and "Command Assistance". The "Command Assistance" section contains a link "View administrative scripting command for last action" which is highlighted with a red box. At the bottom right of the help panel, it says "© Copyright IBM Corporation 2013".

- Works in concert with the administrative console
 - Last run commands are made available to Rational Application Developer
 - Commands can be pasted directly to Jython scripts

- Administrative console access
 - Under **Help**, click **View administrative scripting command for last action**
 - The last command run is displayed
 - Place the cursor over the command to get command information
 - Command can be copied into a Jython script

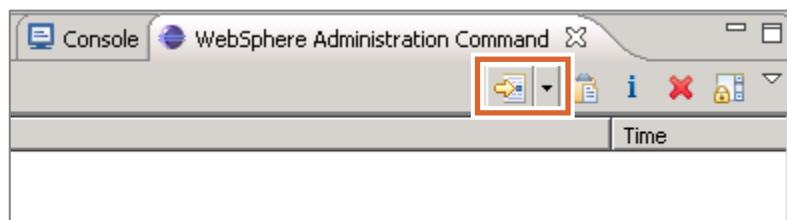
Figure 4-31. Command assistance

WA680 / VA6801.0

Notes:

Using command assistance within IBM Assembly and DeployTools

- Command assistance setup includes the following steps:
 - Add the WebSphere Administration Command view:
Click **Window > Show View > Other > Server > WebSphere Administration Command**
 - Open the WebSphere Administration Command view
 - Use **Select Server to Monitor** to connect to the server



- Select the command that you want
- Use **Insert into Editor** to copy the command to a Jython script file

© Copyright IBM Corporation 2013

Figure 4-32. Using command assistance within IBM Assembly and DeployTools

WA680 / VA6801.0

Notes:



Unit summary

Having completed this unit, you should be able to:

- Describe the features and usage of wsadmin
- Explain the usage of the Help Administrative object

© Copyright IBM Corporation 2013

Figure 4-33. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. Which of the following is not one of the five Java objects that perform different operations?
 - A. AdminConfig
 - B. AdminControl
 - C. AdminTask
 - D. Help
 - E. AdminStart
2. What is the default protocol type for wsadmin?
 - A. SOAP
 - B. RMI
 - C. None
3. The default behaviors for wsadmin can be changed by editing which file?

© Copyright IBM Corporation 2013

Figure 4-34. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.



Checkpoint answers

1. Which of the following is not one of the five Java objects that perform different operations?

E. AdminStart

2. What is the default protocol type for wsadmin?

A. SOAP

3. The default behaviors for wsadmin can be changed by editing which file?

wsadmin.properties

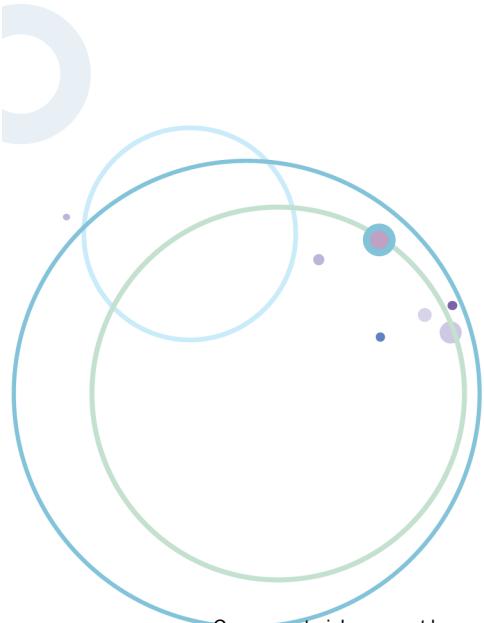
© Copyright IBM Corporation 2013

Figure 4-35. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 1



Using wsadmin to explore the
WebSphere Application Server
environment

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 4-36. Exercise 1

WA680 / VA6801.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Start wsadmin in its three supported modes
- Run simple commands that use the Jython syntax
- Run simple commands that use the Administrative objects to query the WebSphere Application Server environment

© Copyright IBM Corporation 2013

Figure 4-37. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 5. Jython basics

What this unit is about

This unit provides a primer of the Jython programming language.

What you should be able to do

After completing this unit, you should be able to:

- Describe the basic elements of Jython

Unit objectives

After completing this unit, you should be able to:

- Describe the basic elements of Jython

© Copyright IBM Corporation 2013

Figure 5-1. Unit objectives

WA680 / VA6801.0

Notes:

A Jython program (also called a script) is a simple text file that contains statements that are interpreted as they are input. It contains comments, variables, statements, and code blocks. Comments are shown at the start of a line or following white space or code, but not within a string literal. A hash character (#) within a string literal is just a hash character. You must assign a value to a variable before referencing it. Names starting with underscore are generally reserved for internal variables. You can use the statement separator character ';' to put multiple statements on the same physical line, when a compound statement is entered interactively. A compound statement is followed by a blank line to indicate completion. The parser cannot guess which is the last line. Python uses indentation to group statements. Each line that belongs to the same code block must have the same indentation.

What is Jython?

- Jython is an object-oriented, open source programming language
 - Based the Python language specification
 - Implemented in Java
 - wsadmin uses Jython V2.1
- Seamless integration with Java
 - Directly access Java classes
- Dynamic execution
 - Jython code is interpreted.
- Interactive interpreter
 - Command-line interpreter to interactively run Jython code

© Copyright IBM Corporation 2013

Figure 5-2. What is Jython?

WA680 / VA6801.0

Notes:



Jython Features

- Dynamic typing
 - No requirement to declare variables
- High-level data types
 - Built-in and easy to use
 - Example: sequence types and dictionaries
- Rich standard library
 - Includes the Python standard library
- Module-based code packaging
 - Jython code is packaged into modules for importing purposes.

© Copyright IBM Corporation 2013

Figure 5-3. Jython Features

WA680 / VA6801.0

Notes:

Jython benefits

- Interpreted language
 - Reduces program development time
 - Can be used interactively
 - Ideal for scripting
- Compact and readable
 - High-level data types allow you to perform complex operations using a small number of statements.
 - Statement grouping is done by indentation; no special block delimiter characters are needed.
 - No variable or argument declarations are necessary.
- Java security
 - Jython inherits the Java security framework.
- Extensible
 - Can add a new built-in function or module to the interpreter
- Reusable
 - Modules can be reused in other programs.
 - Jython comes with a large collection of readily usable standard modules.

© Copyright IBM Corporation 2013

Figure 5-4. Jython benefits

WA680 / VA6801.0

Notes:

Basic program elements

- Comments
 - Start with the hash character, "#", and continue until the end of the line.
- Variables
 - Created when you assign a value to them
 - Name can start with an underscore ('_') or a letter, followed by one or more letters, digits or underscores.
 - Names are case-sensitive.
- Statements
 - Simple statements
 - Single clause that is expressed on one logical line (for example, the assignment statement)
 - Compound statements
 - Contain one or more blocks of code (for example, control flow statements)
- Code blocks
 - Group of statements that are associated with a compound statement
 - Indentation marks the beginning and end of the block

© Copyright IBM Corporation 2013

Figure 5-5. Basic program elements

WA680 / VA6801.0

Notes:

Example python program

```
# Say hello!           ← Comment line
s = "Hello world!"   ← Variable assignment
if 1:                ← if statement
    print "*****"
    print "*", s, "*"
    print "*****"
                                ← Indented code block
                                ← Blank line indicates end of block
                                (required in interactive mode only)

*****  

* Hello world! *      ← Execution result  

*****
```

© Copyright IBM Corporation 2013

Figure 5-6. Example python program

WA680 / VA6801.0

Notes:

Backslash is a special character

- Backslash \ is a statement continuation character.
 - Allows you to continue the current statement on the next line
- Backslash within a string can be used to represent special characters.
 - \n is the newline character
 - \r is the return character
 - \\" is the backslash
 - \b is the character for the bell

© Copyright IBM Corporation 2013

Figure 5-7. Backslash is a special character

WA680 / VA6801.0

Notes:

Jython allows a statement to continue beyond the end of a line if it contains an open parenthesis, bracket, brace, or triple-quote string. Such a statement can continue over any number of lines until the delimiter is closed. Any other statement can be continued to the next line by ending it with a backslash ("\\"). Backslash is used for special characters as shown.

Three types of reserved words

- Statement introducers:

assert, break, class, continue, def, del,
elif, else, except, exec, finally, for,
from, global, if, import, pass, print,
raise, return, try, while

- Parameter introducers:

as, import, in

- Operators:

and, in, is, lambda, not, or

© Copyright IBM Corporation 2013

Figure 5-8. Three types of reserved words

WA680 / VA6801.0

Notes:

Jython maintains a list of reserved key words. These reserved words fall under certain categories: statement introducers, parameter introducers, and operators. Examples of statement introducers are assert, break, class, if, else. The key words as, import and in, are considered parameter introducers, and in, is, not, or are considered operators. Python also has a special reserved word NONE. The None object is a single instance of the PyNone class and it is used to represent the absence of a value. This object is equivalent to the Java null value, and always maps to false. In interactive mode, the special variable '_' has the value of the last evaluated expression. It should be treated as read-only.



The None object

- Single instance of the PyNone class
- Used to represent the absence of a value
- Equivalent to the Java null value
- Always maps to false

© Copyright IBM Corporation 2013

Figure 5-9. The None object

WA680 / VA6801.0

Notes:

Basic built-in data type categories (1 of 2)

- Numbers
 - Integers
 - Floating-point numbers
 - Complex numbers
- Sequences
 - Ordered collections of objects
 - Elements that are accessed with an index
 - Can be mutable and immutable
 - Consist of strings, lists, and tuples

© Copyright IBM Corporation 2013

Figure 5-10. Basic built-in data type categories (1 of 2)

WA680 / VA6801.0

Notes:

Jython provides built in data types that are broken down into four main categories numbers, sequences, mappings, and files. The numbers category includes the Integer, Floating point, and complex number data types. The sequence category contains the data structures list and tuples, which are mutable and immutable. The mappings category provides the dictionary data structure that can be used to store collections by using key value pairs.



Basic built-in data type categories (2 of 2)

- Mappings
 - Keyed collections of objects (key/value pairs)
 - Elements are randomly accessed with a key
- Files
 - Operating system files

© Copyright IBM Corporation 2013

Figure 5-11. Basic built-in data type categories (2 of 2)

WA680 / VA6801.0

Notes:

Order of precedence (1 of 2)

Operators in order of precedence	Description
'abc', {k:v}, [a,b,c], (a,b,c)	String, dictionary, list, and tuple construction operators
C.a, f(...), s[i:j], s[i]	Attribute reference, function call, slicing and indexing
**, ~x, +x, -x	Exponentiation, bitwise NOT, unary plus, and unary minus
*, /, %	Multiplication, division, and modulo (remainder)
+, -	Addition or concatenation, and subtraction

© Copyright IBM Corporation 2013

Figure 5-12. Order of precedence (1 of 2)

WA680 / VA6801.0

Notes:

Order of precedence (2 of 2)

Operators in order of precedence	Description
<<, >>	Left bit-shift and right bit-shift
&, ^,	Bitwise AND, bitwise exclusive-OR, and bitwise OR
<, <=, >, >=, ==, !=	Comparison operators
is, is not	Identity test operators
in, not in	Membership test operators
not x	Boolean NOT
and, or	Boolean AND, Boolean OR
lambda	Lambda expression (anonymous function)

© Copyright IBM Corporation 2013

Figure 5-13. Order of precedence (2 of 2)

WA680 / VA6801.0

Notes:

Numeric data types (1 of 2)

- Integer
 - Instance of the `PyInteger` class
 - Limited in size to 32 bits
 - Literal consists of an optional sign and one or more digits without a decimal point
 - Examples: `123`, `-7`, `+12`
- Long
 - Instance of the `PyLong` class
 - Limited in size by the available memory
 - Literal consists of an optional sign, one or more digits without a decimal point and the letter "l" or "L"
 - Examples: `1234567L`, `-91`, `+21L`

© Copyright IBM Corporation 2013

Figure 5-14. Numeric data types (1 of 2)

WA680 / VA6801.0

Notes:

Numeric data types (2 of 2)

- **Float**
 - Instance of the `PyFloat` class
 - Floating point number with 64-bit precision
 - Literals can be:
 - Any number with a decimal point
 - Any number with a decimal point and an exponent expression (the letter “e” or “E”) followed by an integer with an optional sign
 - Examples: `3.2`, `-45.6`, `3.4e7`, `8765.43E-12`
- **Complex**
 - Instance of the `PyComplex` class
 - Complex number has a real part and an imaginary part
 - Literals are written as real part + imaginary part followed by the letter “j” or “J”.
 - Examples: `3.5 + 1.2j`, `2.0j`

© Copyright IBM Corporation 2013

Figure 5-15. Numeric data types (2 of 2)

WA680 / VA6801.0

Notes:

Sequence data type: Strings

- Instances of the `PyString` class
- Immutable sequence of characters
- Literal can be enclosed in single quotation marks, double quotation marks, or triple quotation marks
 - Triple quotation marks allow you to split the literal across multiple lines.
- Can be subscripted
 - First character of a string has an index of 0

```
'This is a string.'
This is a string.
"This one has 3 single quotes '''"
This one has 3 single quotes '''
\"Eureka!\""
"Eureka!""
s = "Jython"
s[1]
Y
```

© Copyright IBM Corporation 2013

Figure 5-16. Sequence data type: Strings

WA680 / VA6801.0

Notes:

An important characteristic of string methods is that they always return a copy of the string and never modify it in place because strings are immutable. There are also many functions that you can invoke against a string.

Common string functions and methods

`len(aSequence)`

- Returns the length of a sequence

`find(substring, [, start [, end]])`

- Returns the index in a string where the first occurrence of substring is found
- The optional start and end indexes can be used to restrict the scope of the search.

`lower()` and `upper()`

- Returns a copy of the string with all letters converted to lowercase or uppercase

`lstrip()`, `rstrip()` and `strip()`

- Returns a copy of the string with leading, trailing, or both, white space removed

`split([separator] [, maxSplit])`

- Returns a list of strings that are obtained from breaking the string at each occurrence of separator
- If separator is not specified, the white space character acts as a delimiter.
- The maxSplit parameter, if specified, limits the number of splits performed. The last string gets all the remaining characters.

© Copyright IBM Corporation 2013

Figure 5-17. Common string functions and methods

WA680 / VA6801.0

Notes:

The len function returns the length of a sequence. The find method returns the first index in a string where substring is found. The start and end indexes are optional and restrict the search within those bounds when supplied. The lower and upper methods return a copy of the string with all letters converted to either lowercase or uppercase, depending of which method you start. You can also invoke the lstrip, rstrip, and strip methods to return a copy of the string with leading, trailing or all white spaces that are removed from the string. One of the more frequently used methods is the split method. The reason that split is used so frequently is because the administrative objects usually return a string that has multiple values. To traverse each value within the string, you would split them and insert each value into a list before processing them. The split method returns a list of strings that are obtained from breaking a string at each occurrence of a particular separator. If the separator is not specified, the white space character acts as delimiter. The maxSplit parameter, if specified, limits the number of splits that are performed, hence leaving the last string with the remaining characters.

Sequence slicing

- The sequence slice notation `[i:j]` can be used to return a subset of a sequence
`aString[i:j]` returns the substring of `aString` starting at index `i` and to, but not including, index `j`
- Both `i` and `j` are optional and default to 0 and the length of the sequence.
- “Out-of-bounds” slices are handled gracefully:
 - The sequence size replaces an index that is too large.
 - An upper bound smaller than the lower bound returns an empty sequence

```
s = "WebSphere"
len(s)
9
s[0:3]
Web
s[3:9]
Sphere
s[::]
WebSphere
s[3:999]
Sphere
s[6:4]

s[-6:]
Sphere
```

© Copyright IBM Corporation 2013

Figure 5-18. Sequence slicing

WA680 / VA6801.0

Notes:

You can use sequence slicing to retrieve parts of a string, instead of split. The sequence slicing notation is `[i:j]` to return a subset of the sequence.

Sequence data type: Lists

- Instances of the `PyList` class
- Collection of heterogeneous objects that are mutable
 - Can be modified without creating a new list
 - Items do not must be all of the same type
- List items are contained between square brackets and separated by a comma.
- Indexes start at 0
- Can be sliced, concatenated, and nested
- `len(aList)` returns the length of a list

```

list = [1, "two", 3.0]
list[0]
1
list[:2]
[1, two]
list[len(list)-1:]
[3.0]
list2 = ["three", 4]
list[2] = list2
list
[1, two, [three, 4]]
list[2][0]
three

```

© Copyright IBM Corporation 2013

Figure 5-19. Sequence data type: Lists

WA680 / VA6801.0

Notes:

List is another commonly used sequence data type. The list is an instance of `PyList` class, which can contain a collection of heterogeneous objects that are mutable.

List methods (1 of 2)

`append(anItem)`

- Adds item `anItem` to the end of the list

`count(anItem)`

- Returns the number of times `anItem` occurs in the list

`extend(aList)`

- Adds the elements of list `aList` to the end of the list

`index(anItem)`

- Searches the first item in the list whose value is `anItem` and returns its index.
- Throws an exception if `anItem` is not found in the list

`insert(ind, anItem)`

- Inserts item `anItem` before the list element at index `ind`
- For example, `aList.insert(0, anItem)` inserts `anItem` at the front of the list, and `aList.insert(len(aList), anItem)` add to the end.

© Copyright IBM Corporation 2013

Figure 5-20. List methods (1 of 2)

WA680 / VA6801.0

Notes:

List methods (2 of 2)

`pop([ind])`

- Retrieves the item at index `ind` in the list, and removes it from the list.
- `pop()` returns and removes the last item in the list.

`remove(anItem)`

- Searches the first item in the list whose value is `anItem` and removes it
- Throws an exception if `anItem` is not found in the list

`reverse()`

- Reverses the order of the items in the list

`sort()`

- Sorts the items in the list in ascending order

© Copyright IBM Corporation 2013

Figure 5-21. List methods (2 of 2)

WA680 / VA6801.0

Notes:

List methods examples

```
aList = [1, "two", 3, 3, 3]
aList.append("two")
aList
[1, two, 3, 3, 3, two]
print aList.count(4), aList.count("two"), aList.count(3),
    aList.count(3.0)
0 2 3 3
aList.index(3)
2
aList.insert(3, "four")
aList
[1, two, 3, four, 3, 3, two]
aList.remove("two")
aList
[1, 3, four, 3, 3, two]
aList.sort()
aList
[1, 3, 3, three, four, two]
```

© Copyright IBM Corporation 2013

Figure 5-22. List methods examples

WA680 / VA6801.0

Notes:

This code is an example of how to create a list and use the list methods. The second statement appends the string two to the list; the string two is appended to the end of the list. The fourth statement uses the count method to find out how many times a certain string or number occurs within the list. If the string or number is not present, it returns 0 rather than throwing an error. The *index method* returns the first occurrence of the wanted number or string. Remove removes the first occurrence of the wanted string. Sort and reverse are useful features that can be useful when writing simple or elaborate algorithms.

List comprehensions

- Provide a quick way to create a list whose elements are derived from applying some function or expression to each element of an existing list
- Simplest form consists of:
[*expression* for *variableName* in *sequence*]
 - Returns a list by evaluating *expression* for each element in *sequence*
- Extended form can add a filter:
[*expression* for *variableName* in *sequence* if *condition*]
 - Tests *condition* before evaluating *expression* for each element

```
fruitList = [' pear', ' orange ', 'apple ']
[fruit.strip() for fruit in fruitList]
[pear, orange, apple]
```

© Copyright IBM Corporation 2013

Figure 5-23. List comprehensions

WA680 / VA6801.0

Notes:

This example creates a version of the original list with all of its elements that are stripped of any white space. This form provides a quick way to create a list whose elements are derived from applying some function or expression to each element of an existing list.

Sequence data type: Tuples

- Instances of the PyTuple class
- Collection of heterogeneous objects that are immutable
 - Similar to lists, but items cannot be changed
- Tuple items are contained between parentheses and separated by a comma
- Can be used when a constant list is needed

```
aTuple = (12, "WebSphere", 3.1415)
aTuple[1]
WebSphere
empty = ()                                # Empty tuple
empty
()
len(empty)
0
newTuple = aTuple, ("abc", "def")    # Nesting tuples
newTuple
((12, WebSphere, 3.1415), (abc, def))
```

© Copyright IBM Corporation 2013

Figure 5-24. Sequence data type: Tuples

WA680 / VA6801.0

Notes:

Tuples are a sequence data type that can hold a collection of items. Tuples are an instance of the PyTuple class. The main difference between the list data type and tuples, is that tuples are immutable. Since this data type is immutable, every time a tuple is modified a new tuple is created and returned.

Mappings: Dictionaries

- Instances of the `PyDictionary` or `PyStringMap` (if keys are of type string only) class
- Collection of heterogeneous objects that are mutable and accessed randomly
 - Indexed by unique keys that are used to access their corresponding value
 - Items are not stored in a particular order.
- Literals are formed as a comma-delimited list of `key:value` pairs that are surrounded by braces.
- Can be nested

```
dictionary = {"one": 1, "two": 2, "three": 3}
dictionary
{'two': 2, 'one': 1, 'three': 3}
len(dictionary)
3
dictionary["two"]
2
dictionary["three"] = 333
dictionary
{'two': 2, 'one': 1, 'three': 333}
```

© Copyright IBM Corporation 2013

Figure 5-25. Mappings: Dictionaries

WA680 / VA6801.0

Notes:

Dictionaries are collection of heterogeneous objects that are mutable and accessed randomly. They are indexed with unique keys that are used to access their corresponding values.

Boolean expressions

- Jython 2.1 does not have a built-in Boolean type.
- Values are considered true except for:
 - Value of 0 or None
 - An empty string, list, or mapping
- Boolean expressions evaluate to either 1 for true or 0 for false.
- A Boolean type, `bool`, was added to Python 2.3, along with two new constants `True` and `False`.

```

2 > 1
1
"c" < "a"
0
list = []
if list:
    print "True"
else:
    print "False"

False

```

© Copyright IBM Corporation 2013

Figure 5-26. Boolean expressions

WA680 / VA6801.0

Notes:

Simple statements: Assignment

- Binds an object to an identifier
- Syntax: *variable assignmentOperator value*
- Simple assignment operator: =
 - Example: `length = 12`
- Augmented assignment operators:
 - `+=`, `-=`, `*=`, `/=`, `**=`, `%=`, `<=>`, `&=`, `|=`, `^=`
 - `x op= y` is equivalent to `x = x op y`
- An *unpacking assignment* allows you to assign multiple values to multiple variables in one statement:
 - For example:
 - `a, b = "Hello ", "world"`
 - equivalent to:
 - `a = "Hello "; b = "world"`

© Copyright IBM Corporation 2013

Figure 5-27. Simple statements: Assignment

WA680 / VA6801.0

Notes:

The assignment statement in Jython binds an object to an identifier. The syntax for assigning a variable is simple. You specify the name of the variable, the assignment operator, and the value to set. You do not have to specify a type for the variable. There are also various augmented assignment operators such as `+=`, `-=`, and so forth. Unpacking assignment allows you to assign multiple values to multiple variables in one statement. It is a good programming practice to assign one variable on a line.

The print statement

- Does the following actions:
 - Evaluates an expression
 - Converts the result to a string if needed
 - Writes the string to standard output or a specified file
- Syntax:
 - `print expression`
 - `print >> aFile, expression`
- Automatically inserts a newline character at the end of the line
 - A trailing comma suppresses the newline character.
- Multiple expressions can be printed in one statement by separating them with a comma.

```
i = 2**16
print "The value of i is", i
The value of i is 65536
print "Hello",; print "world!"
Hello world!
```

Extra space is automatically inserted before the value of i

New line is suppressed because of comma

© Copyright IBM Corporation 2013

Figure 5-28. The print statement

WA680 / VA6801.0

Notes:

Other simple statements (1 of 2)

- **del statement**
 - Removes a variable, or an item in a sequence
 - Syntax:
 - `del anIdentifier | aSequenceItem | aSequenceslice`

```
aList = ["red", 24, "white", 25, 60, 71, "blue"]  
del aList[1]  
aList  
['red', 'white', 25, 60, 71, 'blue']  
del aList[2:5]  
aList  
['red', 'white', 'blue']  
del aList           # Deletes the aList variable
```

- **pass statement**
 - Does nothing
 - Can be used when a statement is syntactically required but no logical action is wanted

© Copyright IBM Corporation 2013

Figure 5-29. Other simple statements (1 of 2)

WA680 / VA6801.0

Notes:

Compound statements: if elif else

- Provides support for conditional logic by evaluating an expression and running the appropriate code block that is based on its Boolean value
- Syntax:

```
if expression:  
    code block  
elif expression:  
    code block  
else:  
    code block
```

- Can have zero or more elif parts and zero or one else part
- Can be nested
 - Indentation determines which if statement they belong to

```
if expression1  
    if expression2  
        if expression3  
            code block1  
    else:  
        code block2
```

← else clause belongs to first nested if statement because of indentation

© Copyright IBM Corporation 2013

Figure 5-30. Compound statements: if elif else

WA680 / VA6801.0

Notes:

The colon is not required to indicate the start of code blocks. Using the colon to indicate that start of a code block is helpful for humans and for syntax highlighting programs. Indentation is important in Jython. Code blocks must be lined up or the wrong block of code gets ran.

The while statement

- Loop statement that evaluates an expression and runs the specified block while it remains true
- Syntax:

```
while expression:  
    code block  
else:  
    code block
```

- else clause is optional:

- Run when the expression becomes false, but not if the loop is terminated by a break statement

```
x = 12  
while x > 0:  
    print x,  
    x -= 1  
  
12 11 10 9 8 7 6 5 4 3 2 1
```

© Copyright IBM Corporation 2013

Figure 5-31. The while statement

WA680 / VA6801.0

Notes:

The for statement

- Iterates over the items of a sequence, from first to last
 - Each element is assigned to a variable, and the block is run for each element.
- Syntax:
 - for variableName in aSequence:
 - > code block
 - else:
 - > code block
- else clause is optional:
 - Run when the loop finishes iterating over all the items in the sequence, but not if it is terminated by a break statement

© Copyright IBM Corporation 2013

Figure 5-32. The for statement

WA680 / VA6801.0

Notes:

Example: The for statement

```
aList = ["dolphin", "shark", "seal"]
for x in aList:
    print x, len(x)
else:
    print "Length of each element in list was printed successfully"

dolphin 7
shark 5
seal 4
Length of each element in list was printed successfully
```

© Copyright IBM Corporation 2013

Figure 5-33. Example: The for statement

WA680 / VA6801.0

Notes:

Using the range() function in a for loop

- The `range()` built-in function is used to generate a list that contains an arithmetic progression of integers.
- Can be used emulate a traditional `for` loop
- Syntax: `range([start,] stop [, step])`

```
range(12)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

- To iterate over a sequence with an index, use `range()` and `len()`

```
aList = ["WebSphere", "Scripting", "and", "Automation"]
for i in range(len(aList)):
    print i, aList[i]

0 WebSphere
1 Scripting
2 and
3 Automation
```

© Copyright IBM Corporation 2013

Figure 5-34. Using the `range()` function in a for loop

WA680 / VA6801.0

Notes:

Exceptions

- When an error occurs, the interpreter automatically prints an error message and a stack trace.
 - For example:

```
emptyList = []
print emptyList[0]
Traceback (innermost last):
  File "<input>", line 1, in ?
IndexError: index out of range: 0
```

- Last line displays the exception type and shows the exception argument if available
- To handle unexpected events gracefully, you can catch and raise exceptions with the following statements:
 - try statement
 - raise statement

© Copyright IBM Corporation 2013

Figure 5-35. Exceptions

WA680 / VA6801.0

Notes:

An exception can have an associated value, also known as the exception's argument. The presence and type of the argument depend on the exception type. If an unhandled exception has an argument, it is printed as the last part of the message. To handle unexpected events gracefully, you can catch and raise exceptions by using the `try` statement or the `raise` statement. These situations will be covered further in the next couple of slides.

The try statement

- Allows you to catch and handle any resulting exceptions

- Syntax has two forms:

- Form 1:

```
try:  
    code block  
finally:  
    code block
```

- Form 2:

```
try:  
    code block  
except exceptionSpecifier:  
    code block  
else:  
    code block
```

- Runs the try code block; then the finally code block regardless of whether an exception occurred

- More than one except clause can be specified to handle different exceptions.
- else clause is optional and must follow all except clauses
- Runs the try code block. If an exception is raised, except clauses are searched for one to handle it, and the corresponding code block is run. If no exception is raised, the else code block is run.

© Copyright IBM Corporation 2013

Figure 5-36. The try statement

WA680 / VA6801.0

Notes:

Forms of the `except` clause

- `except exceptionType:`
 - Catches only exceptions of the `exceptionType` class
- `except exceptionType, exceptionInstance:`
 - Catches only exceptions of the `exceptionType` class and assigns the raised exception object to the `exceptionInstance` variable
- `except (exceptionType1, exceptionType2, ...):`
 - Catches any exception of a type in the list
- `except:`
 - Catches all exceptions

```
try:  
    12/0  
except ZeroDivisionError, e:  
    print "You cannot divide by zero:", e  
You cannot divide by zero: integer division or modulo
```

© Copyright IBM Corporation 2013

Figure 5-37. Forms of the `except` clause

WA680 / VA6801.0

Notes:

The raise statement

- Causes a specified exception to be raised
 - Syntax:

```
raise [exceptionType [, attribute [, tracebackObject]]]
```
 - If no argument is specified, the current exception is reraised
 - Valid only within an `except` block
 - `exceptionType`, if specified, can be the name of an exception class or instance
 - `attribute`, if specified, is passed to the exception constructor
 - `raise exceptionType(attribute)` syntax is also acceptable
 - `tracebackObject`, if specified, is a traceback object that is associated with the exception

```
raise NameError("The variable aList is not defined.")  
Traceback (innermost last):  
  File "<input>", line 1, in ?  
NameError: The variable aList is not defined.
```

© Copyright IBM Corporation 2013

Figure 5-38. The raise statement

WA680 / VA6801.0

Notes:

User-defined exceptions

- You can create your own exception class
 - Should be a direct or indirect subclass of the `Exception` class
- Common exception classes include:
 - `ValueError`
 - Raised when an argument of the right type but with an inappropriate value is received
 - `ArithmeticError`
 - Base class for exceptions for arithmetic error, including: `OverflowError`, `ZeroDivisionError`, `FloatingPointError`
 - `EOFError`
 - Raised when an input operation encounters an end-of-file condition (EOF)

© Copyright IBM Corporation 2013

Figure 5-39. User-defined exceptions

WA680 / VA6801.0

Notes:

File input and output: Functions and methods

- File access is supported through the built-in function `open()` and built-in file methods.
- `open(fileName, mode)`
 - Returns a file object
 - Mode is a string that describes how the file is used
- Common built-in file methods:
 - `read(size)`
 - Reads some quantity of data and returns it as a string
 - `size` is an optional numeric argument that specifies the maximum number of bytes to read
 - `readline()`
 - Reads a single line from the file
 - The newline character (`\n`) is left at the end of the string.
 - `write(string)`
 - Writes the contents of `string` to the file
 - `close()`
 - Closes the file and releases resources that are allocated to the open file

© Copyright IBM Corporation 2013

Figure 5-40. File input and output: Functions and methods

WA680 / VA6801.0

Notes:

Working with files in Jython is easy. File access is supported through the built-in method `open()` and other methods. The `open` method returns a file object, and the file is opened with the specified mode.

Modules

- A module is a file that contains Jython code for importing code
 - Contains variable, function and class definitions, and statements
 - Modules or the main top-level program imports modules with the `import` statement
 - Importing a module for the first time runs it.
 - The file name is the module name with a `.py` extension.
- When importing a module, the `sys.path` variable specifies the search path with these rules:
 - Current directory first
 - Next, if defined, list of directories that are specified by the environment variable `PYTHONPATH`
 - Finally, search continues following an installation-dependent default path

```
sys.path
['.', 'C:\\Programfiles\\IBM\\WebSphere\\AppServer
 \\optionalLibraries\\jython\\Lib']
```

© Copyright IBM Corporation 2013

Figure 5-41. Modules

WA680 / VA6801.0

Notes:

The three building blocks of a Jython program are modules, functions, and classes. A module is a collection of statements that define variables, functions, and classes. Modules are the primary units that are used for importing code. Within a module, the module name is available as the value of the global variable `__name__`.

Functions

- Callable objects are defined:

```
def functionName( [parameters] ) :  
    code block
```

- Keyword `def` introduces a function definition
 - Must be followed by the function name and a parenthesized list of 0 or more parameters
 - Statements that form the body of the function follow and must be indented
 - First statement of the function body can optionally be a string literal representing the function documentation string, or *docstring*.
- Parameters are passed via *pass by object reference*
- The `return ([expression])` statement ends the execution of a function and returns the value of *expression* if specified.
 - `return` without an expression argument returns `None`.

© Copyright IBM Corporation 2013

Figure 5-42. Functions

WA680 / VA6801.0

Notes:

Function argument example

```

def func(a, b=0, c="WebSphere", *d, **e):
    print a, b, c, d, e

func(1, 2, 3)
1 2 3 () {}

func(1, 2)
1 2 WebSphere () {}

func(1)
1 ← Only one parameter passed;
      others use default values

func(1, 2, 3, 4, 5, 6)
1 2 3 (4, 5, 6) {}

func(b="IBM", a=1)
1 IBM WebSphere () {}

func("IBM", "WebSphere", "Application", "Server", "Version 8.5")
IBM WebSphere Application (Server, Version 8.5) {}

func("IBM", "WebSphere", "Application", "Server", Version="8.5")
IBM WebSphere Application (Server,) {Version: 8.5}

```

© Copyright IBM Corporation 2013

Figure 5-43. Function argument example

WA680 / VA6801.0

Notes:

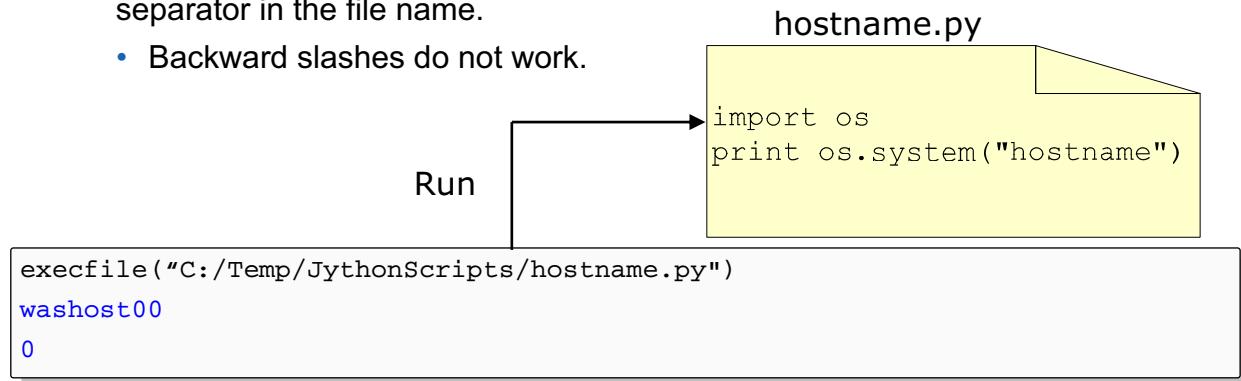
Look at the function that is called *func* that is defined here. It takes five parameters: a, b, c, d, and e. Notice that parameter d has an asterisk in front of it; the single asterisk means that it is a catch all Tuples. Also, notice that parameter e, has two asterisks in front of it; it is a dictionary entry. The purpose of the defined function that is called *func* in this case is to print every parameter.

*d puts all left-over non-keyword arguments into a tuple that is called d.

**e puts all left-over keyword arguments into a dictionary that is called e.

The execfile() function

- `execfile(fileName)`
 - Runs the Jython statements in the file that is specified by `fileName`
 - Does not return a value
 - Can be used to:
 - Call a Jython script from another script
 - Run multiple scripts or the same script multiple times in the same interactive session
 - If running in wsadmin, you must use the forward slash character (/) as the path separator in the file name.
 - Backward slashes do not work.



© Copyright IBM Corporation 2013

Figure 5-44. The execfile() function

WA680 / VA6801.0

Notes:



Jython standard library

- Jython includes a rich set of standard modules that allow you to perform functions such as:
 - Interact with the Jython run time environment
 - Interface with the operating system
 - Perform file and string pattern matching
 - Use mathematical functions
 - Manipulate dates and times
 - Perform data compression and archiving
 - Access the Internet and handle Internet data
 - Perform automated unit testing

© Copyright IBM Corporation 2013

Figure 5-45. Jython standard library

WA680 / VA6801.0

Notes:

Jython includes a rich set of standard modules that allow you to perform functions such as:

- Retrieving command-line arguments
- Redirecting error output
- Terminating a program

Commonly used standard modules

Module name	Functions provided
sys	Interface to the Jython execution environment Examples: Retrieve command-line arguments, access standard output
os	Interface to the operating system Examples: Return the current working directory, retrieve the local line separator character
shutil	Simplified file and directory management Examples: Copy a file, move a file
glob	File pattern matching; returns a list that contains the names of files that match a directory wildcard search pattern
re	String pattern matching with regular expressions Example: Test if a string contains a particular regular expression pattern
datetime	Date and time manipulation Example: Get today's date, perform date arithmetic
pickle	Jython object serialization Examples: Serialize an object, deserialize it
unittest	Automated unit test framework Examples: Create a test case and a test, run a test suite

© Copyright IBM Corporation 2013

Figure 5-46. Commonly used standard modules

WA680 / VA6801.0

Notes:

The sys module

- Contains variables and functions that are related to the Jython run time environment
- Variables that are defined include:
 - argv
 - List of the command-line arguments that are passed to a script
 - Under normal circumstances, `argv[0]` is the name of the script itself.
 - If using wsadmin, `argv[0]` is the first passed argument.
 - modules
 - Dictionary of all currently loaded modules
 - Keys are strings that represent the modules names
 - Values are the modules themselves
 - path
 - List of strings that represent the interpreter search path for modules
 - stdin, stdout, stderr
 - References to the system's standard input, output, and error streams
 - Can be used as file objects
- Available functions include:
 - `exit(exitCode)`
 - Exits Jython with the specified exit code

© Copyright IBM Corporation 2013

Figure 5-47. The sys module

WA680 / VA6801.0

Notes:

The sys module contains variables and functions that are related to the Jython run time environment. You can use the `sys.argv` to retrieve the list of command-line arguments that were passed to the script. `sys.argv[0]` is the name of the script itself; however in wsadmin, `sys.argv[0]` is the first passed argument. `sys.modules` returns a dictionary of all the currently loaded modules. `sys.path` is where loaded modules are found within the directories on that computer.

The os module

- Provides access to operating system calls and file manipulation functions
- Variables that are defined include:
 - `linesep`
 - The line separator string for the platform
 - Or `java.lang.System.getProperty("line.separator")`
 - `sep`
 - The character that is used by the operating system to separate elements of a complete file name
 - Same as `java.io.File.separator`
 - `path`
 - A separate module, usually referred as `os.path`, which contains functions for manipulating path names
- Available functions include:
 - `getcwd()`
 - Returns the path name of the current working directory
 - `system(command)`
 - Runs the `command` string in a command shell

© Copyright IBM Corporation 2013

Figure 5-48. The os module

WA680 / VA6801.0

Notes:

The os module provides functions for interacting with the operating system, including file, directory, and process management. When starting an os module function, the `OSError` exception is raised if something goes wrong during the actual system call. The OS module has variables to find the line separator, file separator, and the path variable for the platform. Most importantly the OS module allows your Jython script to run OS level commands by starting the function `system`.

os module variable and function examples

```
os.linesep
'\r\n'
os.sep
'\\'
os.getcwd()
'C:\Program
 Files\IBM\WebSphere\AppServer\profiles\SamplesProfile\bin'
os.system("dir wsadmin.*")
Volume in drive C has no label.
Volume Serial Number is 0C25-A801

Directory of C:\Program
Files\IBM\WebSphere\AppServer\profiles\SamplesProfile\bin

03/24/2013  12:12 PM      319 wsadmin.bat
              1 File(s)      319 bytes
              0 Dir(s)  54,416,306,176 bytes free
0
```

© Copyright IBM Corporation 2013

Figure 5-49. os module variable and function examples

WA680 / VA6801.0

Notes:

Using Java libraries

- Jython allows you to easily use existing Java library code
 - Use the `import` statement to import a package or class
- You can access:
 - The standard Java API
 - A user developed Java library

```
from java import util           Imports java.util.*
v = util.Vector(2)
v
[]
v.add(12)
1
v
[12]
v.add("Thirteen")
1
v
[12, Thirteen]
```

Imports java.util.*

Instantiates a vector and uses the vector add() method to add elements to it

© Copyright IBM Corporation 2013

Figure 5-50. Using Java libraries

WA680 / VA6801.0

Notes:

Jython allows you to easily use existing Java library code. Use the `import` statement to import a Java library from the wanted package or class. You can import the standard Java library and user developed Java libraries as well. You can also use this feature to prototype Java applications.

Classes

- **class statement**
 - Used to create a class
 - Syntax:

```
class ClassName [ (baseClasses) ] :  
    code block
```
 - *ClassName* is any legal identifier
 - *code block* contains the Jython statements that define the class
 - *baseClasses* are the optional names of parent classes that the class inherits
 - Provides support for multiple inheritance of Jython classes
- Methods are defined by `def` statements within the class block.
 - First argument must be the `self` object
 - The caller is not required to supply a value for it.
 - Called with “dot” notation

© Copyright IBM Corporation 2013

Figure 5-51. Classes

WA680 / VA6801.0

Notes:

Class example

```
class GreetingsClass:  
    numOfGreetings = 2  
    def sayHello(self): print "Hello!"  
    def sayBonjour(self): print "Bonjour!"  
  
GreetingsClass  
<class main.GreetingsClass at 573973046>  
GreetingsClass.numOfGreetings  
2  
g = GreetingsClass()  
g.sayHello()  
Hello!  
g.sayBonjour()  
Bonjour!
```

© Copyright IBM Corporation 2013

Figure 5-52. Class example

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe the basic elements of Jython

© Copyright IBM Corporation 2013

Figure 5-53. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. How does Jython determine the start and end of a logical code block?
2. Name the three sequence types that are provided by Jython.
3. Which looping statement is used to iterate over sequences?
4. Which variable provides access the command-line arguments that are passed to a Jython script?
5. True or false: Jython supports the inheritance of multiple Java classes.

© Copyright IBM Corporation 2013

Figure 5-54. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

4.

5.

Checkpoint answers

1. How does Jython determine the start and end of a logical code block?

Answer: Using indentation

2. Name the three sequence types that are provided by Jython.

Answer: Strings, lists, and tuples

3. Which looping statement is used to iterate over sequences?

Answer: The for statement

4. Which variable provides access the command-line arguments that are passed to a Jython script?

Answer: The sys.argv variable returns a list of the command-line arguments.

5. True or false: Jython supports the inheritance of multiple Java classes.

Answer: False

© Copyright IBM Corporation 2013

Figure 5-55. Checkpoint answers

WA680 / VA6801.0

Notes:

Unit 6. Jython development by using the IADT

What this unit is about

This unit provides an overview of the IBM Assembly and Deploy Tools and describes its Jython development and debugging features.

What you should be able to do

After completing this unit, you should be able to:

- Provide an overview of the IBM Assembly and Deploy Tools (IADT)
- Describe the IADT features that are used for Jython development

How you will check your progress

- Checkpoint
- Lab exercises

References

IBM Assembly and Deploy Tools for WebSphere Administration

<http://www.ibm.com/support/docview.wss?uid=swg27021755>



Unit objectives

After completing this unit, you should be able to:

- Provide an overview of the IBM Assembly and Deploy Tools (IADT)
- Describe the IADT features that are used for Jython development

© Copyright IBM Corporation 2013

Figure 6-1. Unit objectives

WA680 / VA6801.0

Notes:

IBM Assembly and Deploy Tools for WebSphere Administration

- Publish server-side code to WebSphere V8.5
- Jython scripting editor and debugger
- XML deployment descriptor, binding editors, and code validators
- Tools for assembling and deploying OSGi applications, bundles, fragments, and composites
- Tools for JAX-WS web service deployment
- Tools for JDBC providers and access to databases
- EJB deployment
- Enhanced EAR editor

© Copyright IBM Corporation 2013

Figure 6-2. IBM Assembly and Deploy Tools for WebSphere Administration

WA680 / VA6801.0

Notes:

The IBM Assembly and Deploy Tools help WebSphere Administrators deploy applications to a WebSphere Application Server. The tools include many simple wizards and visual editors that support the Java EE and OSGi programming models and help you configure and manage your applications and your WebSphere Application Server. The tools are not licensed for application development purposes. The tools are primarily used for creating automation scripts, configuring applications for deployment, and deploying applications to WebSphere Application Server Version 8.0 and 8.5.



Jython support

- Color-coded keyword highlight feature
- Statement completion assistance
- Script execution
- Debugging support
- Integration with WebSphere command assistance feature

© Copyright IBM Corporation 2013

Figure 6-3. Jython support

WA680 / VA6801.0

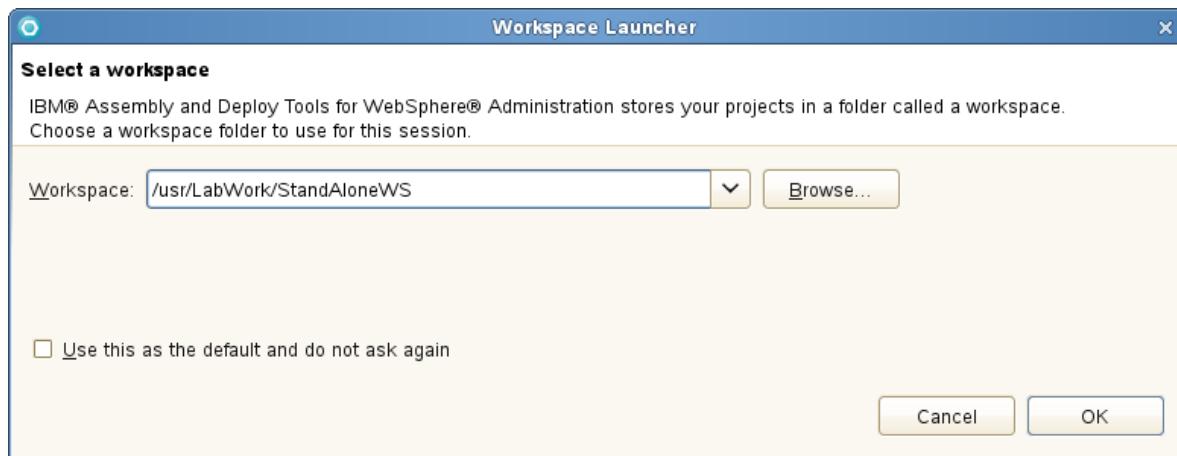
Notes:

Restriction: The IBM Assembly and Deploy Tools provides tools and support for IBM WebSphere Application Server Version 8.0 and 8.5. The IBM Assembly and Deploy Tools do not support earlier versions of WebSphere Application Server, feature packs, or applications that are created for older WebSphere Application Server versions.



Selecting a workspace

- When you start the IADT you can either select an empty workspace folder, or an existing workspace folder.
- Use an empty workspace folder if you are creating a Jython Project.
- You can import Jython scripts that exist into a workspace.



© Copyright IBM Corporation 2013

Figure 6-4. Selecting a workspace

WA680 / VA6801.0

Notes:

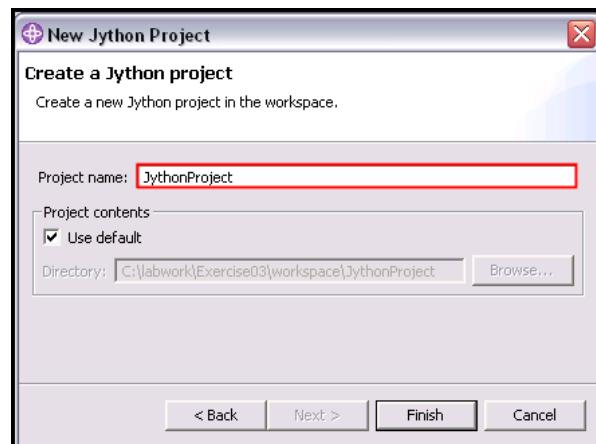
The workspace chooser dialog allows you to choose the location of your workspace. This dialog is displayed on first startup in the absence of a -data argument. The default location that is provided by this dialog is a workspace child of your home directory.

Unless you have an existing workspace from a previous Eclipse version, you can keep this default or choose some other location. You should not store your workspace inside the Eclipse installation directory because that makes it more difficult to upgrade to a newer version of Eclipse. You should not copy or move the workspace directory because it might contain metadata with absolute file system paths, which is invalid if the workspace is copied elsewhere.



Creating Jython projects

- A Jython project is a folder of Jython script files
- To create a Jython project:
 - Select **File > New > Project**
 - Select **Jython > Jython Project**
 - Set the project name
 - Click **Finish**



© Copyright IBM Corporation 2013

Figure 6-5. Creating Jython projects

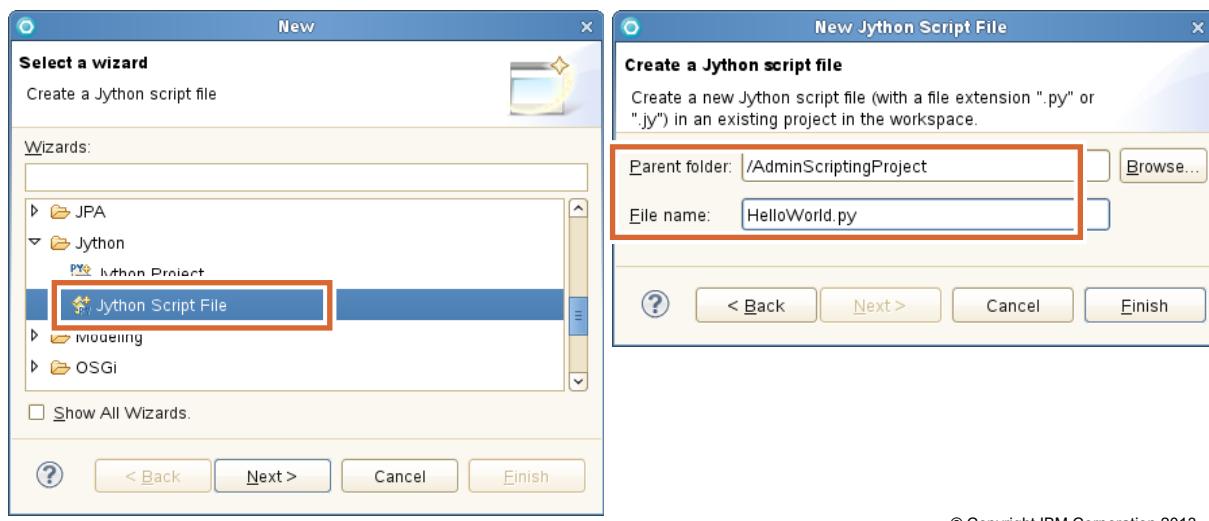
WA680 / VA6801.0

Notes:

The IBM Assembly and Deploy Tools includes many wizards to create projects that include Java EE, Java, XML, and OSGi projects. In addition to project creation wizards, there are several wizards to create different file types that include Jython scripts, servlets, JSPs, EJBs, test cases, web services, and many more. Refer to the IBM Assembly and Deploy Tools documentation.

Creating Jython script files

- To create a Jython script:
 - Select **File > New > Other**
 - Select **Jython > Jython Script File**
 - Set the parent folder and file name
 - Click **Finish**



© Copyright IBM Corporation 2013

Figure 6-6. Creating Jython script files

WA680 / VA6801.0

Notes:

The Jython scripts are edited by using the Jython editor.



Perspective and views

- Perspectives
 - Perspectives show a predetermined number of views to facilitate a role, like development.
 - Each perspective provides a set of functionality that is aimed at accomplishing a specific type of task.
 - For example, the **Debug** perspective contains the views that are used while debugging Jython scripts
 - The default perspective is Java EE
- Views
 - Views support different editors, and provide alternative presentations and ways to navigate the information in the workbench.
 - For example, the **Navigator** view displays projects and other resources.

© Copyright IBM Corporation 2013

Figure 6-7. Perspective and views

WA680 / VA6801.0

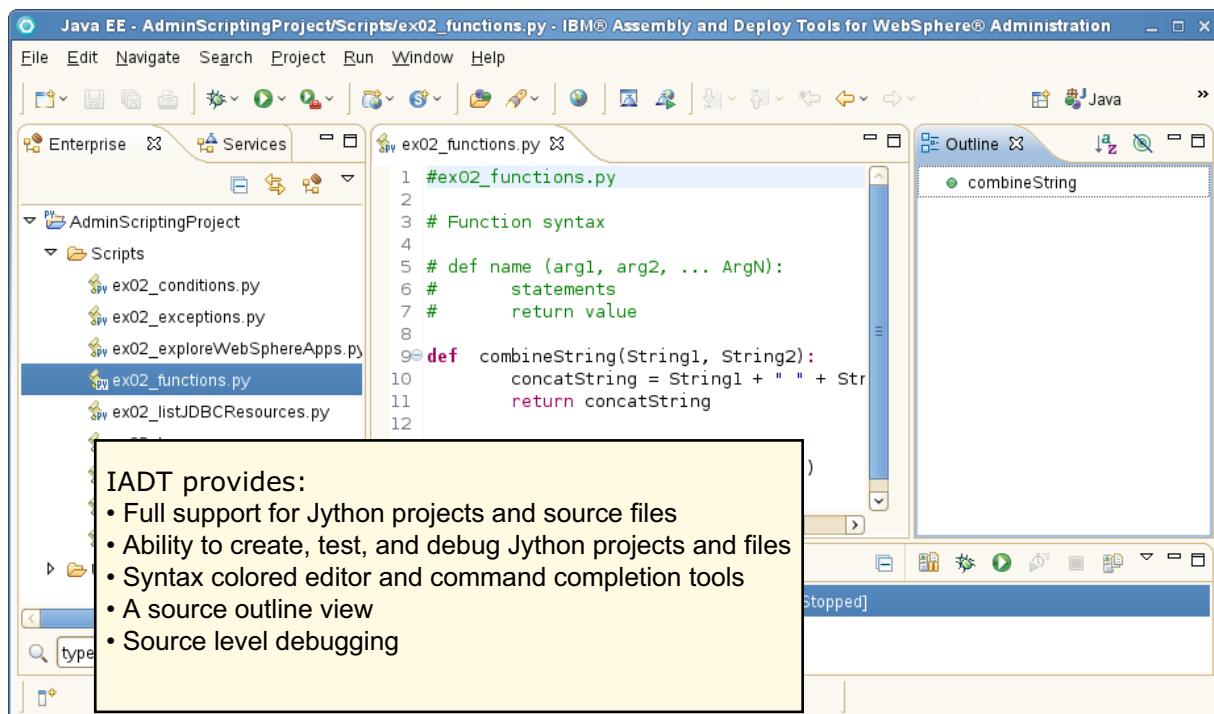
Notes:

A perspective defines the initial set and layout of views in the workbench window.

A view might be displayed by itself, or stacked with other views in a tabbed notebook. You can change the layout of a perspective by opening and closing views and by docking them in different positions in the workbench window.



Create, test, and debug Jython scripts



© Copyright IBM Corporation 2013

Figure 6-8. Create, test, and debug Jython scripts

WA680 / VA6801.0

Notes:

The Jython editor has many text editing features, such as syntax highlighting, unlimited undo or redo, and automatic tab indentation. When tagging a comment in a Jython script with "#TODO", the editor automatically creates a corresponding task as a reminder in the Tasks view.

If you open the task later, the editor automatically synchronizes to that TODO entry in the script source. Other helpful features are content assist and tips that provide a list of acceptable continuations that depend on where the cursor is located.

Jython editor features

- Color syntax highlight feature

- Keywords (orange)
- Wsadmin (red)
- Comments (green)
- Strings (blue)

```

#  
# TODO: enter JYTHON code and save  
  
print "This is a sample script."  
  
variableValue=6  
  
if variableValue==6:  
    print AdminApp.list()  
else:  
    print "Exiting script."

```

- Code completion

- Cursor on code
- Press **Ctrl+Space**
- Select code from list

```

#  
  
print "This is a sample script."  
  
variableValue=6  
  
if variableValue==6:  
    print AdminApp.

```

AdminApp.help()
AdminApp.help (method or option)
AdminApp.list()
AdminApp.listModules (application name, options)
AdminApp.listModules (application name)

© Copyright IBM Corporation 2013

Figure 6-9. Jython editor features

WA680 / VA6801.0

Notes:

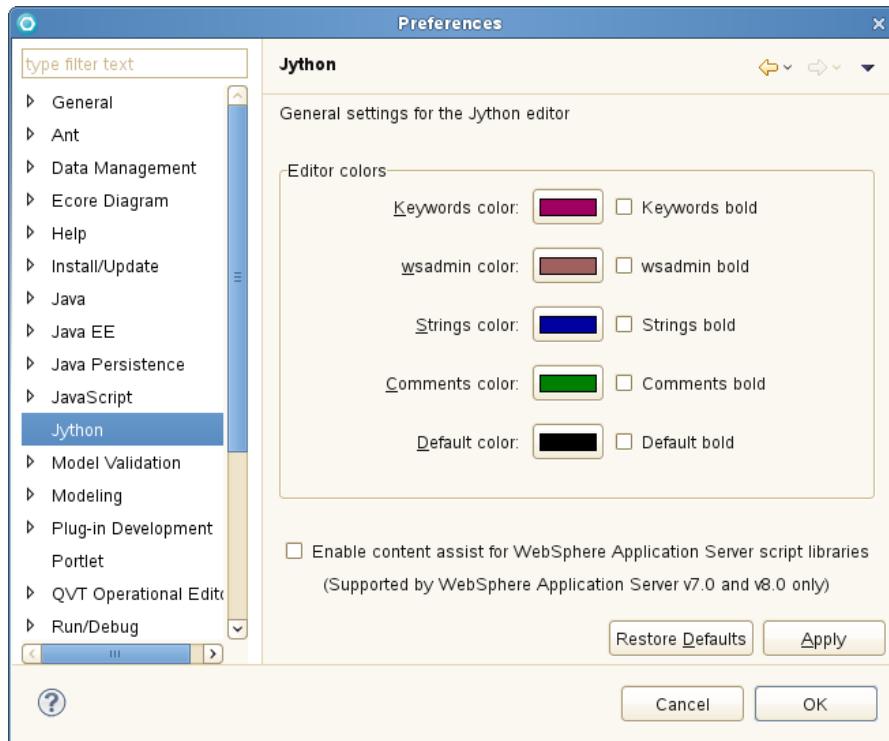
In the Jython editor, press **Ctrl + Space** on code to complete. This action provides a list of available code completions.

If you select a line in the content assist list, you can view information for that line. Clicking or pressing **Enter** on a selected line in the list inserts the selection into the editor.

The Jython editor is not integrated to a compiler. As a result, the Jython editor does not check syntax on your scripts.

Color preferences

- Select **Windows > Preferences**
- Select **Jython**
- Change associated colors
- Click **OK**



© Copyright IBM Corporation 2013

Figure 6-10. Color preferences

WA680 / VA6801.0

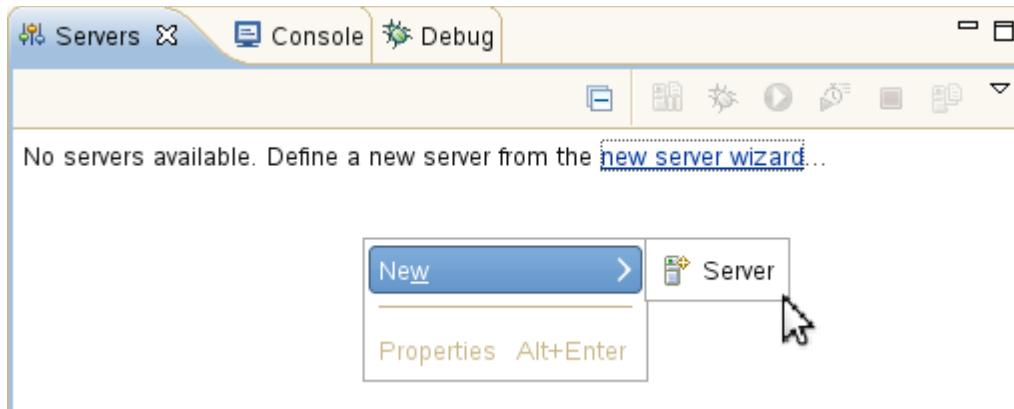
Notes:

To display the Jython script library as content assists or tips in the Jython editor, select and enable the **Enable content assist for WebSphere Application V7 script libraries** check box.



Defining a target server (1 of 3)

- Select the Servers view
- Click the link **new server wizard**
- Or right-click and select **New > Server**



© Copyright IBM Corporation 2013

Figure 6-11. Defining a target server (1 of 3)

WA680 / VA6801.0

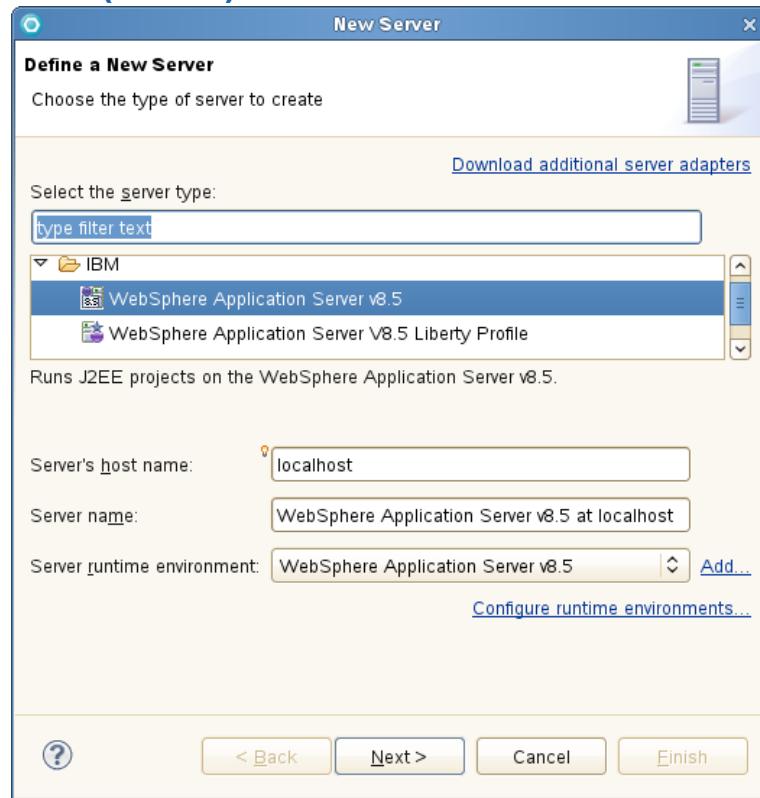
Notes:

You can create a server to identify the runtime environment that you want to use for testing your project resources. The term that creates a server defines creating a pointer from the workbench to an existing installation of an application server or server profile.



Defining a target server (2 of 3)

- The new server wizard guides you through the configuration
- Select the server type
 - Host name
 - Server name
 - Server runtime environment
- Click **Next**



© Copyright IBM Corporation 2013

Figure 6-12. Defining a target server (2 of 3)

WA680 / VA6801.0

Notes:

In the Select the server type list, select the type of server or test environment where you want to publish or test your resources.

In the Server host name field, you can provide the fully qualified DNS name or IP address of the host machine that where the server is running. By default, this field is pre-filled with the default address: localhost.

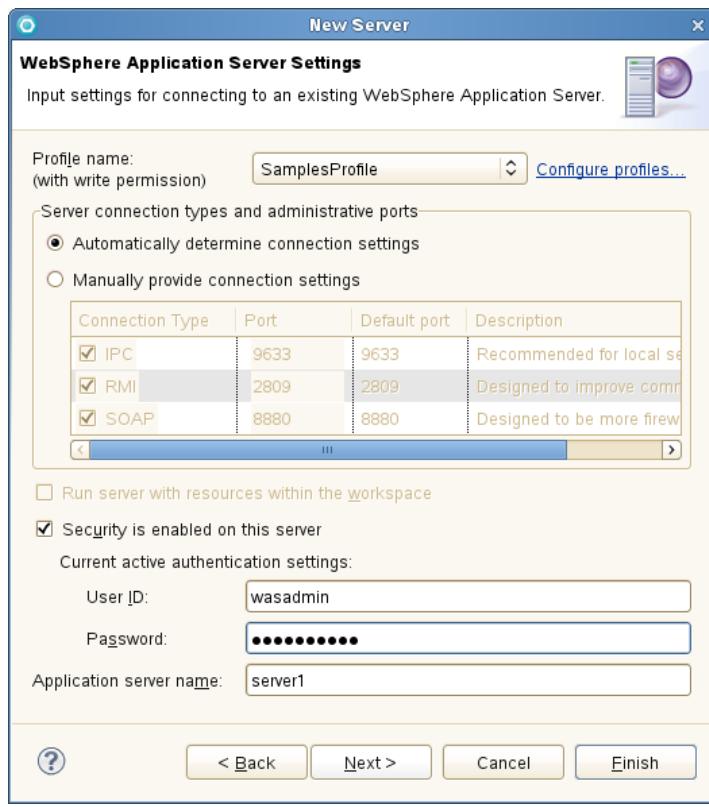
(Optional) In the Server name field, type a label to identify this server entry in the Servers view. By default, this field is completed with the following naming convention: server type at host name.

In the Server runtime environment list, select the runtime environment of an application server for compiling, testing, or running your application. To modify the server runtime environment, select the Configure runtime environments link. To add a server runtime environment, select the Add link.



Defining a target server (3 of 3)

- Select the profile name
- If security is enabled on the server, supply the authentication data
 - User ID
 - Password
 - Server name
- Click **Finish**



© Copyright IBM Corporation 2013

Figure 6-13. Defining a target server (3 of 3)

WA680 / VA6801.0

Notes:

Profile name: In the drop-down list, select the name of the profile of WebSphere Application Server. A profile is the set of files that define the runtime environment.

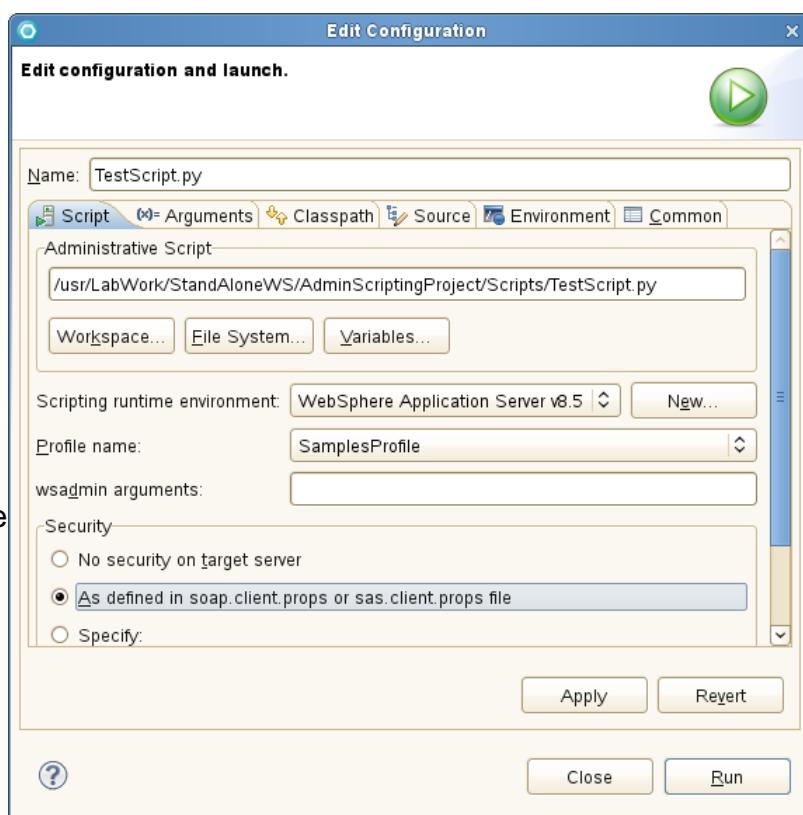
The **Automatically determine connection settings** option is available only for local servers and by default is selected when working with a local server. It retrieves the port values directly from the configuration files of the server that is defined in its profile and attempts to connect to one of these available ports.

Security is enabled on this server: Enables the security feature that comes with WebSphere Application Server. When security is not enabled, all other security settings are ignored.

Application server name: Specifies a logical name for the application server. For WebSphere Application Server, the logical name is unique and assigned to a server that distinguishes it from all other server instances within the node. This server name must already be created in the application server and its default setting is server1.

Script launcher

- To run a Jython script:
 - Right-click the script name in the Navigator
 - Select **Run As > Administrative Script**
 - Select a scripting runtime environment
 - Select a profile name
 - Specify security credentials
 - Click **Run**



© Copyright IBM Corporation 2013

Figure 6-14. Script launcher

WA680 / VA6801.0

Notes:

In the **Administrative Script** field, complete either of the following to specify the location of your script file:

Set the path to the script file.

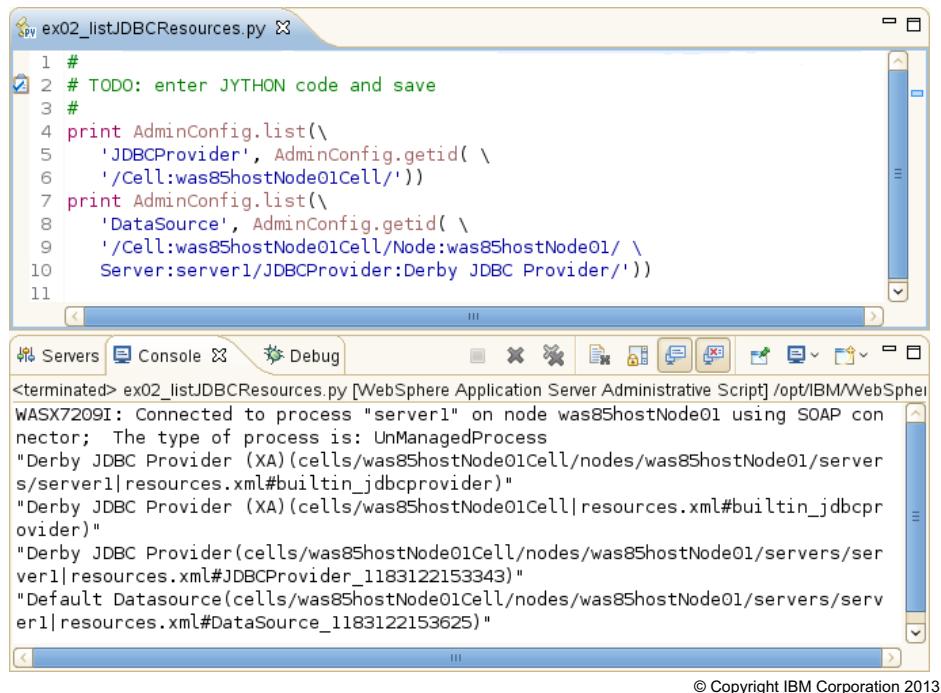
Set the Scripting runtime and profile from the list of installed WebSphere Application Server environment and profiles. If the profile remains **[Unspecified]**, the IADT chooses the profile that is assigned to the default role.

To pass arguments to the Jython script, do not specify them in this wsadmin arguments field, as this field is target for the wsadmin scripting client. Instead, select the **Arguments** tab and in the Program arguments text box specify the program arguments.

In the **Security** section, specify the security properties if security is enabled on the target server.

Running a script and viewing output in the Console

- A connection with the target server is made, the scripts run, and output is displayed in the Console view.



The screenshot shows the WebSphere Application Server Administrative Scripting interface. At the top, there is a script editor window titled "ex02_listJDBCResources.py" containing Python code. Below the script editor is a "Console" tab selected in the navigation bar. The console window displays the output of the script execution, which lists various JDBC resources on the server. The output includes messages like "Connected to process" and a list of "Derby JDBC Provider" entries. The bottom right corner of the interface has a copyright notice: "© Copyright IBM Corporation 2013".

```

1 #
2 # TODO: enter JYTHON code and save
3 #
4 print AdminConfig.list(
5     'JDBCProvider', AdminConfig.getid(
6         '/Cell:was85hostNode01Cell/'))
7 print AdminConfig.list(
8     'DataSource', AdminConfig.getid(
9         '/Cell:was85hostNode01Cell/Node:was85hostNode01/
10        Server:server1/JDBCProvider:Derby JDBC Provider/'))
11

```

<terminated> ex02_listJDBCResources.py [WebSphere Application Server Administrative Script] /opt/IBM/WebSphere
WASX7209I: Connected to process "server1" on node was85hostNode01 using SOAP connector; The type of process is: UnManagedProcess
"Derby JDBC Provider (XA)(cells/was85hostNode01Cell/nodes/was85hostNode01/servers/server1/resources.xml#builtin_jdbcprovider)"
"Derby JDBC Provider (XA)(cells/was85hostNode01Cell/resources.xml#builtin_jdbcprovider)"
"Derby JDBC Provider(cells/was85hostNode01Cell/nodes/was85hostNode01/servers/server1/resources.xml#JDBCProvider_1183122153343)"
"Default Datasource(cells/was85hostNode01Cell/nodes/was85hostNode01/servers/server1/resources.xml#DataSource_1183122153625)"

Figure 6-15. Running a script and viewing output in the Console

WA680 / VA6801.0

Notes:

The Console View displays various console types that depend on the type of development and the current set of user settings.

The Process Console is used to view the runtime messages from a server and shows three different kinds of text:

- Standard output
- Standard error
- Standard input

See the IADT documentation for a description of the Process Console commands.

Jython debugger

- The Jython debugger enables you to detect and diagnose errors in a Jython script.
- The debugger controls the execution of code by:
 - Suspending execution at preset breakpoints
 - Allowing you to step through code
- The debugger allows you to examine the contents of variables.

© Copyright IBM Corporation 2013

Figure 6-16. Jython debugger

WA680 / VA6801.0

Notes:

The Jython debugger enables you to detect and diagnose errors in Jython script that is used for WebSphere Application Server administration. With the debugger, you can control the execution of your code by setting line breakpoints, suspending execution, stepping through your code, and examining the contents of variables.

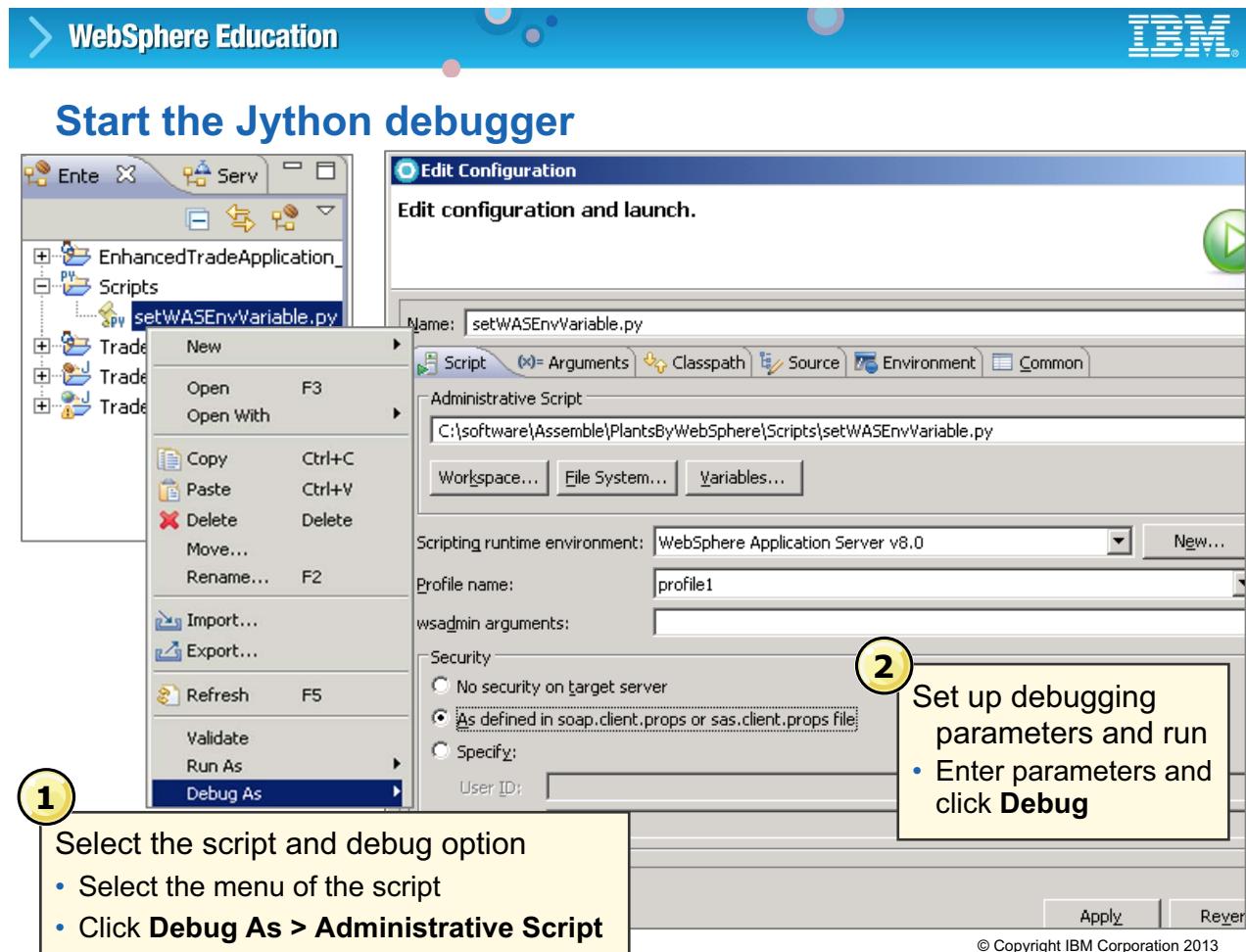


Figure 6-17. Start the Jython debugger

WA680 / VA6801.0

Notes:

A launch configuration is a mechanism for defining and saving different workbench configurations that can be started separately. You can use launch configurations for starting debug sessions.

You can create a launch configuration for starting a Jython script debug session or, when you choose a Navigator view Debug action for Jython script, a launch configuration is created for you.

WebSphere Education

IBM

Jython debugger perspective

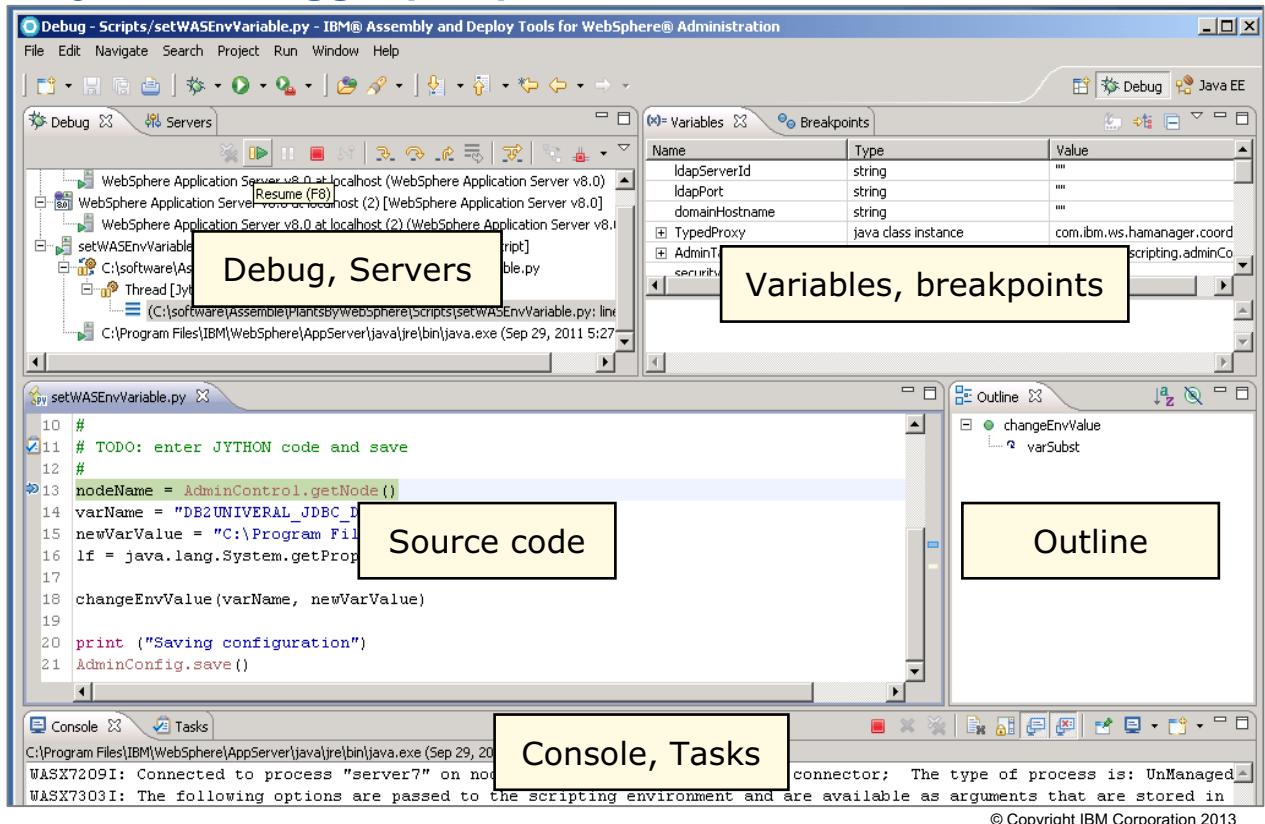


Figure 6-18. Jython debugger perspective

WA680 / VA6801.0

Notes:

The Jython debugger perspective provides many views to help you debug your Jython scripts. This screen capture of the debugger perspective is customized to highlight the commonly used debugger views.

Setting breakpoints

- Breakpoints are set in the Jython editor.
- Breakpoints are temporary markers that are placed in the code that tell the debugger to stop the script at a selected line.
- To set breakpoints:
 - Double-click the marker bar beside the code line
 - Or
 - Right-click in the marker bar beside the code line and select **Toggle Breakpoint**



© Copyright IBM Corporation 2013

Figure 6-19. Setting breakpoints

WA680 / VA6801.0

Notes:

When you are debugging Jython script, you can set line breakpoints. Breakpoints are saved upon termination of a debug session. When the workbench is running the script and encounters a breakpoint, the script temporarily stops running. Execution is suspended at the breakpoint before the script runs, at which point you can check the contents of variables. You can then step over (execute) and see what effect the statement has on the script. With the debugger, setting breakpoints is easily accomplished by using context menus in the editor. After they are set, you can disable breakpoints so they do not suspend execution and then, later, enable them again.

Stepping through breakpoints

- **Step Into**

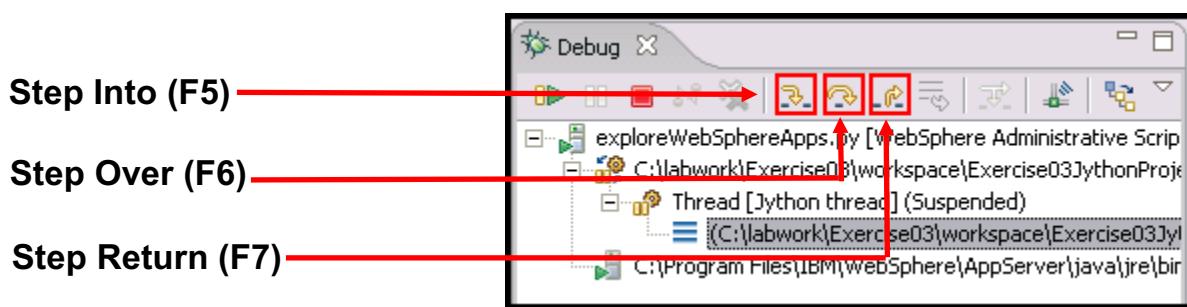
- The next expression on the selected line is run, and execution suspends at the next executable line in the method that is invoked.

- **Step Over**

- The currently selected line is run and suspends on the next executable line.

- **Step Return**

- Execution resumes until the next return statement in the current method is run, execution suspends on the next executable line.



© Copyright IBM Corporation 2013

Figure 6-20. Stepping through breakpoints

WA680 / VA6801.0

Notes:

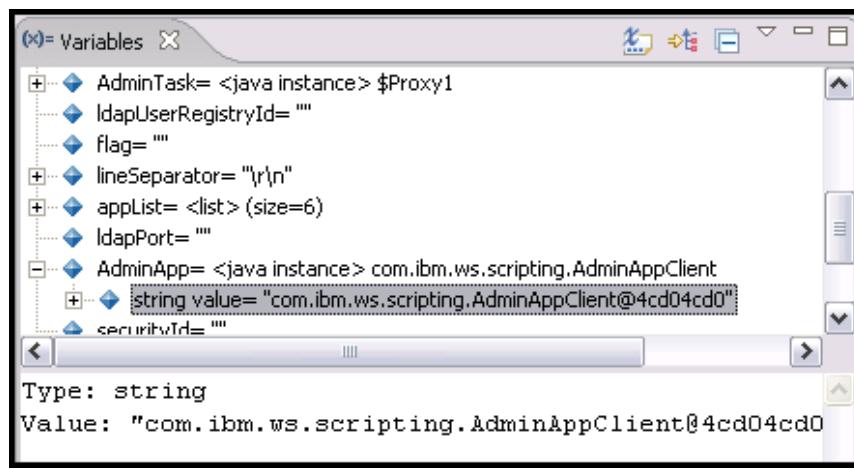
The following lists the shortcuts that you can use to control the Debug perspective through the keyboard:

- **CTRL-SHIFT-B:** Add/Remove breakpoint
- **CTRL-D:** Display
- **CTRL-Q:** Inspect
- **CTRL-R:** Run to Line
- **CTRL-U:** Run snippet
- **F5:** Step into
- **F6:** Step over
- **F7:** Run to return
- **F8:** Resume
- **F9:** Relaunch last

- **CTRL-F11:** Debug Last Launched Run Last Launched

Examining variables

- The Variables view displays variable values while debugging Jython scripts.
- The variable values cannot be edited.
- Complex variables can be expanded to show the elements that make up the variable.



© Copyright IBM Corporation 2013

Figure 6-21. Examining variables

WA680 / VA6801.0

Notes:

By default, the Variables view displays only user variables. Click the Variable view down arrow icon and select **Show All Jython Variables** to see all available variables.

Administrative console command assistance

- Used to view wsadmin scripting commands in the Jython language for the last action that is run in the administrative console.
- If a command assistance link is listed in the help portlet, **wsadmin** commands exist for the last console action that is completed, and command assistance is available for that action.
- If the help portlet does not have a command assistance link in it, no command assistance data is available for the last console action.

© Copyright IBM Corporation 2013

Figure 6-22. Administrative console command assistance

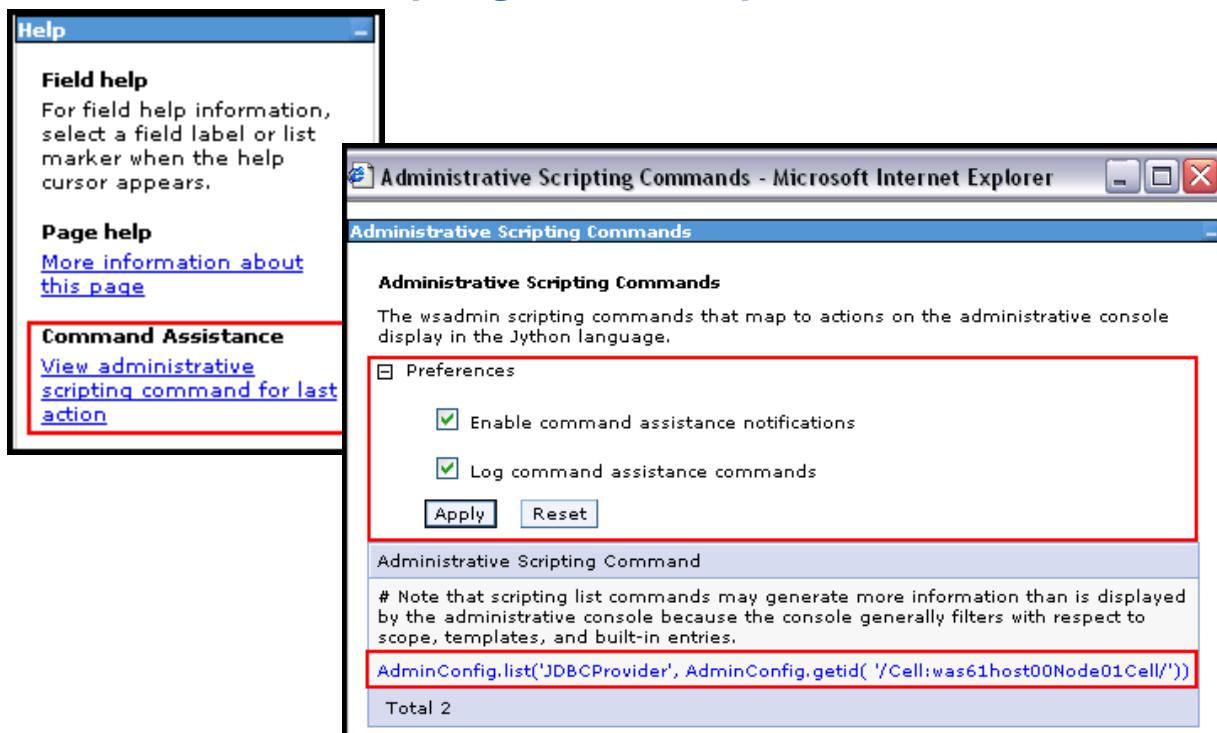
WA680 / VA6801.0

Notes:

Using command assistance, you can view wsadmin scripting commands in the Jython language for the last action that is run in the administrative console.

Use command assistance to see wsadmin scripting commands that correspond to actions in the administrative console. Seeing these commands might help you develop the commands necessary to administer WebSphere Application Server from the wsadmin utility.

Administrative scripting command preferences



© Copyright IBM Corporation 2013

Figure 6-23. Administrative scripting command preferences

WA680 / VA6801.0

Notes:

If the browser supports JavaScript, the Command assistance window automatically refreshes the command list to reflect the most recent console action. If the browser does not support Java scripts, click the link again under **Command assistance** in the help portal to refresh the command list.

Logs to <server>/logs directory in a file named
commandAssistanceJythonCommands_wsadmin.log



Command assistance notifications

- Enablement of the notifications allows integration with the IBM Assembly and Deploy Tools (IADT) Jython editor to help writing scripts.
- The command assistance emits Java Management Extensions (JMX) notifications.
- The commands can be inserted only into an open Jython editor.

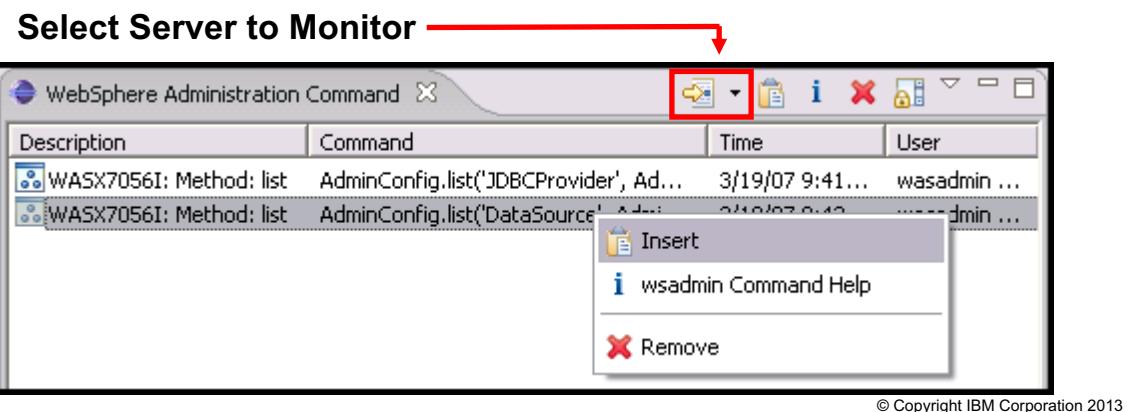


Figure 6-24. Command assistance notifications

WA680 / VA6801.0

Notes:

The WebSphere Administrative Command view in the IBM Assembly and Deploy Tools displays the notification events. These notifications are Jython commands.

To access the command notifications from the IBM Assembly and Deploy Tools :

1. Right-click the server process and select WebSphere administrative command assistance.
2. Click the *Select Server to Monitor* button and select the application server process to monitor.
3. Log in to the Integrated Solutions Console to generate the Jython notifications.

To insert the command notifications into a Jython script:

1. Open the Jython script editor into which you want to insert the command.
2. Right-click the command notification to insert
3. Click **Insert**.

Unit summary

Having completed this unit, you should be able to:

- Provide an overview of the IBM Assembly and Deploy Tools (IADT)
- Describe the IADT features that are used for Jython development

© Copyright IBM Corporation 2013

Figure 6-25. Unit summary

WA680 / VA6801.0

Notes:



Checkpoint questions

1. List two features the Jython editor provides to help build Jython scripts.
2. How can you set breakpoints in the Jython editor?
3. True or false: Command assistance notifications can be integrated into the IBM Assembly and Deploy Tools Jython editor.

© Copyright IBM Corporation 2013

Figure 6-26. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. List two features the Jython editor provides to help build Jython scripts.

Answer: Color syntax highlights, content assist, breakpoint assignment, and keyword help.

2. How can you set breakpoints in the Jython editor?

Answer: Right-click in the marker bar and select Toggle Breakpoint or double-click the marker bar beside the code line.

3. True or false: Command assistance notifications can be integrated into the IBM Assembly and Deploy Tools Jython editor.

Answer: True.

© Copyright IBM Corporation 2013

Figure 6-27. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 2



Using the IBM Assembly and Deploy Tools
(IADT) to develop Jython scripts

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 6-28. Exercise 2

WA680 / VA6801.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Create, debug, and run Jython scripts in the IADT
- Use the development tool aids provided by the IADT
- Explore other functions of the Administrative objects

© Copyright IBM Corporation 2013

Figure 6-29. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 7. Administrative object basics: Help and AdminConfig

What this unit is about

This unit describes the purpose and common usage of the Help and AdminConfig object.

What you should be able to do

After completing this unit, you should be able to:

- Describe how to use the Help object
- Describe the steps that are involved in using the AdminConfig object
- Identify the primary methods and resources that help in performing each step
- Explain key concepts that are related to managing configuration objects
- Explain how to use the AdminConfig object to query, create, modify, or delete a configuration object

Unit objectives

After completing this unit, you should be able to:

- Describe how to use the Help object
- Describe the steps that are involved in using the AdminConfig object
- Identify the primary methods and resources that help in performing each step
- Explain key concepts that are related to managing configuration objects
- Explain how to use the AdminConfig object to query, create, modify, or delete a configuration object

© Copyright IBM Corporation 2013

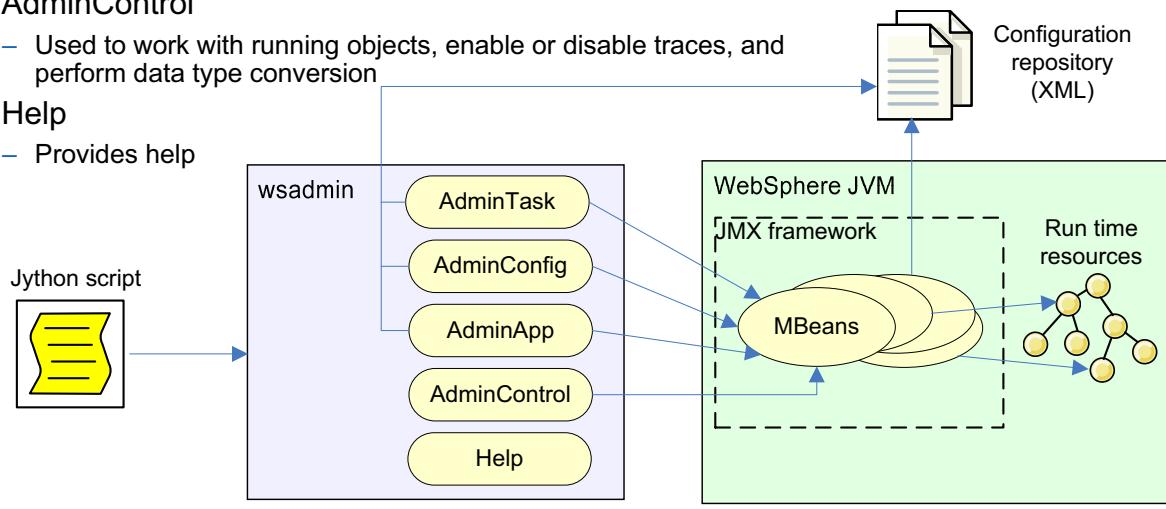
Figure 7-1. Unit objectives

WA680 / VA6801.0

Notes:

What are administrative objects?

- AdminTask
 - Provides task-oriented commands to conduct common configuration and operational changes
- AdminConfig
 - Use to create or change static configuration objects
- AdminApp
 - Used to install, modify, or administer applications
- AdminControl
 - Used to work with running objects, enable or disable traces, and perform data type conversion
- Help
 - Provides help



© Copyright IBM Corporation 2013

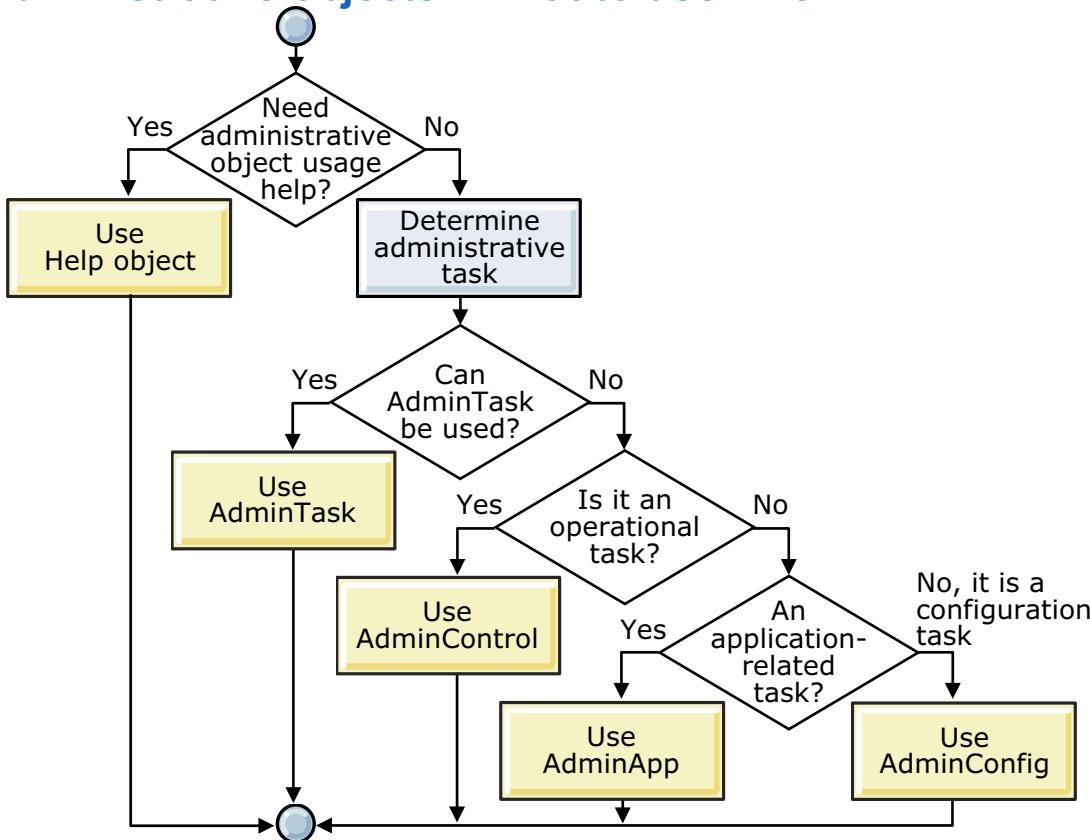
Figure 7-2. What are administrative objects?

WA680 / VA6801.0

Notes:

The administrative objects allow changes to the configuration and operation of a WebSphere Application Server environment. The AdminConfig, AdminTask, and AdminApp objects handle configuration modifications. You can make configuration changes without being connected to a server. The AdminControl object requires the server to be started because it can be started only on running JMX MBeans. The AdminControl administrative object is used to work with running objects within the JVM. The AdminConfig administrative object is used to create or change static configuration objects, which are stored in the configuration repository. The AdminApp administrative object is used to install, modify, and update applications. The AdminTask provides task-oriented commands to conduct common configurations and operational changes. The Help administrative object provides help on all of the administrative objects.

Administrative objects: What to use when



© Copyright IBM Corporation 2013

Figure 7-3. Administrative objects: What to use when

WA680 / VA6801.0

Notes:

This flowchart provides guidelines on using administrative objects, which are based on the type of administrative task. First check to see whether the AdminTask object can be used to perform the required task. Using the AdminTask object is the easiest and the most straight-forward way to affect the change.

When to use the Help object

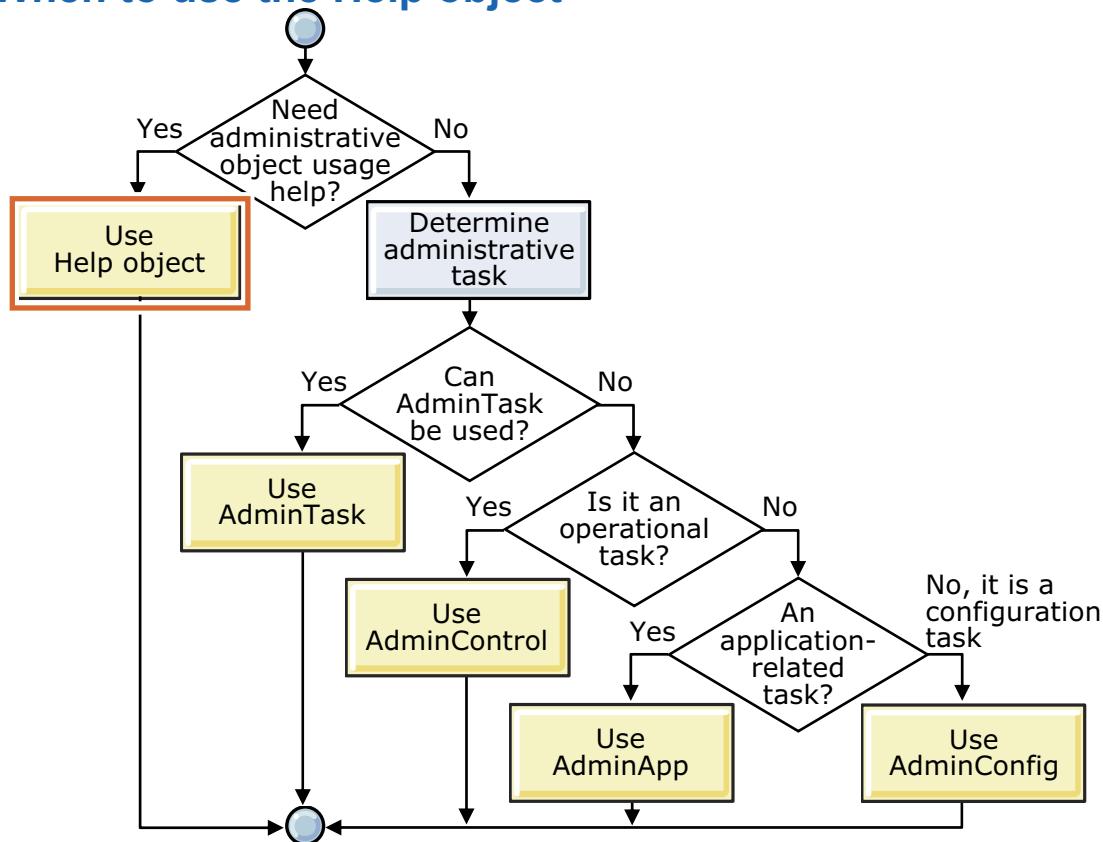


Figure 7-4. When to use the Help object

WA680 / VA6801.0

© Copyright IBM Corporation 2013

Notes:

The Help object

- Provides information about:
 - Available methods for the five administrative objects
 - Operations and attributes of running MBeans
 - WebSphere informational, warning, and error codes
 - wsadmin command-line options
- In Jython, use the `print` statement to improve the readability of the output from the Help object methods
 - Also applies to other administrative object methods that return a string

```
Help.help()
'WASX7028I: The Help object has two purposes: \n\n\tFirst, provide general help
information for the the objects\n\tsupplied by wsadmin for scripting: Help, Admi
nApp, AdminConfig,\n\tand AdminControl.\n\n\ ...
← Output without print statement
shows line break control characters

print 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. ...
← Output with print statement displays line breaks properly
```

© Copyright IBM Corporation 2013

Figure 7-5. The Help object

WA680 / VA6801.0

Notes:

You must use the `print` statement to properly format the output.

Methods for getting help on the administrative objects

- `AdminApp()`
 - Provides a description of the AdminApp object and lists its methods
- `AdminConfig()`
 - Provides a description of the AdminConfig object and lists its methods
- `AdminControl()`
 - Provides a description of the AdminControl object and lists its methods
- `AdminTask()`
 - Provides a description of the AdminTask object and lists its methods
- `help()`
 - Provides a description of the Help object and lists its methods

```
print Help.AdminConfig()
WASX7053I: The AdminConfig object communicates with the
Config Service in a WebSphere server to manipulate configuration data
for a WebSphere installation. AdminConfig has commands to list, create,
remove, display, and modify configuration data, as well as commands to
display information about configuration data types.

...
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
```

© Copyright IBM Corporation 2013

Figure 7-6. Methods for getting help on the administrative objects

WA680 / VA6801.0

Notes:

Each of the four wsadmin objects can be called as a method on the Help object.

Method for getting help on WebSphere messages

- `message (messageID)`
 - `messageID` is a string that represents the message code
 - Returns an explanation and user action for `messageID`
- Can be used to see the description of any WebSphere message, including those generated by wsadmin

```
print Help.message("WASX7115E")
Explanation: wsadmin failed to read an ear file when preparing to copy it to a
temporary location for AdminApp processing.
User action: Examine the wsadmin.traceout log file to determine the problem; this
problem can result from an incorrect file path or a file permission problem.

print Help.message("ADMU3000I")
Explanation: This is an informational message indicating that the server has
started.
User action: None

print Help.message("DSRA0010E")
Explanation: A database error occurred for which the SQL state and the Error code
are listed.
User action: Use the information associated with the error code to fix the problem
in the database.
```

© Copyright IBM Corporation 2013

Figure 7-7. Method for getting help on WebSphere messages

WA680 / VA6801.0

Notes:

The Help object can be used to obtain more information about error messages.

Method for getting help on starting wsadmin

```
print Help.wsadmin()
```

WASX7001I: wsadmin is the executable for WebSphere scripting.

Syntax:

```
wsadmin
  [ -h(elp) ]
  [ -? ]
  [ -c <command> ]
  [ -p <properties_file_name>]
  [ -profile <profile_script_name>]
  [ -f <script_file_name>]
  [ -javaoption java_option]
  [ -lang language]
  [ -wsadmin_classpath classpath]
  [ -profileName profile]
```

```
...
```

© Copyright IBM Corporation 2013

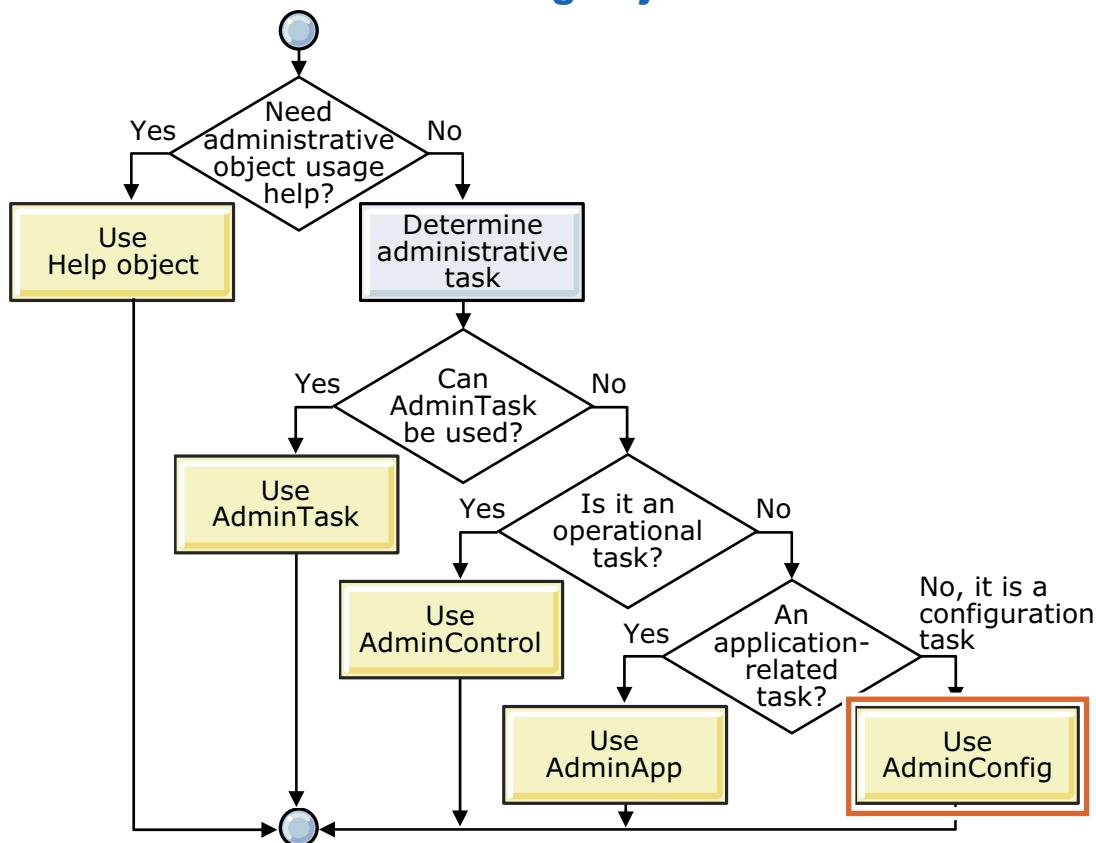
Figure 7-8. Method for getting help on starting wsadmin

WA680 / VA6801.0

Notes:

The Help object is used to obtain information about wsadmin.

When to use the AdminConfig object



© Copyright IBM Corporation 2013

Figure 7-9. When to use the AdminConfig object

WA680 / VA6801.0

Notes:

The AdminConfig object is used to make configuration changes that the AdminTask and AdminApp objects do not support.

The AdminConfig object

- Used to manage the configuration information that is stored in the repository
 - Query, create, modify, and delete configuration objects
- Methods can be started with or without a connection to a server
 - Can be used with wsadmin's *-conntype NONE* option
 - If connected to a server in a distributed server environment, methods are available only if wsadmin is connected to the deployment manager
- Using the AdminConfig object requires understanding the WebSphere configuration model
 - Configuration types and their attributes
 - Two ways to learn:
 - Looking at the WebSphere configuration model Javadoc documentation in <was_root>\web\configDocs
 - Examining the XML configuration files under the appropriate config directory

© Copyright IBM Corporation 2013

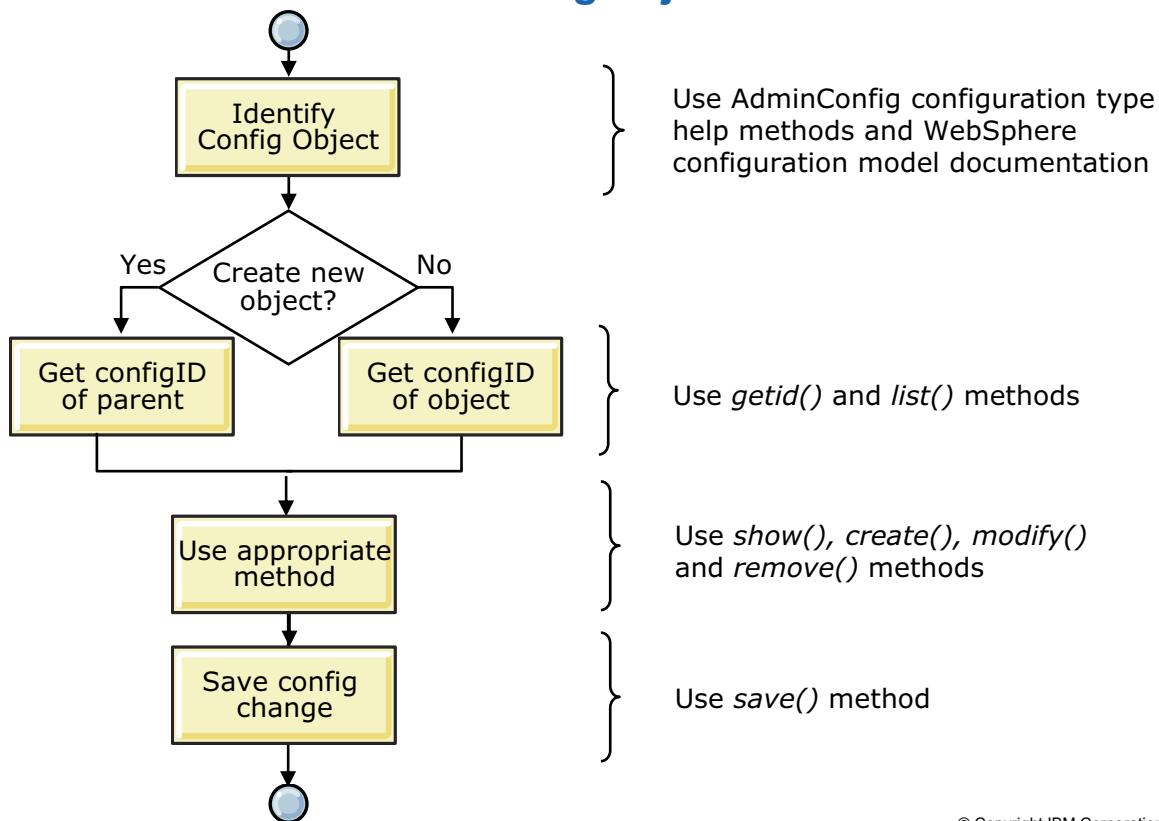
Figure 7-10. The AdminConfig object

WA680 / VA6801.0

Notes:

The AdminConfig administrative object is used to configure information that is stored in the configuration repository. The configuration repository is the config folder inside the profile. An administrator should not modify the files of the configuration repository. WebSphere Application Server provides the AdminConfig administrative object to encapsulate all of the files. In a distributed server environment, the AdminConfig methods are available only if wsadmin is connected to the deployment manager. These commands are available if connected to a node agent or a managed application server.

How to use the AdminConfig object



© Copyright IBM Corporation 2013

Figure 7-11. How to use the AdminConfig object

WA680 / VA6801.0

Notes:

This flowchart displays the general approach to use when making configuration changes with the AdminConfig object. The parts of this flowchart provide a summary of the different resources can be used to complete administrative tasks.

Identifying a configuration object's type

- Use the following AdminConfig configuration type help methods:
 - `types()`
 - Displays all of the available configuration object types
 - `attributes(configurationType)`
 - Shows the name and type of each attribute of `configurationType`
 - `required(configurationType)`
 - Shows the name and type of the required attributes of `configurationType`
 - `defaults(configurationType)`
 - Shows the name, type and default values for the attributes of `configurationType`
 - `parents(configurationType)`
 - Displays the configuration object types that contain `configurationType`
- Look at the WebSphere configuration model Javadoc documentation in `<was_root>\web\configDocs`
 - Open `index.html` to see the home page
 - View types by package or class name

© Copyright IBM Corporation 2013

Figure 7-12. Identifying a configuration object's type

WA680 / VA6801.0

Notes:

Examples: Using the configuration type help methods (1 of 4)

```
print AdminConfig.types()  
AccessPointGroup  
ActivationSpec  
ActivationSpecTemplateProps  
ActivitySessionService  
AdminObject  
...  
DataReplication  
DataReplicationDomain  
DataSource  
...
```

← Displays all available configuration types (partial output shown)

© Copyright IBM Corporation 2013

Figure 7-13. Examples: Using the configuration type help methods (1 of 4)

WA680 / VA6801.0

Notes:

Examples: Using the configuration type help methods (2 of 4)

```

print AdminConfig.attributes("DataSource")
authDataAlias String
authMechanismPreference ENUM(BASIC_PASSWORD, KERBEROS)
category String
connectionPool ConnectionPool
datasourceHelperClassname String
description String
diagnoseConnectionUsage boolean
jndiName String
logMissingTransactionContext boolean
manageCachedHandles boolean
mapping MappingModule
name String
preTestConfig ConnectionTest
propertySet J2EEResourcePropertySet
provider J2EEResourceProvider@ Shows the name and type of all the
providerType String
relationalResourceAdapter J2CResourceAdapter@ Reference type (indicated by
statementCacheSize int
xaRecoveryAuthAlias String) @ symbol)

```

© Copyright IBM Corporation 2013

Figure 7-14. Examples: Using the configuration type help methods (2 of 4)

WA680 / VA6801.0

Notes:

Examples: Using the configuration type help methods (3 of 4)

```
print AdminConfig.required("DataSource")  
Attribute          Type    ← Displays the attributes that are  
name             String   required for the "DataSource" type
```

```
print AdminConfig.parents("DataSource")  
JDBCProvider      ← Shows the configuration object  
                    types that contain a "DataSource"  
                    type
```

© Copyright IBM Corporation 2013

Figure 7-15. Examples: Using the configuration type help methods (3 of 4)

WA680 / VA6801.0

Notes:

Examples: Using the configuration type help methods (4 of 4)

```

print AdminConfig.defaults("DataSource")
Attribute          Type           Default
name               String
jndiName           String
description        String
category           String
providerType       String
authMechanismPreference ENUM
BASIC_PASSWORD
authDataAlias      String
manageCachedHandles boolean
logMissingTransactionContext boolean
xaRecoveryAuthAlias String
diagnoseConnectionUsage boolean
statementCacheSize int
datasourceHelperClassname String
provider           J2EEResourceProvider
propertySet         J2EEResourcePropertySet
connectionPool      ConnectionPool
preTestConfig       ConnectionTest
mapping             MappingModule
relationalResourceAdapter J2CResourceAdapter

```

Shows the default attribute values for the "DataSource" type

false
 true
 false
 10



© Copyright IBM Corporation 2013

Figure 7-16. Examples: Using the configuration type help methods (4 of 4)

WA680 / VA6801.0

Notes:

Attribute types

- Simple type
 - Includes string, integer, boolean, and enumeration
 - Example:
 - name String
 - authMechanismPreference ENUM(BASIC_PASSWORD, KERBEROS)
- Configuration object type
 - Represents a nested configuration object
 - Example: `outputStreamRedirect StreamRedirect`
- Generic configuration object type with a list of subtypes
 - Represents a nested configuration object with a type specified by the generic type or one of its listed subtypes
 - Nested object's type can be the generic type or one of its listed subtypes
 - Example: `processDefinition ProcessDef (JavaProcessDef)`
- Collection of objects of simple type, configuration object type, or generic configuration object type
 - Indicated by adding an asterisk (*) after `attributeType`
 - Example: `environment Property (TypedProperty, DescriptiveProperty) *`
- Reference type
 - Represents a reference to another configuration object
 - Indicated by adding the @ character after `attributeType`
 - Example: `provider J2EEResourceProvider@`

© Copyright IBM Corporation 2013

Figure 7-17. Attribute types

WA680 / VA6801.0

Notes:

Curly braces ({}) indicate optional items. Do not type them as part of the syntax. All wsadmin input and output consist of strings. These strings can represent objects of different types. The generic configuration object type that is shown, the `processDefinition` attribute can a value of type `ProcessDef` or `JavaProcessDef`.

The "@" symbol indicates that the attribute is a reference to another object. You cannot change a reference by using modify commands.

The "*" symbol, if present, indicates that the attribute contains a collection of objects of the specified type.



Using the WebSphere configuration model documentation

- Open <was_root>\web\configDocs\index.html to see the top-level navigation tree
- Browse types by package or class name

The screenshot shows a Microsoft Internet Explorer window displaying the 'WebSphere Application Model Documentation'. The address bar indicates the URL is 'C:\Program Files\IBM\WebSphere\AppServer\web\configDocs\index.html'. The left sidebar is titled 'Packages' and lists several Java packages: 'activitysessionjbext', 'activitysessionservice', 'activitysessionwebapext', 'adminservice', 'appcfg', 'app-deployment', 'application', 'applicationbnd', 'applicationclientext', 'applicationcontext', 'applicationserver', 'appmgt-service', 'appprofileapplicationclientext', 'appprofileapplicationcontext', 'appprofilejbext', 'appprofileservice', and 'acconfigfilewebapext'. The main content area is titled 'WebSphere Application And Configuration Model Documentation' and provides a summary of the models used by the WebSphere Application Server version 6.1, including J2EE Application Models, IBM J2EE Bindings and Extensions Models, and Application Server Configuration Models. It also describes the structure of the documentation and the definition of a model class.

© Copyright IBM Corporation 2013

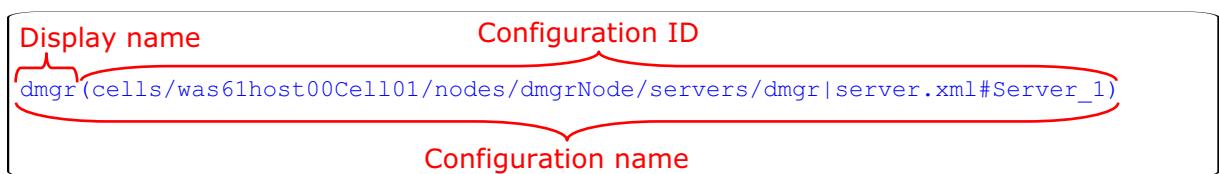
Figure 7-18. Using the WebSphere configuration model documentation

WA680 / VA6801.0

Notes:

Getting a configuration ID

- Configuration objects have a *configuration name*.
 - Consists of an optional *display name* followed by a *configuration ID*
 - For example, the configuration name of a deployment manager object:



- The *configuration ID* uniquely identifies configuration objects.
 - Since *display name* is optional, *configuration ID* is really the unique identifier
- You can obtain a *configuration name* with one of the following methods:
 - `getid(objectContainmentPath)`
 - `list(configurationType {}, scope{})`

© Copyright IBM Corporation 2013

Figure 7-19. Getting a configuration ID

WA680 / VA6801.0

Notes:

All configuration objects have a configuration name. The configuration name consists of an optional display name, followed by a configuration ID.

The getid method

- `getid(objectContainmentPath)`
 - Returns a string that contains the *configuration name* of the object that is specified by *objectContainmentPath*
 - Can return more than one configuration name if multiple matches are found
 - *objectContainmentPath* is a string that represents the hierarchical containment path of the configuration object
 - Syntax: “/type:name/”
 - *type* is the object’s configuration type
 - *name* is the object’s *display name* (optional)

```
Containment path
print AdminConfig.getid("/Server:trade1/")
Display name           Configuration ID
trade1(cells/was61host00Cell01/nodes/TradeNode01/servers/trade1|server.xml#Server_
1169660824562)
Configuration ID (continued)
                                         Configuration name
```

© Copyright IBM Corporation 2013

Figure 7-20. The getid method

WA680 / VA6801.0

Notes:

The `getid()` method returns the configuration name of an object by specifying its hierarchical containment path. Multiple names are returned if the containment path is not exact (for example, if the name portion is omitted) and more than one match is found.

Containment paths

- A *containment path* is a string with one or more `type:name` pairs
 - Each pair represents a configuration object in the containment hierarchy path of the target object
 - Syntax: `/type1:name1/type2:name2/type3:name3/ ...`
 - Pairs must reflect the correct hierarchical order
 - *name* can be omitted in the last pair
 - Syntax: `/type:/`
 - Represents all objects in the path of the *type*
 - For example:
 - `/Cell:cellName/Node:nodName/Server:serverName/` represents the path to the server object *serverName* in cell *cellName* and node *nodName*.
- A configuration object's containment path can be determined by examining the XML configuration files in the repository
 - Locate the configuration file that contains the definition of the configuration object
 - Find the XML element that defines the existing configuration object or one similar to it if it is a new object
 - Examine the nesting hierarchy of XML elements to discern the configuration object's containment hierarchy

© Copyright IBM Corporation 2013

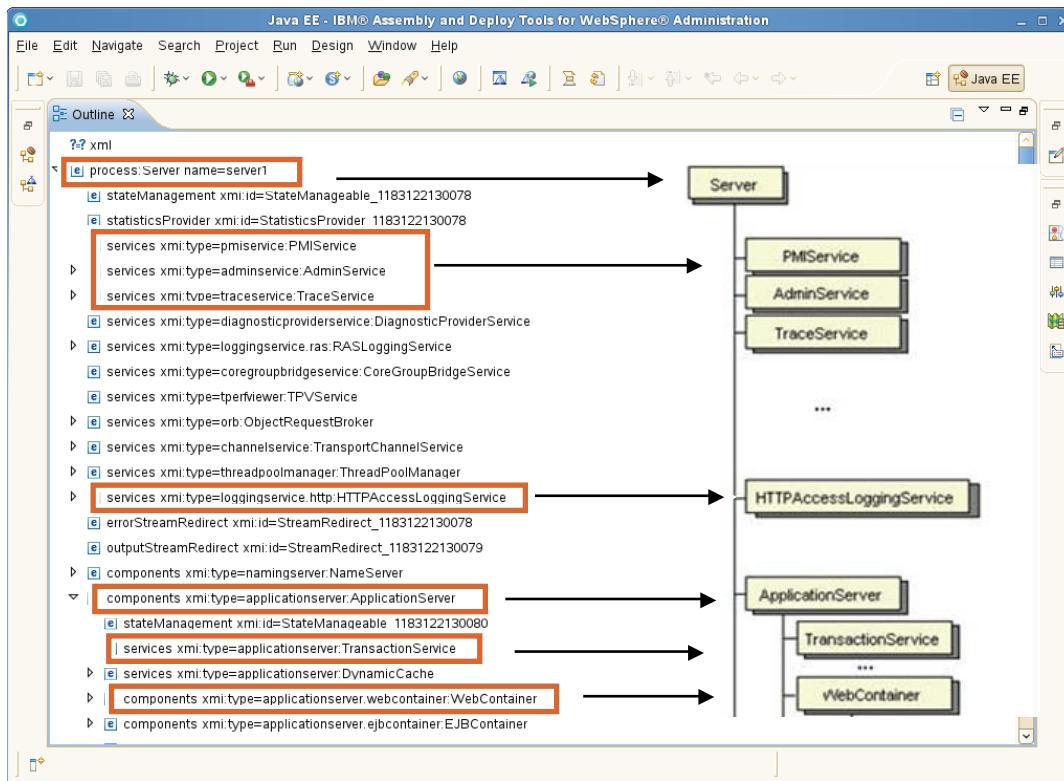
Figure 7-21. Containment paths

WA680 / VA6801.0

Notes:

The containment path is the full path of the object.

Example of containment relationships in the configuration files



© Copyright IBM Corporation 2013

Figure 7-22. Example of containment relationships in the configuration files

WA680 / VA6801.0

Notes:

Containment path and getid method examples

```

print AdminConfig.getId("/Server:/")           Displays the names of all configuration
dmgr(cells/was61host00Cell01/nodes/dmgrNode/servers/dmgr|server.xml#Server_1)
plants1(cells/wass61host00Cell01/nodes/TradeNode01/servers/tradel|server.xml#Server_1169660824562)   Displays the names of all configuration objects of type
...                                         "Server" in a specific cell and node
print AdminConfig.getId("/Cell:was61host00Cell01/Node:Node02/Server:/")
nodeagent(cells/was80host00Cell01/nodes/Node02/servers/nodeagent|server.xml
#Server_1169656421140)
plants2(cells/was80host00Cell01/nodes/Node02/servers/plants2|server.xml#Server_1169660828093)
print AdminConfig.getId("/Cell:was61host00Cell01/Node:Node02/Server:plants2/")
plants2(cells/was80host00Cell01/nodes/Node02/servers/plants2|server.xml#Server_1169660828093)

```

© Copyright IBM Corporation 2013

Figure 7-23. Containment path and getid method examples

WA680 / VA6801.0

Notes:

Containment paths that are seen here show the complete identity of an object. The containment path uniquely distinguishes objects that have the same name.

The list method

- `list(configurationType {, scope})`
 - Returns a string that contains the configuration names of all configuration objects of type `configurationType`
 - `scope` is the configuration ID of a cell, node, or server object
 - Narrows the search down to the specified scope

```

print AdminConfig.list("DataSource")      ← Displays the names of all configuration
"Quote                                objects of type "DataSource"
Datasource(cells/was61host00Cell01/applications/QuoteWS.ear/deployments/
QuoteWS|resources.xml#DataSource_1109615286909)"
DefaultEJBTimerDataSource(cells/was61host00Cell01/nodes/Node01/servers/pla-
nts1|resources.xml#DataSource_1000001)
PLANTS(cells/was61host00Cell01/clusters/PlantsCluster|resources.xml#DataSo-
urce_1169661420500)

Node1ID = AdminConfig.getId("/Node:Node01/")
print AdminConfig.list("DataSource", Node1ID)
DefaultEJBTimerDataSource(cells/was61host00Cell01/nodes/TradeNode01/server-
s/plants1|resources.xml#DataSource_1001)
    
```

↑ ← Retrieves the configuration ID
of the "TradeNode01" node

Displays the names of all configuration objects of type "DataSource" on node "Node01"

© Copyright IBM Corporation 2013

Figure 7-24. The list method

WA680 / VA6801.0

Notes:

The list command displays multiple objects of the same type.

Querying a configuration object

- `showAttribute(configurationID, attributeName)`
 - Displays the value of *attributeName* for the configuration object (*configurationID*)
- `show(configurationID {, attributeList })`
 - Shows the value of all of the attributes of the configuration object (*configurationID*), or the attributes that are specified in the optional *attributeList*
- `showall(configurationID {, attributeList })`
 - Recursively shows the value of all of the attributes of the configuration object (*configurationID*), or the attributes that are specified in the optional *attributeList*
 - Attributes of contained objects are also shown

```
plantsdsID = AdminConfig.getId("/DataSource:PLANTS/")

print AdminConfig.showAttribute(plantsdsID, "name")
PLANTS
```

© Copyright IBM Corporation 2013

Figure 7-25. Querying a configuration object

WA680 / VA6801.0

Notes:

The configuration ID is used to query objects for their attributes.

show method example

```

print AdminConfig.show(plantsdsID) ← Shows the value of all of the attributes of the
[authDataAlias dmgrNode/PlantsApp] "TRADE" datasource
[authMechanismPreference BASIC_PASSWORD]
[connectionPool ← Configuration ID of a nested
(cells/was80host00Cell01/clusters/PlantsCluster|resources.xml#Con
nectionPool_1169661420625)] configuration object
[datasourceHelperClassname com.ibm.websphere.rsaadapter.DB2UniversalDataStoreHelp
er]
[description "Plants application Datasource"]
[diagnoseConnectionUsage false]
[jndiName jdbc/tradeds]
[logMissingTransactionContext true]
[manageCachedHandles false]
[name PLANTS] ← Configuration name of a configuration
[propertySet object reference
(cells/was61host00Cell01/clusters/TradeCluster|resources.xml#J2EERe
sourcePropertySet_1169661420625)]
. . .

```

© Copyright IBM Corporation 2013

Figure 7-26. show method example

WA680 / VA6801.0

Notes:

The show method displays all of the attributes for an object. Nested objects are not shown in their entirety. Nested objects are shown with their object reference.

showall method example

```

print AdminConfig.showall(tradedsID) ← Recursively shows the value of all of
[authDataAlias dmgrNode/PlantsApp]      the attributes of the "PLANTS"
[authMechanismPreference BASIC_PASSWORD] datasource and its contained objects
[connectionPool "[[agedTimeout 0]
[connectionTimeout 180]
[freePoolDistributionTableSize 0]
[maxConnections 10]
[minConnections 1]
[numberOfFreePoolPartitions 0]
[numberOfSharedPoolPartitions 0]
[numberOfUnsharedPoolPartitions 0]
[properties []]
[purgePolicy EntirePool]
[reapTime 180]
[stuckThreshold 0]
[stuckTime 0]
[stuckTimerTime 0]
[surgeCreationInterval 0]
[surgeThreshold -1]
[testConnection false]
[testConnectionInterval 0]
[unusedTimeout 1800]]"]
. . .

```



Attribute values of nested connection pool object

© Copyright IBM Corporation 2013

Figure 7-27. showall method example

WA680 / VA6801.0

Notes:

The showAll method shows all objects, including nested objects. Nested objects are shown in their entirety.

Creating a configuration object

- Use either the `create` or `createUsingTemplate` method to create a configuration object.
- All create methods use a template to create an object, unless there are no templates that are defined for the object type.
 - `create` uses the *default template* for the object configuration type if available.
 - `createUsingTemplate` uses the template that is specified in one of its arguments.
- A *template* defines pre-set values for selected attributes of a configuration object type.
 - A *default template* contains the default values for selected attributes of the type.
 - Other templates can be defined for a type.
 - Some types allow you to create more templates.
 - Available templates are in the `<profile_root>\config\templates` directory.
 - Use the `listTemplates` method to determine the list of templates available for a configuration type.

© Copyright IBM Corporation 2013

Figure 7-28. Creating a configuration object

WA680 / VA6801.0

Notes:

Objects are created from templates. The `create` method creates an object from the default template. The `createUsingTemplate` method creates an object from a specified template.

The create method

- `create(type, parent, attributes {, parentAttributeName})`
 - Creates a configuration object.
 - `type` is a configuration object type.
 - `parent` is the configuration ID of the object's parent.
 - `attributes` is a list that contains a name-value pair for each object attribute to be initialized.
 - `parentAttributeName` is optional and specifies the name of the attribute in the parent object where the new object is attached.

```

parentID = AdminConfig.getId("/JDBCProvider:DB2 Universal
JDBC Driver Provider (XA)/") Attribute list

AdminConfig.create("DataSource", parentID, [{"name": "testDataSource"}])

'testDataSource(cells/was80host00Cell01/clusters/TradeCluster
|resources.xml#DataSource_1175302512703)' Configuration name of created datasource is returned

```

© Copyright IBM Corporation 2013

Figure 7-29. The create method

WA680 / VA6801.0

Notes:

The `createUsingTemplate` and `listTemplates` methods

- `createUsingTemplate(type, parent, attributes, template)`
 - Creates a configuration object of *type* with *parent*, with the *attributes* and *template* supplied.
 - *type* is a configuration object type.
 - *parent* is the configuration ID of the object's parent.
 - *attributes* is a list that contains a name-value pair for each object attribute to be initialized.
 - Specified attribute values override settings in the template.
 - *template* is the configuration ID of a template object, returned by using the `listTemplates` method.
- `listTemplates(type {, displayName})`
 - Returns a list of the *configuration names* of templates available for the *type*.
 - *displayName* is an optional filter string.
 - Causes the method to return a list of only the templates whose display name contains the supplied string.

© Copyright IBM Corporation 2013

Figure 7-30. The `createUsingTemplate` and `listTemplates` methods

WA680 / VA6801.0

Notes:

Templates can be found by using the `listTemplates` method.

Example: Creating a configuration object from a template

```

    ← Lists the configuration name of all "DataSource" templates
print AdminConfig.listTemplates("DataSource")
"Cloudscape JDBC Driver DataSource(templates/system|jdbc-resource-
provider-templates.xml#DataSource_DB2J_1)"

. . .
"DB2 Universal JDBC Driver DataSource(templates/system|jdbc-resource-
provider-templates.xml#DataSource_DB2_UNI_1)"

    ← Lists the configuration name of the "DataSource"
    ← template that matches the provided display name
. . .
print AdminConfig.listTemplates("DataSource", "DB2 Universal JDBC Driver
XA DataSource")
"DB2 Universal JDBC Driver XA DataSource(templates/system|jdbc-resource-
provider-
-templates.xml#DataSource_DB2_UNI_2)"

templateID = AdminConfig.listTemplates("DataSource", "DB2 Universal JDBC
Driver XA DataSource")

AdminConfig.createUsingTemplate("DataSource", parentID, [{"name",
"testDataSource2"}], templateID)
'testDataSource2(cells/was80host00Cell01/clusters/TradeCluster|resources.x
ml#DataSource_1175305945140)' ← Configuration name of created datasource is returned

```

© Copyright IBM Corporation 2013

Figure 7-31. Example: Creating a configuration object from a template

WA680 / VA6801.0

Notes:

Setting attribute values (1 of 4)

- Attribute values are specified in a list that contains name-value pairs:
 - [[attributeName1, attributeValue1], [attributeName2, attributeValue2], [attributeName3, attributeValue3], ...]
 - Name-value pairs are lists also
 - Nested lists are used to specify the value of nested objects
- Simple attribute type
 - Specify the value as a single literal
 - Examples:
 - String: ["name", "testDatasource"]
 - Integer: ["size", 100]
 - Boolean: ["enabled", "true"] or ["enabled", "false"]
 - Enumeration: ["authMechanismPreference", "BASIC_PASSWORD"]
 - where the literal is one of the valid values for the enumeration

© Copyright IBM Corporation 2013

Figure 7-32. Setting attribute values (1 of 4)

WA680 / VA6801.0

Notes:

Attribute values are specified in a list that contains name-value pairs. These named value pairs are also lists. There are simple attribute types that are Strings, Integers, Booleans, and enumerations. Configuration object types can specify the value as a new attribute list for the nested object.

Setting attribute values (2 of 4)

- Configuration object type
 - Specify the value as a new attribute list for the nested object
 - Example:
Attribute list of nested object
 - [“connPool”, [[connPoolAttr1, connPoolValue1], [connPoolAttr2, connPoolValue2], ...]]
 - Alternatively, you can:
 - Create the parent object first without specifying a value for the nested object attribute
 - Create the nested object separately specifying the configuration ID of the parent object in its *parent* argument

© Copyright IBM Corporation 2013

Figure 7-33. Setting attribute values (2 of 4)

WA680 / VA6801.0

Notes:

Setting attribute values (3 of 4)

- Generic configuration object type with a list of subtypes
 - Specify the value as a new attribute list for the nested object, and indicate the actual type in the attribute name
 - Example:


```
["pDef:JavaProcessDef", [[JProcDefAttr1, JProcDefValue1], [JProcDefAttr2, JProcDefValue2], ... ]]
```
- Collection of objects of simple type, configuration object type, or generic configuration object type
 - For a collection of strings (`String*`), specify the value as one string with each element separated by a semi-colon (`;`)
 - Example: 

```
["classpath", "c:/db2/db2.jar;c:/db2java/db2java.jar"]
```
 - For a collection of nested objects, specify the value as sequence of attribute lists, one for each object
 - Example:


```
["properties", [ [prop1Attr, prop1Value], [prop2Attr, prop2Value], ... ] ]
```

© Copyright IBM Corporation 2013

Figure 7-34. Setting attribute values (3 of 4)

WA680 / VA6801.0

Notes:

Setting attribute values (4 of 4)

- Reference type
 - Specify the value as a configuration name string
 - Example: **Configuration name of referenced object**
- As a good practice, you should assign literal values to variables and substitute the variable name in place of the literal.
 - Improves readability
 - Reduces the possibility of typographical errors and syntax errors
 - Example:
 - **connPoolAttributes** = [[connPoolAttr1, connPoolValue1], [connPoolAttr2, connPoolValue2], ...]
 - AdminConfig.create("DataSource", parentID, [..., ["connPool", **connPoolAttributes**], ...])

© Copyright IBM Corporation 2013

Figure 7-35. Setting attribute values (4 of 4)

WA680 / VA6801.0

Notes:

Modifying a configuration object (1 of 3)

- `modify(objectID, attributes)`
 - Changes specified attributes of the specified configuration object.
 - `objectID` is the configuration ID of the object to modify.
 - `attributes` is a list that contains a name-value pair for each object attribute to be modified.
- If the attribute to be modified is a collection that already has values, by default, the specified value is added (appended) to the collection.
 - To replace the entire collection, change it to an empty list first ([]), then append the replacement values.

Example:

```
print AdminConfig.attributes("JDBCProvider")
classpath String* ← "classpath" is a collection of strings
description String
implementationClassName String
name String ← "name" is a simple string
nativepath String*
propertySet J2EEResourcePropertySet
providerType String
xa boolean
```

© Copyright IBM Corporation 2010

Figure 7-36. Modifying a configuration object (1 of 3)

WA680 / VA6801.0

Notes:

Modifying a configuration object (2 of 3)

```
db2ProviderID = AdminConfig.getid("/JDBCProvider:DB2 Universal JDBC Driver Provider (XA) /")
                                         ↑ Displays the initial values of "classpath" and "name" attributes
print AdminConfig.show(db2ProviderID, ["name", "classpath"])

[classpath
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar;${DB2UNIVERSAL_JDBC_DRIVE
R_PATH}/db2jcc_license_cu.jar;${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_lic
nse_cisuz.jar]
[name "DB2 Universal JDBC Driver Provider (XA)"]

                                         ↑ Changes the "name" attribute
AdminConfig.modify(db2ProviderID, [[{"name", "My DB2 Universal driver"}]])
print AdminConfig.showAttribute(db2ProviderID, "name")
My DB2 Universal driver
```

© Copyright IBM Corporation 2013

Figure 7-37. Modifying a configuration object (2 of 3)

WA680 / VA6801.0

Notes:

Modifying a configuration object (3 of 3)

```

Method appends the value to the string collection
AdminConfig.modify(db2ProviderID, [{"classpath", "This should be
appended"}])
print AdminConfig.showAttribute(db2ProviderID, "classpath")
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar;${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar;${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_c
isuz.jar;This should be appended

Resets the collection to empty
AdminConfig.modify(db2ProviderID, [{"classpath", []}])
print AdminConfig.showAttribute(db2ProviderID, "classpath")
[]

Changes the "classpath" attribute again
AdminConfig.modify(db2ProviderID, [{"classpath", "Replacement string"}])
print AdminConfig.showAttribute(db2ProviderID, "classpath")
Replacement string ← "classpath" attribute value is now replaced

```

© Copyright IBM Corporation 2013

Figure 7-38. Modifying a configuration object (3 of 3)

WA680 / VA6801.0

Notes:

Deleting a configuration object

- `remove(configurationID)`
 - Removes the configuration object that is identified by *configurationID*
 - Removes the object definition from the configuration repository only
 - If a running instance of the configuration object exists when you remove its definition, the change has no effect on the running instance.

```
testDSID = AdminConfig.getId("/DataSource:testDataSource/")
AdminConfig.remove(testDSID)
''
```

Retrieves the configuration ID of "testDataSource"

Removes the "testDataSource" configuration object

© Copyright IBM Corporation 2013

Figure 7-39. Deleting a configuration object

WA680 / VA6801.0

Notes:

An administrative object is deleted with the `remove` method. The configuration ID is used as the key to delete the administrative object.

Saving or discarding configuration changes

- Configuration changes must be explicitly saved.
 - Changes are first kept in a local workspace.
 - Use the `save()` method to apply the changes to the master configuration repository.
 - Only exception is if changes were the result of using the single command execution option (`-c`) of `wsadmin`.
 - A save is automatically performed after the command is run.
- To discard changes that were made since the last save, use the `reset()` method.
 - Also, it is a good practice to issue an `AdminConfig.reset()` at the beginning of a script to start with a fresh workspace.

```
AdminConfig.reset()
''

AdminConfig.save()
'''
```

© Copyright IBM Corporation 2013

Figure 7-40. Saving or discarding configuration changes

WA680 / VA6801.0

Notes:

All changes are made to a temporary workspace. This workspace is unique for each user. You must use the `AdminConfig.save` method to save the changes in your workspace. The `reset` method can be used to undo all of the changes.

AdminConfig method reference (1 of 3)

Method name	Description
attributes	Shows the attributes for a specified type
checkin	Checks a file into the configuration repository
convertToCluster	Converts a server to be the first member of a new ServerCluster
create	Creates a configuration object, which is given a type, a parent, and a list of attributes, and optionally an attribute name for the new object
createClusterMember	Creates a new server that is a member of an existing cluster
createDocument	Creates a new document in the configuration repository
createUsingTemplate	Creates an object by using a particular template type
defaults	Displays the default values for attributes of a specified type
deleteDocument	Deletes a document from the configuration repository
existsDocument	Tests for the existence of a document in the configuration repository
extract	Extract a file from the configuration repository
getCrossDocumentValidationEnabled	Returns true if cross-document validation is enabled
getid	Shows the configuration ID of an object, which is given a string version of its containment

© Copyright IBM Corporation 2013

Figure 7-41. AdminConfig method reference (1 of 3)

WA680 / VA6801.0

Notes:

AdminConfig method reference (2 of 3)

Method name	Description
getObjectName	Returns a string version of the ObjectName of the running MBean that corresponds to a specified configuration ID
getSaveMode	Returns the mode that is used when "save" is invoked
getValidationLevel	Returns the validation that is used when files are extracted from the repository
getValidationSeverityResult	Returns the number of messages of a specified severity from the most recent validation
hasChanges	Returns true if unsaved configuration changes exist
help	Shows help information
installResourceAdapter	Installs a J2C resource adapter with the specified RAR file name and an option string in the node
list	Lists all configuration objects of a specified type
listTemplates	Lists all available configuration templates of a specified type
modify	Changes specified attributes of a specified configuration object
parents	Shows the objects that contain a specified type
queryChanges	Returns a list of unsaved files

© Copyright IBM Corporation 2013

Figure 7-42. AdminConfig method reference (2 of 3)

WA680 / VA6801.0

Notes:

AdminConfig method reference (3 of 3)

Method name	Description
remove	Removes the specified configuration object
required	Displays the required attributes of a specified type
reset	Discards unsaved configuration changes
save	Commits unsaved changes to the configuration repository
setCrossDocumentValidationEnabled	Sets the cross-document validation enabled mode
setSaveMode	Changes the mode that is used when "save" is invoked
setValidationLevel	Sets the validation that is used when files are extracted from the repository
show	Shows the attributes of a specified configuration object
showall	Recursively show the attributes of a specified configuration object, and all the objects that are contained within each attribute
showAttribute	Displays only the value for the single attribute specified
types	Shows all the supported configuration types
uninstallResourceAdapter	Uninstalls a J2C resource adapter with the specified resource adapter configuration ID
validate	Invokes validation

© Copyright IBM Corporation 2013

Figure 7-43. AdminConfig method reference (3 of 3)

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe how to use the Help object
- Describe the steps that are involved in using the AdminConfig object
- Identify the primary methods and resources that help in performing each step
- Explain key concepts that are related to managing configuration objects
- Explain how to use the AdminConfig object to query, create, modify, or delete a configuration object

© Copyright IBM Corporation 2013

Figure 7-44. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. Which administrative object can be used to get help information about all administrative objects?
2. True or false: Use the AdminConfig object when you must make a configuration change that is not supported by the AdminTask or AdminApp object.
3. Which AdminConfig method can be used to list the available configuration types?
4. Which AdminConfig method can be used to display a type's attributes?
5. Fill in the blank: Configuration objects are uniquely identified by their _____.

© Copyright IBM Corporation 2013

Figure 7-45. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.
- 5.

Checkpoint answers

1. Which administrative object can be used to get help information about all administrative objects?
Answer: Help
2. True or false: Use the AdminConfig object when you must make a configuration change that is not supported by the AdminTask or AdminApp object.
Answer: True
3. Which AdminConfig method can be used to list the available configuration types?
Answer: The `types()` method
4. Which AdminConfig method can be used to display a type's attributes?
Answer: The `attributes()` method
5. Fill in the blank: Configuration objects are uniquely identified by their configuration ID.

© Copyright IBM Corporation 2013

Figure 7-46. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 3



Using the Help and AdminConfig objects

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

WA680 / VA6801.0

Figure 7-47. Exercise 3

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Use the Help object to get help on the administrative objects
- Apply the general steps that are required to complete a configuration task by using the AdminConfig object
- Employ the primary methods and resources that help in performing each step
- Use the AdminConfig object to query, create, modify, and delete a configuration object

© Copyright IBM Corporation 2013

Figure 7-48. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 8. Administrative object basics: AdminApp

What this unit is about

This unit describes the purpose and common usage of the AdminApp object.

What you should be able to do

After completing this unit, you should be able to:

- Use wsadmin to manage WebSphere applications
- Describe the basic capabilities of the AdminApp object
- Explain when to use the AdminApp object



Unit objectives

After completing this unit, you should be able to:

- Use wsadmin to manage WebSphere applications
- Describe the basic capabilities of the AdminApp object
- Explain when to use the AdminApp object

© Copyright IBM Corporation 2013

Figure 8-1. Unit objectives

WA680 / VA6801.0

Notes:

When to use the AdminApp object

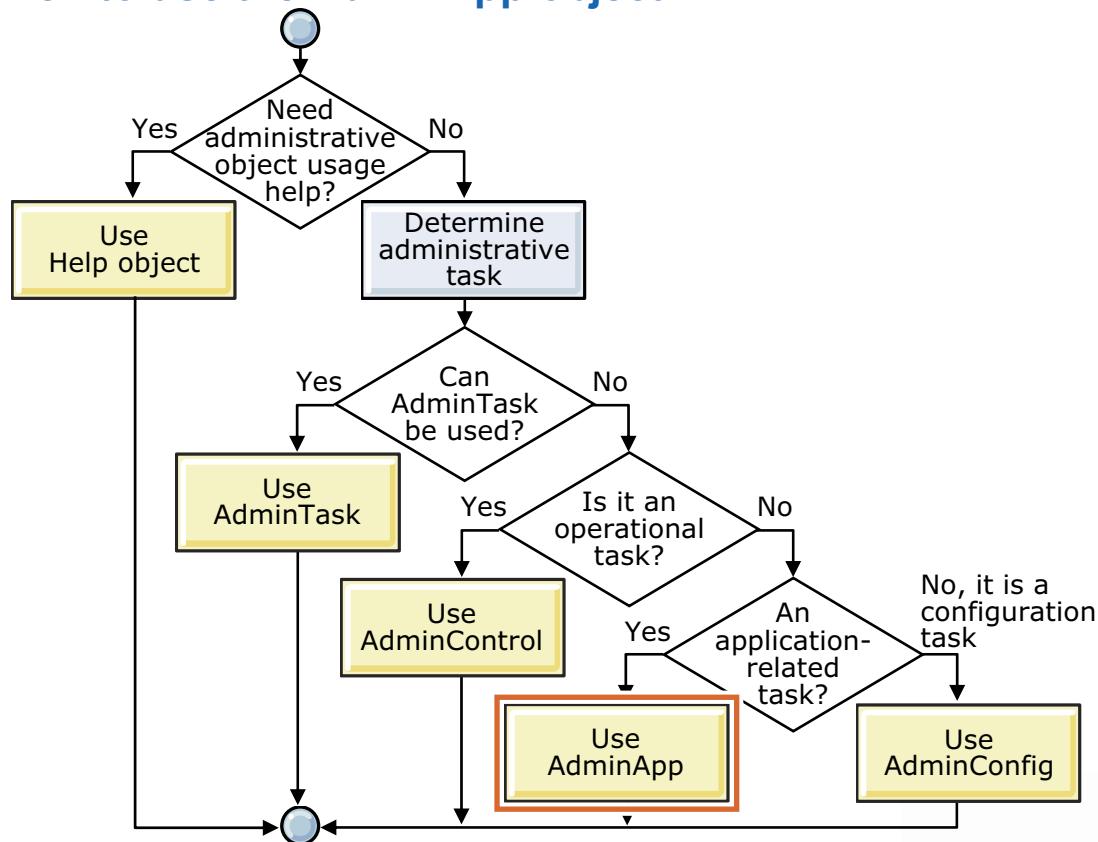


Figure 8-2. When to use the AdminApp object

WA680 / VA6801.0

© Copyright IBM Corporation 2013

Notes:



The AdminApp object

- Used to manage applications
 - Includes functions to:
 - Install and uninstall applications
 - List applications
 - Edit applications or modules
 - Modifications to an installed application are limited to updating application metadata, mapping virtual hosts to web modules, and mapping servers to modules
- Methods can be started with or without a connection to a server
 - Can be used with wsadmin's *-conntype NONE* option
 - If connected to a server in a distributed server environment, methods are available only if wsadmin is connected to the deployment manager
- Several methods have two forms that allow them to be run in two modes:
 - Batch mode
 - Supply all required method parameters on the command line and invoke method
 - Interactive mode
 - Supply all required parameters interactively as prompted by a text-based wizard and invoke method

© Copyright IBM Corporation 2013

Figure 8-3. The AdminApp object

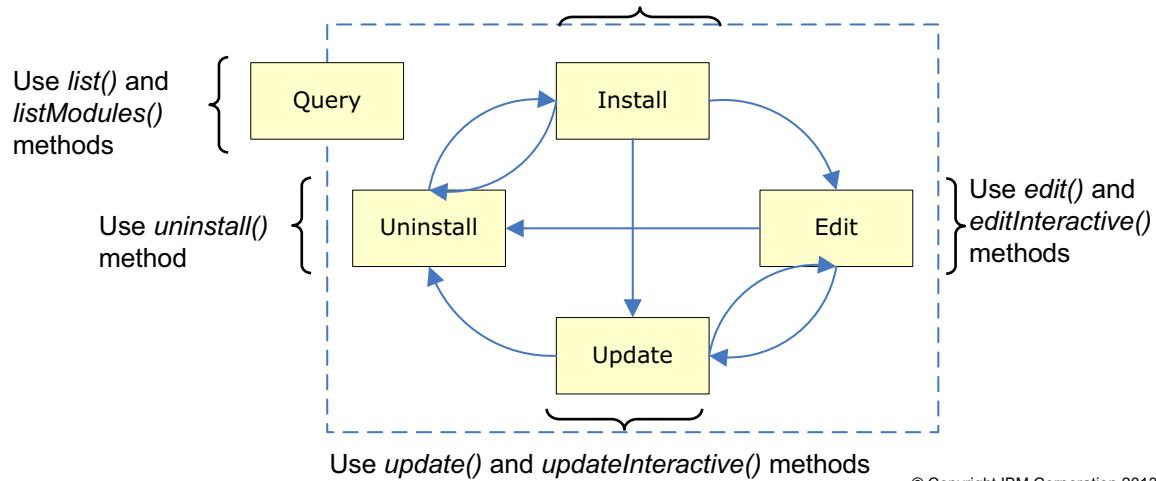
WA680 / VA6801.0

Notes:

Application management task support

- AdminApp supports the following management tasks that are performed during the application lifecycle:
 - Query the list of installed applications
 - Install an application
 - Edit an application's deployment properties
 - Update an entire application or its subcomponents
 - Uninstall an application

Use *install()* and *installInteractive()* methods



© Copyright IBM Corporation 2013

Figure 8-4. Application management task support

WA680 / VA6801.0

Notes:

AdminApp help methods

- `AdminApp.help()`
 - Provides a description of the AdminHelp object and lists its methods
 - Equivalent to `Help.AdminApp()`
- `AdminApp.help(aMethodName)`
 - Displays the description and usage information for *aMethodName*
- `AdminApp.help(anOptionName)`
 - Displays the description and usage information for *anOptionName*

```
print AdminApp.help("install") ← Help request for "install" method
WASX7096I: Method: install
```

Arguments: filename, options

Description: Installs the application in the file specified by "filename" using the options specified by "options". All required information must be supplied in the options string; no prompting is performed.

```
print AdminApp.help("deployejb") ← Help request for "deployejb" installation option
WASX7169I: "deployejb" option; specifies that EJBDeploy should be run
during install; the default is "nodeployejb"
```

© Copyright IBM Corporation 2013

Figure 8-5. AdminApp help methods

WA680 / VA6801.0

Notes:

Querying application and module names

- `list({scope})`
 - Returns a string that contains the display names of applications that are installed on the specified *scope*
 - If *scope* is omitted, all installed applications are listed
 - *scope* is an optional string that specifies the target scope
 - Syntax:
 - “`WebSphere:cell=cellName {, node=nodeName {, server=serverName} }`”
 - or
 - “`WebSphere:cell=cellName {, cluster=clusterName}`”
- `listModules(appName {, "-server"})`
 - Returns a string that contains the names of the modules that are contained in the application that is identified by *appName*
 - *appName* is the application's *display name*
 - If “-server” is specified, the returned module name includes information about the servers to which the module is mapped

© Copyright IBM Corporation 2013

Figure 8-6. Querying application and module names

WA680 / VA6801.0

Notes:

Example: Querying application and module names

```

print AdminApp.list() ← Lists all installed applications
PlantsByWebSphere

print AdminApp.list("WebSphere:cell=washost00Cell01") ← Lists installed
QuoteWS
PlantsByWebSphere
PlantByWebsphereProcessorApplication
isclite

print AdminApp.listModules("PlantsByWebSphere") ← Lists the modules in
PlantsByWebSphere#PlantByWebsphereEJB.jar+META-INF/ejb-jar.xml
PlantsByWebSphere#PlantByWebsphereWeb.war+WEB-INF/web.xml

Lists the modules in "PlantsByWebSphere" and the servers that they are mapped to
↓
print AdminApp.listModules("PlantsByWebSphere", "-server")
PlantsByWebSphere#PlantByWebsphereEJB.jar+META-INF/ejb-
jar.xml#WebSphere:cell=washost00Cel
101,cluster=PlantByWebsphereCluster
PlantsByWebSphere#RoguePlantByWebsphereAdmin.war+WEB-
INF/web.xml#WebSphere:cell=washost00C

" +" character separates multiple target mappings
  
```

© Copyright IBM Corporation 2013

Figure 8-7. Example: Querying application and module names

WA680 / VA6801.0

Notes:

Installing an application

- `install(fileName {, options})`
 - Installs the application that is identified by *fileName* using the options that are specified by *options* in batch mode
 - *fileName* is a string that contains the name of an archive file
 - Must have a *.ear*, *.jar* or *.war* extension
 - If the file is a *.jar* or *.war* file, it is automatically wrapped in a *.ear* file
 - *options* is an optional string or list that contains installation options
 - If omitted, default bindings are used to install the application
- `installInteractive(fileName)`
 - Installs the application that is identified by *fileName* in interactive mode
 - Options are retrieved interactively through a series of prompts
- Make sure to save the configuration after invoking the method
 - `AdminConfig.save()`

© Copyright IBM Corporation 2013

Figure 8-8. Installing an application

WA680 / VA6801.0

Notes:

Example: Installing an application in batch mode

```
AdminApp.install("C:\Software\EarsAndWars\MyIVT.ear",
["-node", "washost00Node01", "-server", "server1"] )
```

Options are provided on the method invocation line.
Here, they specify the target server

```
ADMA5016I: Installation of My_IVT_Application started.
ADMA5058I: Application and module versions are validated with versions
of deployment targets.
ADMA5005I: The application My_IVT_Application is configured in the
WebSphere Application Server repository.
. . .
SECJ0400I: Successfully updated the application My_IVT_Application with
the appContextIDForSecurity information.
ADMA5011I: The cleanup of the temp directory for application
My_IVT_Application
is complete.
ADMA5013I: Application My_IVT_Application installed successfully.
. . .

AdminConfig.save()
```

© Copyright IBM Corporation 2013

Figure 8-9. Example: Installing an application in batch mode

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (1 of 7)

```
AdminApp.installInteractive("C:\Software\EarsAndWars\MyIVT.ear")
```

Task[1]: Specifying application options ← Options are retrieved through a series of task prompts

Specify the various options that are available to prepare and install your application.

```
Precompile JavaServer Pages files: [No]: ←
Directory to install application: []:
Distribute application: [Yes]: ←
Use Binary Configuration: [No]:
Deploy enterprise beans: [No]: ← Press Enter to skip an optional parameter or to accept the default
Application name: [My_IVT_Application]: value
Create MBeans for resources: [Yes]:
Enable class reloading: [Yes]:
Allow dispatching includes to remote resources: [No]:
Allow servicing includes from remote resources: [No]:
```

Options are displayed along with their default values

© Copyright IBM Corporation 2013

Figure 8-10. Example: Installing an application in interactive mode (1 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (2 of 7)

Task[2]: Selecting servers

Specify targets such as application servers or clusters of application servers where you want to install the modules that are 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 serve as routers for requests to this application.

```
Module: My IVT EJB Module Display Name  
URI: MyIVTStatelessSession.jar,META-INF/ejb-jar.xml  
Server:  
[WebSphere:cell=washost00Node01Cell,node=washost00Node01,server=server1]:  
Module: My IVT Webapp Display Name  
URI: MyIVTWebApp.war,WEB-INF/web.xml  
Server:  
[WebSphere:cell=washost00Node01Cell,node=washost00Node01,server=server1]:
```

Press **Enter** to accept the default server mapping target for each module

© Copyright IBM Corporation 2013

Figure 8-11. Example: Installing an application in interactive mode (2 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (3 of 7)

Task[7]: Binding enterprise Bean to JNDI names

Notices skip from Task 2 to Task 7. Only the task options applicable to the application are prompted for.

Each non-message-driven enterprise bean in your application or module must be bound to a Java Naming and Directory Interface (JNDI) name.

EJB module: My IVT EJB Module Display Name

EJB: My_IVT_EJB_Name

URI: MyIVTStatelessSession.jar,META-INF/ejb-jar.xml

Target Resource JNDI Name: [ejb/My_IVT_Session_Bean_JNDI_Name]:

Task[9]: Mapping EJB references to enterprise beans

Each Enterprise JavaBeans (EJB) reference that is defined in your application must map to an enterprise bean.

Module: My_IVT_Webapp_Display_Name

EJB:

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

© Copyright IBM Corporation 2013

Figure 8-12. Example: Installing an application in interactive mode (3 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (4 of 7)

Task[16]: Selecting virtual hosts for Web modules

Specify the virtual host where you want to install the Web modules that are contained in your application. You can install Web modules on the same virtual host or disperse them among several hosts.

Web module: My_IVT_Webapp_Display_Name
URI: MyIVTWebApp.war,WEB-INF/web.xml
Virtual host: [default_host]:

Task[17]: Edit the Context root of web module

Context root defined in the deployment descriptor can be edited.

Web module: My_IVT_Webapp_Display_Name
URI: MyIVTWebApp.war,WEB-INF/web.xml
ContextRoot: [/MyIVT]:

Task[20]: Selecting method protections for unprotected methods for 1.x EJB

Specify whether you want to leave the method as unprotected or assign protection that denies all access.

© Copyright IBM Corporation 2013

Figure 8-13. Example: Installing an application in interactive mode (4 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (5 of 7)

Task[24]: Specifying EJB deploy options

Specify the options to deploy enterprise beans. Select database type only when all of the modules are mapped to the same database type. If some modules map to a different backend ID, set the database type blank so that the Select current backend ID panel is displayed.

....The EJB deploy option is not enabled.

Task[26]: Assign shared libraries to application or each module

Specify shared libraries that the application or individual modules reference. These libraries must be defined in the configuration at the appropriate scope.

Module: My_IVT_Application
URI: META-INF/application.xml
Shared Libraries: []:

© Copyright IBM Corporation 2013

Figure 8-14. Example: Installing an application in interactive mode (5 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (6 of 7)

Task[27]: Specifying JSP deploy options

Specify the options for JSP precompiler.

....The JSP deploy option is not enabled.

Task[28]: Edit the JSP reload attributes for web module.

Servlet and JSP 's reload attributes can be specified per module.

Web module: My_IVT_Webapp_Display_Name
URI: MyIVTWebApp.war,WEB-INF/ibm-web-ext.xmi
JSP enable class reloading: [Yes]:
JSP reload interval in seconds: [10]:

Task[30]: Copy WSDL files

Copy WSDL files

....This task does not require any user input

© Copyright IBM Corporation 2013

Figure 8-15. Example: Installing an application in interactive mode (6 of 7)

WA680 / VA6801.0

Notes:

Example: Installing an application in interactive mode (7 of 7)

Task[31]: Specify options to deploy Webservices

Specify options to deploy Webservices

....WebServices Deploy option is not enabled.

ADMA5016I: Installation of My_IVT_Application started.

ADMA5058I: Application and module versions are validated with versions of deployment targets.

. . .

ADMA5005I: The application My_IVT_Application is configured in the WebSphere Application Server repository.

SECJ0400I: Successfully updated the application My_IVT_Application with the appContextIDForSecurity information.

ADMA5011I: The cleanup of the temp directory for application My_IVT_Application is complete.

ADMA5013I: Application My_IVT_Application installed successfully.

..

AdminConfig.save() ← Remember to save

© Copyright IBM Corporation 2013

Figure 8-16. Example: Installing an application in interactive mode (7 of 7)

WA680 / VA6801.0

Notes:



AdminApp method options

- Options identify the targets of the method and specify what tasks it performs
 - Vary depending on the method and the contents of the application
- There are many options available for the *install*, *update*, and *edit* methods. To see the complete list, you can:
 - Use the `options()` method
 - Refer to the WebSphere Application Server Information Center
- Types of options:
 - Option with no value
 - Example:
 - `-nopreCompileJSPs`
 - Option with one value
 - Example:
 - `-node`, `myTestNode`
 - Option with a list of values, called a *task option*
 - Associated with a task performed during the execution of the method
 - Example:
 - `-MapWebModtoVH`, `[["MyWebApp", "myWeb.war", "WEB-INF/web.xml", newVH]]`

© Copyright IBM Corporation 2013

Figure 8-17. AdminApp method options

WA680 / VA6801.0

Notes:

Method options syntax

- Options can be specified in a string or list syntax:
 - No value and single value options
 - Option string syntax:
 - “[-noValueOption1 -noValueOption2 ... -optionName1 optionValue1 -optionName2 optionValue2 ...]”
 - Option list syntax:
 - [“-noValueOption1”, “-noValueOption2”, ... “-optionName1”, optionValue1, “-optionName2”, optionValue2 ...]
 - Task options
 - Option string syntax:
 - “[-taskName1 [[option1Value1 option1Value2 ...]] -taskName2 [[option2Value1 option2Value2 ...]]]”
 - Option list syntax:
 - [“-taskName1”, [[option1Value1, option1Value2, ...]], “-taskName2”, [[option2Value1, option2Value2, ...]]]
 - Order of option values matters and must match documentation

© Copyright IBM Corporation 2013

Figure 8-18. Method options syntax

WA680 / VA6801.0

Notes:

The options method

- `options()`
 - Displays the general options available for **installing** any application
- `options(fileName)`
 - Displays all the options available for **installing** the application that is specified by *fileName*
 - *fileName* is a string that contains the application's file name
- `options(fileName, operationName)`
 - Displays all the options available for the application that is specified by *fileName* for the operation (*operationName*)
 - *operationName* is one of:
 - "installapp," "updateapp," "addmodule," and "updatemodule"
- `options(applicationName)`
 - Displays all the options available for **editing** the application that is identified by *applicationName*
 - *applicationName* is the application's *display name*
- `options(moduleName)`
 - Displays all the options available for **editing** the application module that is identified by *moduleName*
 - *moduleName* is the application module's name as returned by the *listModules()* method

© Copyright IBM Corporation 2013

Figure 8-19. The options method

WA680 / VA6801.0

Notes:

options method examples

```

print AdminApp.options("C:/Software/EarsAndWars/PlantsByWebSphere.ear")
WASX7112I: The following options are valid for
"C:/Software/EarsAndWars/PlantsByWebSphere.ear"
MapModulesToServers
MapRolesToUsers
BindJndiForEJBNonMessageBinding
MapEJBRefToEJB
. . .
print AdminApp.options("PlantsByWebSphere")
WASX7112I: The following options are valid for "PlantsByWebSphere"
MapModulesToServers
MapRolesToUsers
BindJndiForEJBNonMessageBinding
MapEJBRefToEJB
. . .
print AdminApp.options("PlantsByWebSphere#PlantByWebsphereEJB.jar+META-
INF/ejb-jar.xml")
WASX7112I: The following options are valid for
"PlantsByWebSphere#PlantByWebsphereEJB.jar+META-INF/ejb-jar.xml"
MapModulesToServers
MapRolesToUsers
BindJndiForEJBNonMessageBinding
. . .

```

Options available for installing "PlantsByWebSphere"

Options available for editing "PlantsByWebSphere"

Options available for editing "PlantByWebsphereEJB" module in "PlantsByWebSphere"

© Copyright IBM Corporation 2013

Figure 8-20. options method examples

WA680 / VA6801.0

Notes:

Task options

- A task represents a step that is done as part of the execution of an AdminApp method.
 - Results in one or more entries or groups of entries (rows) being added to or updated in the application's configuration data
 - Example:
 - `AdminApp.install(myEarFile, ["-MapWebModToVH",
 ["First Web Module", "mod1.war", "WEB-INF/web.xml", "default_host"],
 ["Second Web Module", "mod2.war", "WEB-INF/web.xml", "default_host"]])`
 - Results in two row entries that are added to the application configuration data, one for each mapping target
- To determine how to specify a task option, use:
 - WebSphere Application Server Information Center
 - Provides syntax examples
 - `taskInfo(fileName, taskId)`
 - Provides information about the task option *taskId* for the application that is identified by *fileName*
 - Also runs default bindings for the application and displays them

© Copyright IBM Corporation 2013

Figure 8-21. Task options

WA680 / VA6801.0

Notes:

taskInfo method example

```
print AdminApp.taskInfo("C:/Software/EarsAndWars/PlantsByWebSphere.ear",
"MapWebModToVH")
```

MapWebModToVH: Selecting virtual hosts for Web modules

Specify the virtual host where you want to install the Web modules that are contained in your application. You can install Web modules on the same virtual host or disperse them among several hosts.

WASX7348I: ADMINAPP_TASKINFO=WASX7348I: Each element of the MapWebModToVH task consists of the following 3 fields: "Web module", "URI", "Virtual host".

Of these fields, the following are required and used as keys to locate the rows

in the task: "Web module" "URI" and the following may be assigned new values: "Virtual host"

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

Web module: PlantByWebsphereWeb URI: PlantByWebsphereWeb.war,WEB-INF/web.xml Virtual host: default_host	} Default bindings that are used unless overridden
---	--

© Copyright IBM Corporation 2013

Figure 8-22. taskInfo method example

WA680 / VA6801.0

Notes:

Useful task option features: Pattern matching

- Use an asterisk (*) or a regular expression pattern to specify a wildcard value for a required option
 - Applies only to options that cannot be edited (key value options)
 - Simplifies the task of supplying required values
 - Eliminates redundant data entry
- Example:
 - To map all of the web modules in an application to “default_host”, use:

```
AdminApp.install(myEarFile, ["-MapWebModToVH",
    [ [".*", ".*", "default_host"] ] )
```

Instead of:

```
AdminApp.install(myEarFile, ["-MapWebModToVH",
    [ ["First Web Module", "mod1.war,WEB-INF/web.xml", "default_host"],
        ["Second Web Module", "mod2.war,WEB-INF/web.xml "default_host"] ] )
```

© Copyright IBM Corporation 2013

Figure 8-23. Useful task option features: Pattern matching

WA680 / VA6801.0

Notes:

Useful task option features: Target manipulation

- Use a leading "+" or "-" to add or remove server targets
 - Allows multiple targets to be specified in one method invocation
 - Example:
 - AdminApp.install(myEarFile, ["-MapModulesToServers",
["Test App", "testApp.war", WEB-INF/web.xml",
"WebSphere:cell=myCell,node=myNode,server=server1
+WebSphere:cell=myCell,node=yourNode,server=server1"]])
 - Maps *testApp.war* to both *server1* on *myNode* and *server1* on *yourNode*
 - Allows targets to be added or removed individually
 - Example:
 - AdminApp.edit("Test App", ["-MapModulesToServers",
[".*", ".*",
"-WebSphere:cell=myCell,node=myNode,server=server1"]])
 - Removes the mapping of *Test App* to *server1* on *myNode*

© Copyright IBM Corporation 2013

Figure 8-24. Useful task option features: Target manipulation

WA680 / VA6801.0

Notes:



Editing an application

- `edit(appOrModName, options)`
 - Changes the deployment properties of the application or module that is identified by `appOrModName` with `options`, in batch mode
 - `appOrModName` is a string that contains one of the following items:
 - Application's *display name*
 - Module's *module name* as returned by the `listModules()` method
 - `options` is a string or list that contains the options to be changed
- `editInteractive(appOrModName)`
 - Changes the deployment properties of the application or module that is identified by `appOrModName` in interactive mode
 - Options are retrieved interactively through a series of prompts
- Make sure to save the configuration after starting the method
 - `AdminConfig.save()`

© Copyright IBM Corporation 2013

Figure 8-25. Editing an application

WA680 / VA6801.0

Notes:

Editing an application in batch mode

```

Turns off the automatic distribution of binaries for "My_IVT_Application"
AdminApp.edit("My_IVT_Application", ["-nodistributeApp"])
""

AdminConfig.save()
""

AdminConfig.save() Application display name
""

AdminApp.edit("My_IVT_Application", ["-MapWebModToVH",
[["My_IVT_Webapp_Display_Name", "MyIVTWebApp.war,WEB-INF/web.xml",
"proxy_host"]]] ) Replaces the virtual host mapping target for
module "My_IVT_Webapp_Display_Name" to
"proxy_host"

AdminConfig.save()
""

AdminApp.listModules("My_IVT_Application")
'My_IVT_Application#MyIVTStatelessSession.jar+META-INF/ejb-
jar.xml\r\nMy_IVT_App
lication#MyIVTWebApp.war+WEB-INF/web.xml'

AdminConfig.save() Remember to save
""

```

© Copyright IBM Corporation 2013

Figure 8-26. Editing an application in batch mode

WA680 / VA6801.0

Notes:

Updating an application

- `update(appName, contentType, options)`
 - Updates the application that is specified by `appName` using the update type that is specified by `contentType` with the options specified by `options`, in batch mode
 - Supports the addition, removal, and update of application subcomponents or the entire application.
 - `appName` is a string that contains the application's *display name*
 - `contentType` is a string that indicates the type of update. Value values are:
 - “app” updates the entire application
 - “file” updates a single file
 - “modulefile” updates a module
 - “partialapp” updates parts of the application
 - `options` is a string or a list that contains the options to be used for the update
 - All options for the `install()` method are valid
 - Morel options can be used to specify an operation or the file to be used in the update
- `updateInteractive(appName, contentType, options)`
 - Same as the previous method, except that for a `contentType` of “app” or “file”, the user is prompted interactively for information about each relevant task
- Make sure to save the configuration after starting the method
 - `AdminConfig.save()`

© Copyright IBM Corporation 2013

Figure 8-27. Updating an application

WA680 / VA6801.0

Notes:

Examples: Updating an application in batch mode

```

Single file update          "Add" operation
AdminApp.update("My_IVT_Application", "file", ["-operation", "add",
"-contents", "C:/Updates/web.xml", "-contenturi", "META-INF/web.xml"])
                                         ^                         ^
                                         |                         |
                                         Location of source update file  Location of application target file

Update of My_IVT_Application has started.
ADMA5009I: An application archive is extracted at C:\Program
Files\IBM\WebSphere
\AppServer\profiles\SamplesProfile\wstemp\wstemp\app_111ce7ac8e5\ext
ADMA5064I: FileMergeTask completed successfully for My_IVT_Application.
ADMA5005I: The application My_IVT_Application is configured in the
WebSphere Application Server repository.

ADMA5011I: The cleanup of the temp directory for application
My_IVT_Application is complete.
Update of My_IVT_Application has ended.
..
AdminConfig.save()           ← Remember to save
..

```

© Copyright IBM Corporation 2013

Figure 8-28. Examples: Updating an application in batch mode

WA680 / VA6801.0

Notes:

Uninstalling an application

- `uninstall(appName)`
 - Uninstalls the application that is specified by `appName`
 - `appName` is a string that contains the application's *display name*

```
AdminApp.uninstall("My_IVT_Application") ←Uninstalls the "My_IVT_Application"  
ADMA5017I: Uninstallation of My_IVT_Application started.  
ADMA5104I: The server index entry for  
WebSphere:cell=washost00Node01Cell,node=  
washost00Node01 is updated successfully.  
ADMA5102I: The configuration data for My_IVT_Application from the  
configuration  
repository is deleted successfully.  
ADMA5011I: The cleanup of the temp directory for application  
My_IVT_Application  
is complete.  
ADMA5106I: Application My_IVT_Application uninstalled successfully.  
''  
AdminConfig.save() ←Remember to save  
'''
```

© Copyright IBM Corporation 2013

Figure 8-29. Uninstalling an application

WA680 / VA6801.0

Notes:

AdminApp method reference (1 of 2)

Method name	Description
deleteUserAndGroupEntries	Deletes users or groups for all roles, and deletes user IDs and passwords for all RunAs roles that are defined in the application
edit	Edits the properties of an application or module
editInteractive	Edits the properties of an application or module in interactive mode
Export	Exports application to a file
exportDDL	Exports Data Definition Language (DDL) from application to a directory
help	Shows general help or help for an AdminApp method or option
install	Installs an application, using the file name and the option string that are supplied
installInteractive	Installs an application in interactive mode, using the file name and the option string that are supplied
isAppReady	Checks whether the application is ready to be run
list	Lists all installed applications

© Copyright IBM Corporation 2013

Figure 8-30. AdminApp method reference (1 of 2)

WA680 / VA6801.0

Notes:

AdminApp method reference (2 of 2)

Method name	Description
listModules	Lists the modules in a specified application
options	Shows the options available, for a file, application, or in general
publishWSDL	Publishes Web Service Description Language (WSDL) files for an application to a file
searchJNDIReferences	Lists applications that refer to the JNDIName on a node
taskInfo	Provides information about a particular task option for an application file
uninstall	Uninstalls an application,
update	Updates an installed application
updateAccessIDs	Updates the access ID information for users and groups that are assigned to various roles defined in an application
updateInteractive	Updates an installed application in interactive mode
view	Displays the task options and current bindings for an installed application or module,

© Copyright IBM Corporation 2013

Figure 8-31. AdminApp method reference (2 of 2)

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Use wsadmin to manage WebSphere applications
- Describe the basic capabilities of the AdminApp object
- Explain when to use the AdminApp object

© Copyright IBM Corporation 2013

Figure 8-32. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. Name the five types of application management tasks that can be done with the AdminApp object.
2. Which AdminApp method can be used to display the names of all of the applications that are installed in an environment?
3. True or false: AdminApp method options vary depending on the type of method that is called and the contents of the affected application.
4. Which AdminApp method allows you to determine the valid options available for installing a particular application?
5. Which AdminApp method allows us to determine the valid options available for a specific method task on an application?

© Copyright IBM Corporation 2013

Figure 8-33. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.
- 5.

Checkpoint answers

1. Name the five types of application management tasks that can be done with the AdminApp object.

Answer: Query, install, edit, update, and uninstall

2. Which AdminApp method can be used to display the names of all of the applications that are installed in an environment?

Answer: The `list()` method

3. True or false: AdminApp method options vary depending on the type of method that is called and the contents of the affected application.

Answer: True

4. Which AdminApp method allows you to determine the valid options available for installing a particular application?

Answer: The `options()` method

5. Which AdminApp method allows to determine the valid options available for a specific method task on an application?

Answer: The `taskInfo()` method

© Copyright IBM Corporation 2013

Figure 8-34. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 4

Using the AdminApp object

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 8-35. Exercise 4

WA680 / VA6801.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Access online help about how to use the AdminApp object
- Employ the primary methods and resources that are required to use the AdminApp object effectively
- Use the AdminApp object to query, install, edit, update, and uninstall an application

© Copyright IBM Corporation 2013

Figure 8-36. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 9. Administrative object basics: AdminControl

What this unit is about

This unit describes the purpose and common usage of the AdminControl object.

What you should be able to do

After completing this unit, you should be able to:

- Describe the steps that are involved in using the AdminControl object
- Identify the primary methods and resources that help in performing each step
- Explain the key concepts and syntactical rules that are required to use the AdminControl object
- Explain how to use the AdminControl object to modify MBean attributes and perform MBean operations

Unit objectives

After completing this unit, you should be able to:

- Describe the steps that are involved in using the AdminControl object
- Identify the primary methods and resources that help in performing each step
- Explain the key concepts and syntactical rules that are required to use the AdminControl object
- Explain how to use the AdminControl object to modify MBean attributes and perform MBean operations

© Copyright IBM Corporation 2013

Figure 9-1. Unit objectives

WA680 / VA6801.0

Notes:

When to use the AdminControl object

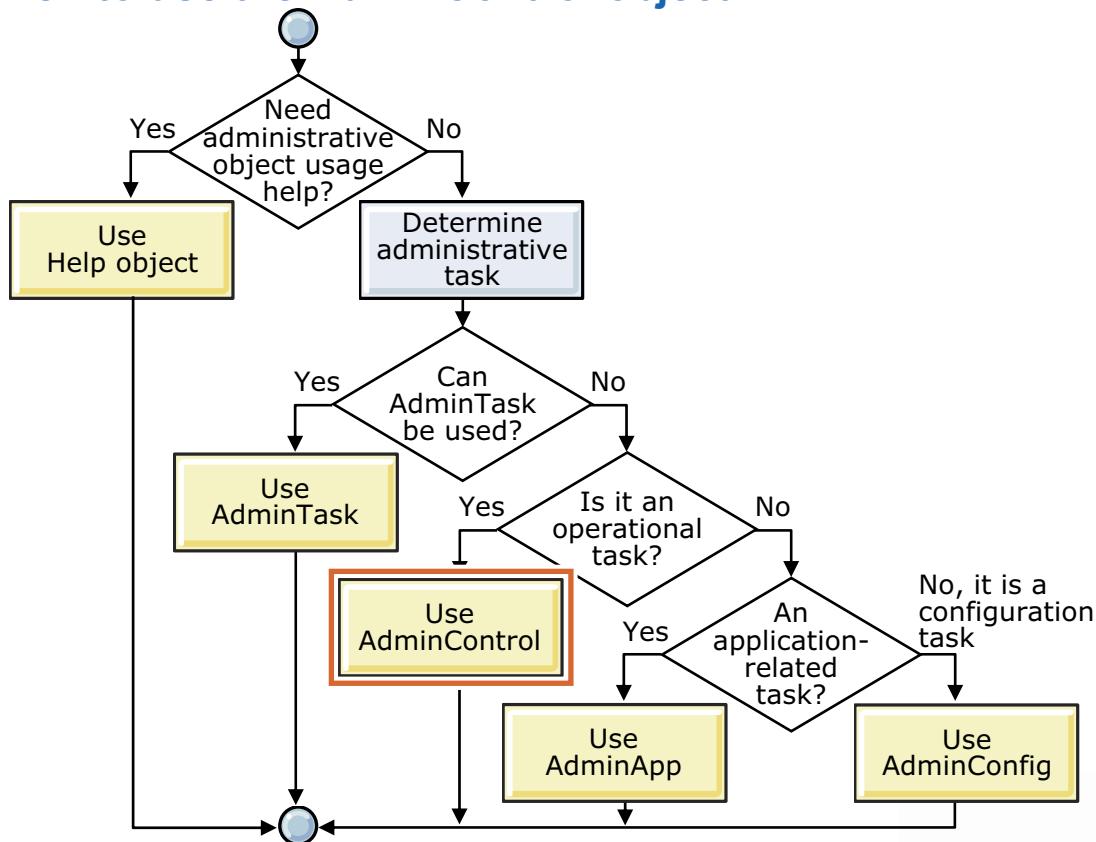


Figure 9-2. When to use the AdminControl object

WA680 / VA6801.0

© Copyright IBM Corporation 2013

Notes:

The AdminControl object (1 of 2)

- Used for operational control of the WebSphere Application Server environment
 - Query running MBean attributes
 - Start MBean operations
 - Allows you to perform functions such as:
 - Query server information
 - Control server state (start, stop) and tracing
- Methods require a connection to a running server
 - Require communication with live JMX MBeans
 - Number and type of MBeans available depends on the type of server process to which wsadmin is connected
- Many methods have two sets of signatures:
 - “Standard” signature: Uses a string representation of an *ObjectName* to identify the target MBean
 - “JMX” signature: Uses an instance of the JMX class *ObjectName* to identify the target MBean
- Use of the AdminControl object requires knowledge of the WebSphere MBean types
 - Documentation that is found in <was_root>\web\mbeanDocs directory

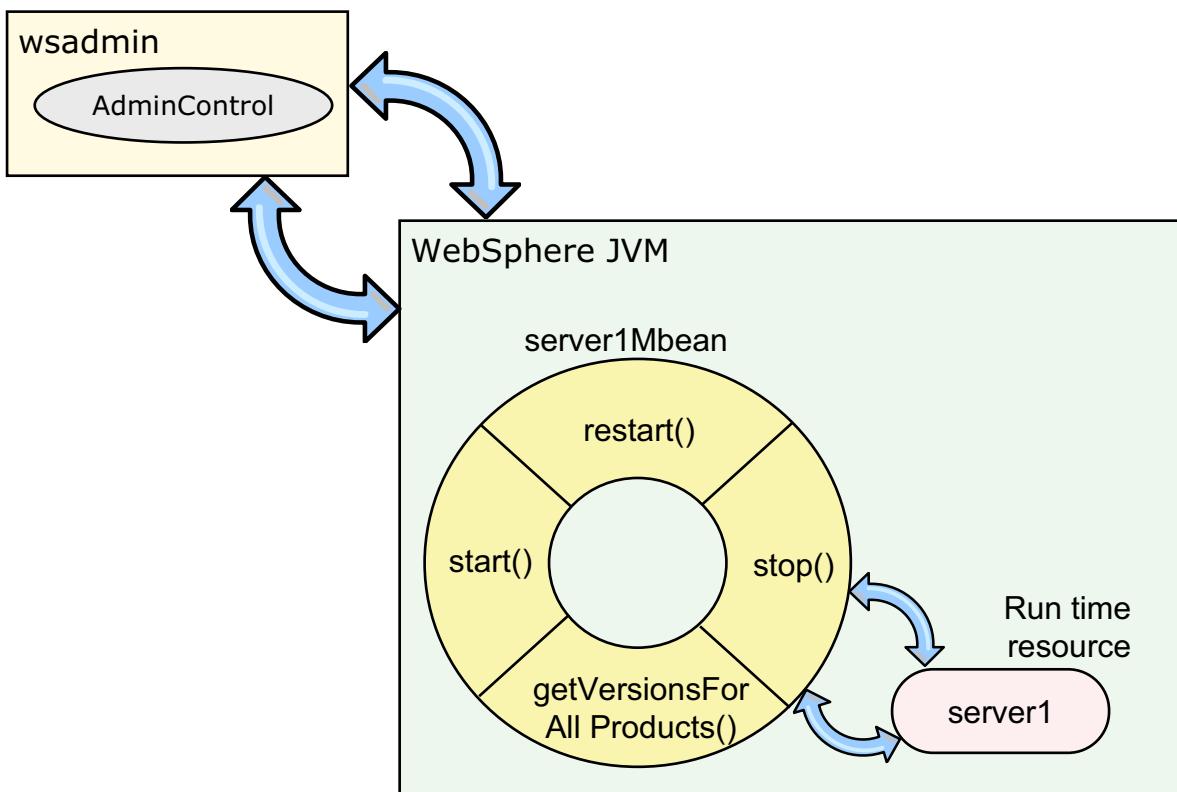
© Copyright IBM Corporation 2013

Figure 9-3. The AdminControl object (1 of 2)

WA680 / VA6801.0

Notes:

The AdminControl object (2 of 2)



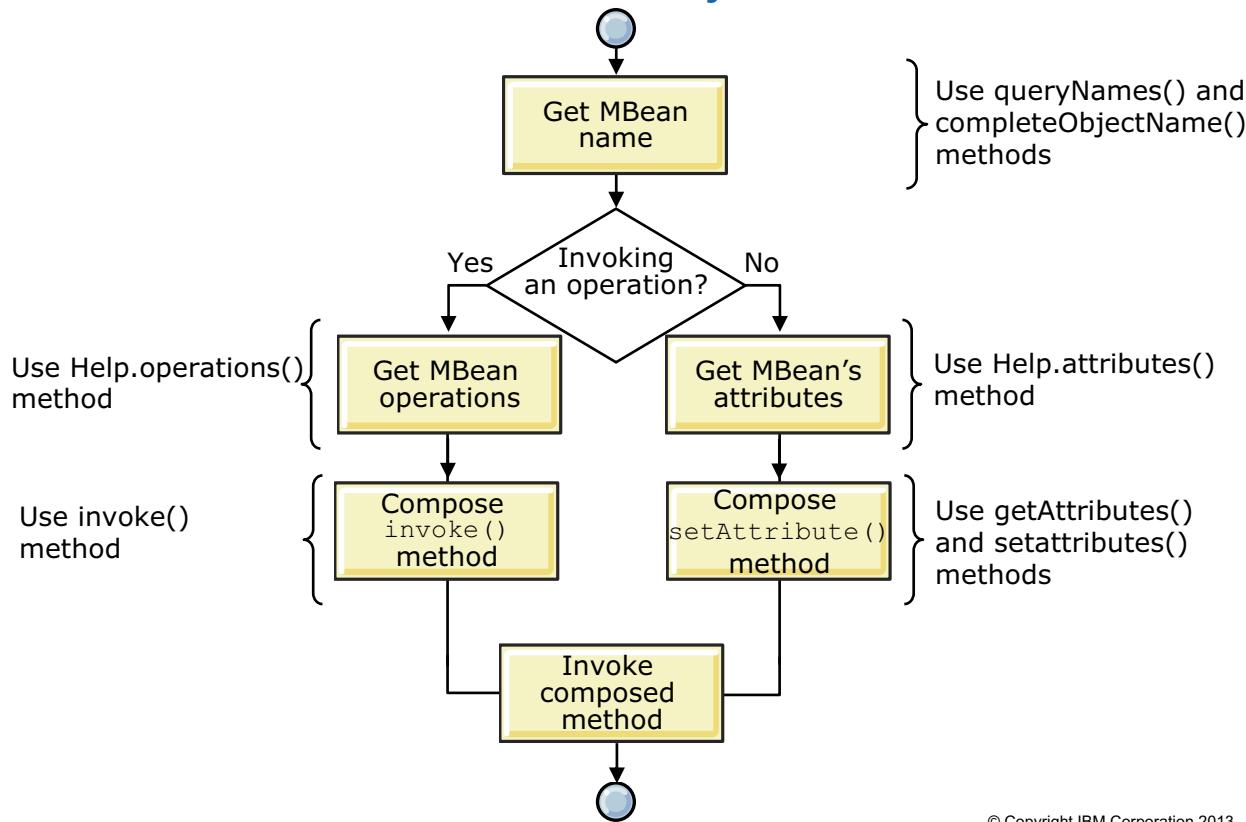
© Copyright IBM Corporation 2013

Figure 9-4. The AdminControl object (2 of 2)

WA680 / VA6801.0

Notes:

How to use the AdminControl object



© Copyright IBM Corporation 2013

Figure 9-5. How to use the AdminControl object

WA680 / VA6801.0

Notes:

MBean object names (1 of 2)

- To start an administrative method on an MBean, a reference to it must be obtained, its *object name*.
- The JMX class *ObjectName* represents the *object name* of an MBean
 - An ObjectName instance uniquely identifies an MBean
- An *object name* can be written as a string with the following format:
 - “domainName:keyProperty1=value1,keyProperty2=value2 ...”
 - Example:

```
"WebSphere:name=NodeAgent,process=nodeagent,platform=common,  
node=Node01,diagnosticProvider=true,version=8.5.0.0,  
type=NodeAgent,mbeanIdentifier=NodAgent,cell=washost00Cell01,  
spec=1.0"
```

© Copyright IBM Corporation 2013

Figure 9-6. MBean object names (1 of 2)

WA680 / VA6801.0

Notes:

MBean object names (2 of 2)

- An *object name* consists of two parts:
 - Domain name
 - Always has a value of WebSphere for MBeans created by WebSphere Application Server
 - Key properties, including:
 - type: Type of object accessible through the MBean (MBean type)
 - name: Display name of object
 - node: Name of node where object runs
 - process: Name of server process in which object runs
 - mbeanIdentifier: Correlates the MBean instance with the corresponding configuration data

© Copyright IBM Corporation 2013

Figure 9-7. MBean object names (2 of 2)

WA680 / VA6801.0

Notes:

Object name templates

- An *object name template* is a string that contains a portion of an object name
 - Used for pattern matching when searching for actual object names
 - At most one *keyProperty=value* pair can be substituted with an asterisk (*) to represent a wildcard
 - Allows you to indicate a wildcard match for the unspecified key properties
 - Specifying more key properties allows you to narrow the scope of the search
 - Example:

```
"WebSphere:type=NodeAgent,* "
```

- Can be used to search for all MBeans of type *NodeAgent*

© Copyright IBM Corporation 2013

Figure 9-8. Object name templates

WA680 / VA6801.0

Notes:

Getting the object name of an MBean (1 of 2)

- `queryNames (objectNameTemplate)`
 - Returns a string that contains one or more *object names* of MBeans that match the *objectNameTemplate*.
 - *objectNameTemplate* is a string that contains an *object name template*
 - Make sure to always use the wildcard character (*) in the template (usually at the end), otherwise the method returns an empty string if it fails to find an exact match.
 - If the *domain name* is omitted, “WebSphere” is assumed.
 - If more than one MBeans match the template, the returned string contains each MBean’s object name that is separated by a line separator character.
 - Use the `split()` method to convert the string to into a list and retrieve each individual object name.

© Copyright IBM Corporation 2013

Figure 9-9. Getting the object name of an MBean (1 of 2)

WA680 / VA6801.0

Notes:

Getting the object name of an MBean (2 of 2)

- `completeObjectName (objectNameTemplate)`
 - Returns a string that contains a single *object name* of an MBean that matches the *objectNameTemplate*.
 - If more than one MBean matches the template, only the object name of the first match is returned.

© Copyright IBM Corporation 2013

Figure 9-10. Getting the object name of an MBean (2 of 2)

WA680 / VA6801.0

Notes:

Examples for queryNames and completeObjectName

Object name template	Lists all MBeans with a display name of "plants1"
<pre>print AdminControl.queryNames ("WebSphere:name=plants1,*")</pre>	
<pre>WebSphere:name=plants1,process=plants1,platform=proxy,node=Node01,j2eeType=J2EEServer,version=8.5.0.0,type=Server,mbeanIdentifier=cells/was85host00Cell01/nodes/Node01/servers/plants1/server.xml#Server_1169660824562,cell=was85host00Cell01,spec=1.0,processType=ManagedProcess</pre>	
 One match that is found; object name of plants1 server MBean returned	
<pre>print AdminControl.queryNames ("WebSphere:type=Server,*")</pre>	
<pre>WebSphere:name=dmgr,process=dmgr,platform=proxy,node=dmgrNode,j2eeType=J2EE Server,version=8.5.0.0,type=Server,mbeanIdentifier=cells/was85host00Cell01/nodes/dmgrNode/servers/dmgr/server.xml#Server_1,cell=was85host00Cell01,spec=1.0,processType=DeploymentManager</pre>	
 Object name of "dmgr" deployment manager MBean	
<pre>...</pre>	
<pre>WebSphere:name=plants1,process=plants1,platform=proxy,node=Node01,j2eeType=J2EEServer,version=8.5.0.0,type=Server,mbeanIdentifier=cells/was85host00Cell01/nodes/Node01/servers/plants1/server.xml#Server_1169660824562,cell=was85host00Cell01,spec=1.0,processType=ManagedProcess</pre>	
 Object name of "plants1" server MBean	

© Copyright IBM Corporation 2013

Figure 9-11. Examples for queryNames and completeObjectName

WA680 / VA6801.0

Notes:

Determining the attributes of an MBean

- Help.attributes (*objectName*)
 - Returns the name, type and access policy of all of the attributes of the MBean identified by *objectName*
 - *objectName* is a string that contains the MBean's *object name*
- Help.attributes (*objectName*, *attributeName*)
 - Returns the name, type and access policy of the attribute *attributeName* of the MBean identified by *objectName*, along with a description of *attributeName*

```
dmgrMBeanName=AdminControl.completeObjectName ("WebSphere:type=Server,name=dmgr,*")
print Help.attributes (dmgrMBeanName)
Attribute          Type          Access
name              java.lang.String  RO
shortName         java.lang.String  RO
threadMonitorInterval int          RW
. . .

print Help.attributes (dmgrMBeanName, "shortName")
Attribute          Type          Access
shortName         java.lang.String  RO
```

Description: The short name of the server.

© Copyright IBM Corporation 2013

Figure 9-12. Determining the attributes of an MBean

WA680 / VA6801.0

Notes:

Determining the operations of an MBean

- `Help.operations(objectName)`
 - Returns the return type and name of the operations available for the MBean identified by *objectName*
 - *objectName* is a string that contains the MBean's *object name*
- `Help.operations(objectName, operationName)`
 - Returns a description of *operationName* for the MBean identified by *objectName*

```
print Help.operations(dmgrMBeanName)
Operation
java.lang.String getName()
java.lang.String getShortName()
int getThreadMonitorInterval()
...
print Help.operations(dmgrMBeanName, "getShortName")
java.lang.String getShortName()

Description: getter for attribute shortName

Parameters:
```

© Copyright IBM Corporation 2013

Figure 9-13. Determining the operations of an MBean

WA680 / VA6801.0

Notes:

Other useful Help methods for MBeans

- `Help.classname (objectName)`
 - Returns the class name of the MBean identified by *objectName*
 - *objectName* is a string that contains the MBean's *object name*
- `Help.constructors (objectName)`
 - Returns the names of the constructors of the MBean identified by *objectName*
- `Help.description (objectName)`
 - Returns the description of the MBean identified by *objectName*
- `Help.notifications (objectName)`
 - Returns the names of the notifications of the MBean identified by *objectName*
- `Help.all (objectName)`
 - Returns all available information for the MBean identified by *objectName*

```

print Help.classname (dmgrMBeanName)
javax.management.modelmbean.RequiredModelMBean
print Help.description (dmgrMBeanName)
Managed object for overall server process.
print Help.notifications (dmgrMBeanName)
Notifications
j2ee.state.starting
j2ee.state.running
j2ee.state.stopping
j2ee.state.stopped
j2ee.state.failed
. . .

```

© Copyright IBM Corporation 2013

Figure 9-14. Other useful Help methods for MBeans

WA680 / VA6801.0

Notes:

Displaying the value of MBean attributes

- `getAttribute(objectName, attributeName)`
 - Returns a string that contains the value of the attribute *attributeName* of the MBean identified by *objectName*
 - *objectName* is a string that contains the MBean's *object name*
 - *attributeName* is a string that contains the name of the MBean attribute to be retrieved
- `getAttribute_jmx(objectName, attributeName)`
 - Same as `getAttribute` except *objectName* is an *ObjectName* instance that identifies the MBean
- `getAttributes(objectName, attributeNames)`
 - Returns a string that contains the values of the attributes that are specified in *attributeNames* for the MBean identified by *objectName*
 - *objectName* is a string that contains the MBean's *object name*
 - *attributeNames* is a string or a list that contains the names of the MBean attributes to be retrieved
 - Example:
“*attributeName1 attributeName2*” or
[“*attributeName1*”, “*attributeName2*”]
- `getAttributes_jmx(objectName, attributeNames)`
 - Returns a JMX AttributeList containing the values of the attributes that are specified in *attributeNames* for the MBean identified by *objectName*
 - *objectName* is an *ObjectName* instance that identifies the MBean

© Copyright IBM Corporation 2013

Figure 9-15. Displaying the value of MBean attributes

WA680 / VA6801.0

Notes:

Example: Displaying the value of MBean attributes

```

serverObjectNameString =
AdminControl.completeObjectName ("WebSphere:type=Server,name=plants1,*")

print AdminControl.getAttribute(serverObjectNameString, "name")
plants1

print AdminControl.getAttributes(serverObjectNameString, ["name",
"cellName", "pid", "nodeName"])
[ [pid 2564] [name plants1] [cellName was85host00Cell01] [nodeName Node01]
]

plants1ObjectName = AdminControl.makeObjectName(serverObjectNameString)

plants1AttrList = AdminControl.getAttributes_jmx(plants1ObjectName,
["name", "cellName", "pid", "nodeName"])

plants1AttrList
[javax.management.Attribute@14e014e0, javax.management.Attribute@15401540,
javax.management.Attribute@15d415d4, javax.management.Attribute@16141614]
```

© Copyright IBM Corporation 2013

Figure 9-16. Example: Displaying the value of MBean attributes

WA680 / VA6801.0

Notes:

Modifying MBean attributes (1 of 2)

- `setAttribute(objectName, attributeName, attributeValue)`
 - Sets the value of the attribute *attributeName* of the MBean identified by *objectName* to *attributeValue*
 - *objectName* is a string that contains the MBean's *object name*
 - *attributeName* is a string that contains the name of the MBean attribute to be modified
 - *attributeValue* is a string that contains the value to be set
- `setAttributes(objectName, attributeNamesAndValues)`
 - Sets the attributes of the MBean identified by *objectName* according to the name-value pairs specified in *attributeNames*
 - Returns a string that contains the names of the attributes that are set
 - *objectName* is a string that contains the MBean's *object name*
 - *attributeNamesAndValues* is a string or a list that contains the name-value pairs of attributes to set
 - String format:
 - “[[attrName1 attrValue1] [attrName2 attrValue2] ...]”
 - List format:
 - [[“attrName1”, attrValue2], [“attrName2”, attrValue2], ...]

© Copyright IBM Corporation 2013

Figure 9-17. Modifying MBean attributes (1 of 2)

WA680 / VA6801.0

Notes:

Modifying MBean attributes (2 of 2)

- `setAttribute_jmx(objectName, attributeObject)`
 - Sets the attribute that is represented by `attributeObject` of the MBean identified by `objectName`
 - `objectName` is an `ObjectName` instance that identifies the MBean
 - `attributeObject` is an instance of the JMX Attribute class that identifies the attribute object and contains its new value

- `setAttributes_jmx(objectName, attributeList)`
 - Sets the attributes that are represented by `attributeList` of the MBean identified by `objectName`
 - Returns a JMX `AttributeList` containing the names of the attributes that are set
 - `objectName` is an `ObjectName` instance that identifies the MBean
 - `attributeList` is an instance of the JMX `AttributeList` class that contains the attributes to be set

© Copyright IBM Corporation 2013

Figure 9-18. Modifying MBean attributes (2 of 2)

WA680 / VA6801.0

Notes:

Considerations for modifying MBean attributes

- Updates to MBean attributes are transient.
 - Does not affect the persistent configuration of the object that they represent.
 - Do not survive a server shut-down.
- Attributes that are supported for manipulation by an MBean are generally different from the attributes that are supported by the corresponding configuration object.
 - Configuration object is likely to contain attributes that cannot be queried or set.

© Copyright IBM Corporation 2013

Figure 9-19. Considerations for modifying MBean attributes

WA680 / VA6801.0

Notes:

Example: Modifying MBean attributes

```

traceServiceObjectName =
AdminControl.completeObjectName ("WebSphere:type=TraceService,process=plant
s1,*")
print Help.attributes(traceServiceObjectName)

Attribute          Type          Access
ringBufferSize      int           RW
traceSpecification java.lang.String  RW
traceFileName       java.lang.String  RO

print AdminControl.getAttribute(traceServiceObjectName,
"traceSpecification")
*=info

AdminControl.setAttribute(traceServiceObjectName, "traceSpecification",
"*=all")
..
print AdminControl.getAttribute(traceServiceObjectName, "traceSpecification")
*=all

```

© Copyright IBM Corporation 2013

Figure 9-20. Example: Modifying MBean attributes

WA680 / VA6801.0

Notes:

Performing operations on MBeans (1 of 2)

- `invoke(objectName, operationName)`
 - Invokes *operationName* on the MBean identified by *objectName* without any parameter and returns the result of the invocation as a string
 - *objectName* is a string that contains the MBean's *object name*
 - *operationName* is a string that contains the name of the MBean operation to perform
- `invoke(objectName, operationName, parameters)`
 - Invokes *operationName* on the MBean identified by *objectName* with the parameters specified in *parameters* and returns the result of the invocation as a string
 - *parameters* is a string that contains the operation's parameters
 - Separate each parameter with a space
 - If a parameter contains a space, enclose it in quotation marks with escape characters (\")
 - Order and type of each parameter must match the output of the `Help.operations(objectName, operationName)` method

© Copyright IBM Corporation 2013

Figure 9-21. Performing operations on MBeans (1 of 2)

WA680 / VA6801.0

Notes:

Performing operations on MBeans (2 of 2)

- `invoke(objectName, operationName, parameters, parameterTypes)`
 - Invokes *operationName* on the MBean identified by *objectName* with the parameters specified in *parameters* whose types are defined in *parameterTypes*, and returns the result of the invocation as a string.
 - *parameterTypes* is a string that contains the types of the parameters that are specified in *parameters*.
 - This form should be used if *operationName* can be started with different parameter signatures and, requires parameter types to be specified.
- `invoke_jmx(objectName, operationName, parameters, parameterTypes)`
 - Invokes *operationName* on the MBean identified by *objectName* with the parameters specified in *parameters* whose types are defined in *parameterTypes*, and returns the result of the invocation as a string.
 - *objectName* is an ObjectName instance that identifies the Mbean.
 - *parameters* is a collection of objects, each representing a parameter for *operationName*.
 - *parameterTypes* is a collection of strings, each representing the types of each supplied parameter.

© Copyright IBM Corporation 2013

Figure 9-22. Performing operations on MBeans (2 of 2)

WA680 / VA6801.0

Notes:

Examples: Performing operations on MBeans (1 of 3)

```

serverName =           ← Gets object name of "plants1" server
AdminControl.completeObjectName ("WebSphere:type=Server,name=plants1,*")

AdminControl.getAttribute(serverName, "state")
'STARTED'   ← "plants1" server is started
AdminControl.invoke(serverName, "stop") ← Stops the "plants1" server
```

AdminControl.getAttribute(serverName, "state")
WASX7015E: Exception running command: ← "plants1" server is stopped
"AdminControl.getAttribute(serverName, "state")); exception information:
 javax.management.InstanceNotFoundException:
WebSphere:name=plants1,process=plants1,
platform=proxy,node=Node01,j2eeType=J2EEServer,version=8.5.0.0,type=Server,
mbeanIdentifier=cells/was85host00Cell01/nodes/Node01/servers/plants1/server.xml#Server_1169660824562,cell=was85host00Cell01,spec=1.0,processType=ManagedProcess

```

© Copyright IBM Corporation 2013

Figure 9-23. Examples: Performing operations on MBeans (1 of 3)

WA680 / VA6801.0

### Notes:

## Examples: Performing operations on MBeans (2 of 3)

```

traceObjectName = Gets object name of TraceService MBean on "plants1" server
AdminControl.completeObjectName ("WebSphere:type=TraceService,process=plants1,*")
AdminControl.getTraceString(traceObjectName, "traceSpecification")
'*=info'
print Help.operations(traceObjectName, "appendTraceString")
void appendTraceString(java.lang.String)

Description: Add the specified trace string to the trace state already
active in
the process runtime, without first clearing the current state.
Parameters: "appendTraceString" has one
Type java.lang.String
Name traceString
Description a String that complies to the TraceString grammar and passes
the checkTraceString() method.

```

← Trace string before change

← "appendTraceString" has one parameter

© Copyright IBM Corporation 2013

Figure 9-24. Examples: Performing operations on MBeans (2 of 3)

WA680 / VA6801.0

### Notes:

## Examples: Performing operations on MBeans (3 of 3)

```
AdminControl.invoke(traceObjectName, "appendTraceString",
"com.ibm.*=all=enabled")
```

AdminControl.getAttribute(traceObjectName, "traceSpecification")
'*=info:com.ibm.*=all' ← Trace string after change
```

Operation parameter

Trace string after change

© Copyright IBM Corporation 2013

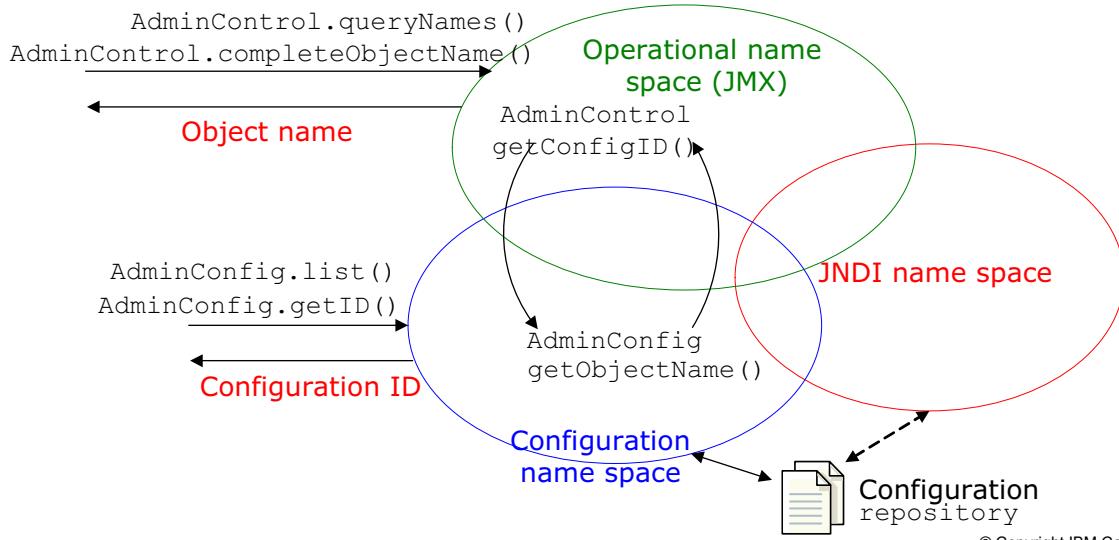
Figure 9-25. Examples: Performing operations on MBeans (3 of 3)

WA680 / VA6801.0

Notes:

Repository object name spaces

- An object in the configuration repository can have an identifier in one or more name spaces:
 - Java Naming and Directory Interface (JNDI) name space
 - Configuration name space
 - Operational (JMX) name space
- The `AdminConfig.getObjectName()` and `AdminControl.getConfigId()` methods allow you to cross between the configuration and operational name spaces.



© Copyright IBM Corporation 2013

Figure 9-26. Repository object name spaces

WA680 / VA6801.0

Notes:

AdminControl method reference (1 of 4)

Method name	Description
completeObjectName	Returns a string that contains the object name of the first MBean that matches the object name template
getAttribute_jmx	Returns the value of the specified attribute of the MBean identified by the JMX <i>ObjectName</i> instance
getAttribute	Returns the value of the specified attribute of the MBean identified by an object name string
getAttributes_jmx	Returns an <i>AttributeList</i> containing the values of the specified attributes for MBean identified by the <i>ObjectName</i> instance
getAttributes	Returns a string that contains name-value pairs for the specified attributes of the MBean identified by the object name string
getCell	Returns the cell name of the connected server
getConfigId	Returns the configuration ID of the configuration object corresponding to the MBean identified by the object name string, if any
getDefaultDomain	Returns the default domain name of the connected server (always "WebSphere")
getDomainName	Returns the domain name of the connected server (always "WebSphere")
getHost	Returns the name of the connected host

© Copyright IBM Corporation 2013

Figure 9-27. AdminControl method reference (1 of 4)

WA680 / VA6801.0

Notes:

AdminControl method reference (2 of 4)

Method name	Description
getMBeanCount	Returns the number MBeans registered with the server
getMBeanInfo_jmx	Returns the JMX <i>MBeanInfo</i> structure that corresponds to the <i>ObjectName</i> instance
getNode	Returns the node name of the connected server
getObjectInstance	Returns the <i>ObjectInstance</i> that matches the object name string
getPort	Returns the string representation of the port in use
getType	Returns the string representation of the connection type in use
help	Shows general and individual method help information
invoke_jmx	Invokes a method on the specified MBean, an <i>ObjectName</i> , the name of the method, an array of parameters, and a parameter type signature
invoke	Invokes a method on the specified MBean, an object name string, and optional parameters and parameter type signature
isRegistered_jmx	Returns true if the MBean identified by the supplied <i>ObjectName</i> is registered
isRegistered	Returns true if the MBean identified by the supplied object name string is registered
makeObjectName	Returns an <i>ObjectName</i> based on the object name string

© Copyright IBM Corporation 2013

Figure 9-28. AdminControl method reference (2 of 4)

WA680 / VA6801.0

Notes:

AdminControl method reference (3 of 4)

Method name	Description
queryNames_jmx	Returns a collection of <i>ObjectName</i> objects that match the <i>ObjectName</i> and <i>QueryExp</i>
queryNames	Returns a string representation of all object names that match the object name template
queryMBeans	Returns the set of ObjectInstances that match the object name string
reconnect	Reconnects to the server and clears out the local cache
setAttribute_jmx	Sets the specified attribute for an MBean, an <i>ObjectName</i> , and an <i>Attribute</i> object
setAttribute	Sets the specified attribute for an MBean, an object name string, an attribute name, and an attribute value
setAttributes_jmx	Sets the attributes of an MBean, an <i>ObjectName</i> , and an <i>AttributeList</i> object
setAttributes	Sets the attributes of an MBean, an object name string, and a name-value pair string or list

© Copyright IBM Corporation 2013

Figure 9-29. AdminControl method reference (3 of 4)

WA680 / VA6801.0

Notes:

AdminControl method reference (4 of 4)

Method name	Description
startServer	Starts the specified application server. This method is only valid in a distributed server environment.
stopServer	Stops the specified application server.
testConnection	Returns a string that indicates whether a connection test to a <i>DataSource</i> object is successful
trace	Sets the wsadmin trace specification

© Copyright IBM Corporation 2013

Figure 9-30. AdminControl method reference (4 of 4)

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe the steps that are involved in using the AdminControl object
- Identify the primary methods and resources that help in performing each step
- Explain the key concepts and syntactical rules that are required to use the AdminControl object
- Explain how to use the AdminControl object to modify MBean attributes and perform MBean operations

© Copyright IBM Corporation 2013

Figure 9-31. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. Fill in the blank: Run time objects (MBeans) are uniquely identified by their _____.
2. True or false: JMX versions of AdminControl methods require that an *ObjectName* instance is used to identify the target MBean instead of an object name string.
3. Which two AdminControl methods can be used to find the MBeans running in a server?
4. Which Help methods allow you to determine an MBean's attributes and operations?
5. True or false: Changes that are made to an MBean's attributes are permanently stored in the configuration repository.

© Copyright IBM Corporation 2013

Figure 9-32. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

4.

5.

Checkpoint answers

1. Fill in the blanks: Run time objects (MBeans) are uniquely identified by their object name.

2. True or false: JMX versions of AdminControl methods require that an *ObjectName* instance is used to identify the target MBean instead of an object name string.

Answer: True

3. Which two AdminControl methods can be used to find the MBeans running in a server?

Answer: `queryNames ()` and `completeObjectName ()`

4. Which Help methods allow you to determine an MBean's attributes and operations?

Answer: `Help.attributes ()` and `Help.operations ()`

5. True or false: Changes that are made to an MBean's attributes are permanently stored in the configuration repository.

Answer: False

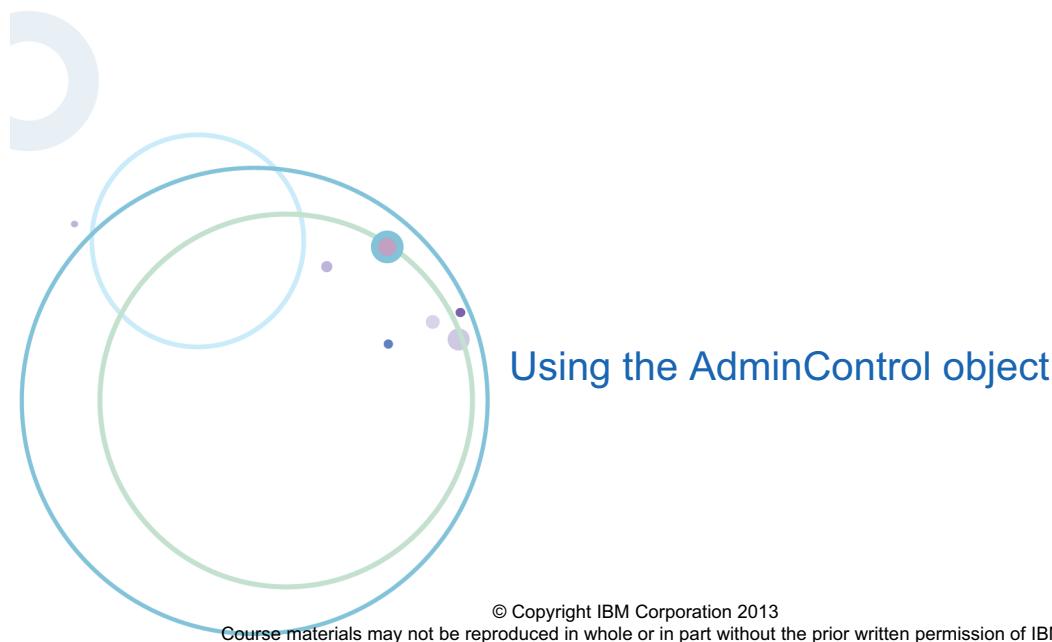
© Copyright IBM Corporation 2013

Figure 9-33. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 5



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 9-34. Exercise 5

WA680 / VA6801.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Apply the general steps that are required to make an operational or attribute change to an MBean by using the AdminControl object
- Use the primary methods and resources that help in performing each step
- Use the AdminControl object to query and modify MBean attributes, and start MBean operations

© Copyright IBM Corporation 2013

Figure 9-35. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 10. Administrative object basics: AdminTask

What this unit is about

This unit describes the purpose and common usage of the AdminTask object.

What you should be able to do

After completing this unit, you should be able to:

- Describe the overall steps that are involved in using the AdminTask object
- Identify the primary methods and resources that help in performing each step
- Explain the general syntax of an AdminTask command
- Explain how to use the AdminTask object to perform administrative tasks in interactive or batch mode

Unit objectives

After completing this unit, you should be able to:

- Describe the overall steps that are involved in using the AdminTask object
- Identify the primary methods and resources that help in performing each step
- Explain the general syntax of an AdminTask command
- Explain how to use the AdminTask object to perform administrative tasks in interactive or batch mode

© Copyright IBM Corporation 2013

Figure 10-1. Unit objectives

WA680 / VA6801.0

Notes:

When to use the AdminTask object

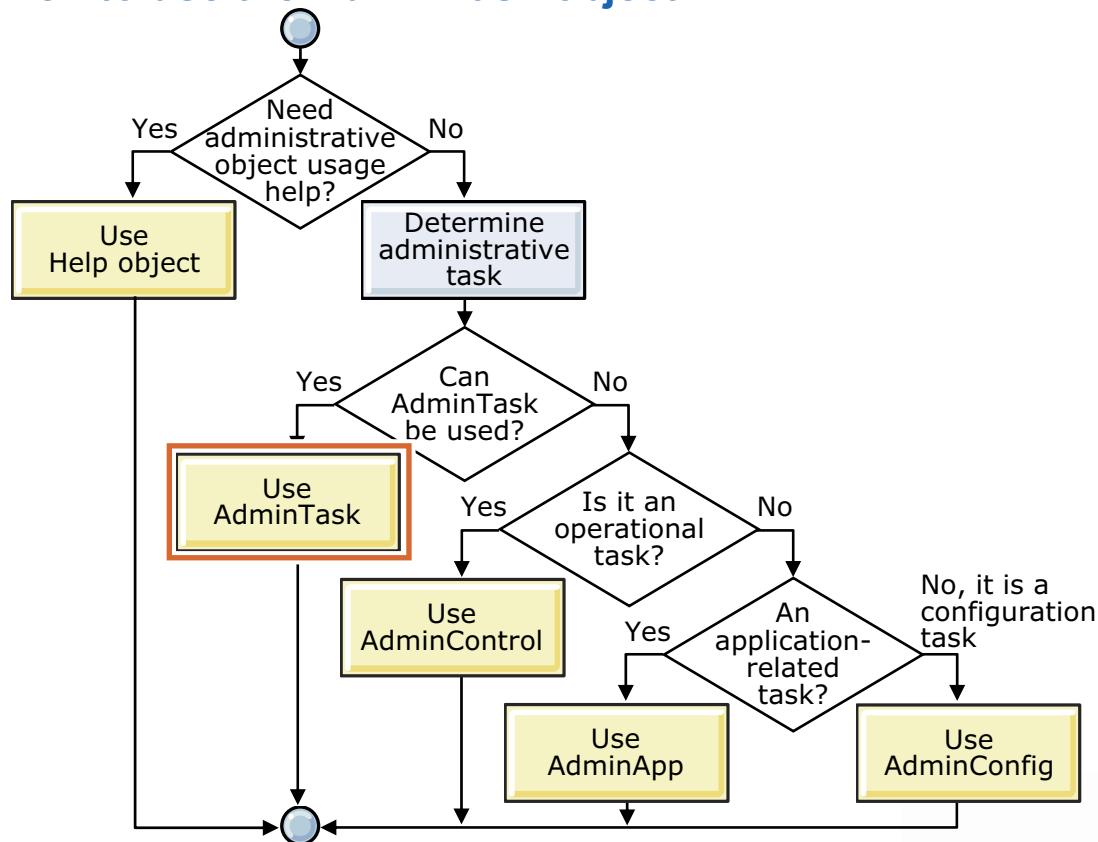


Figure 10-2. When to use the AdminTask object

WA680 / VA6801.0

© Copyright IBM Corporation 2013

Notes:

The AdminTask object

- Provides a set of task-oriented commands for performing administrative functions.
 - Simple alternative way to make configuration and operational changes
- Commands (or methods) are grouped by functionality.
 - Vary by WebSphere Application Server edition
 - Over 55 command groups that comprise more than 550 commands available in WebSphere Application Server Network Deployment V6.1
 - Command groups facilitate finding related commands
- Commands can be started with or without a connection to a server.
 - Can be used with wsadmin's *-conntype NONE* option
- Commands can be run in two modes:
 - Batch mode
 - Supply all required command parameters on the command line and start command
 - Regular way of starting a command
 - Interactive mode
 - Start the command in interactive mode
 - Supply all required parameters interactively as prompted by a text-based wizard
 - View the generated command line and run the command

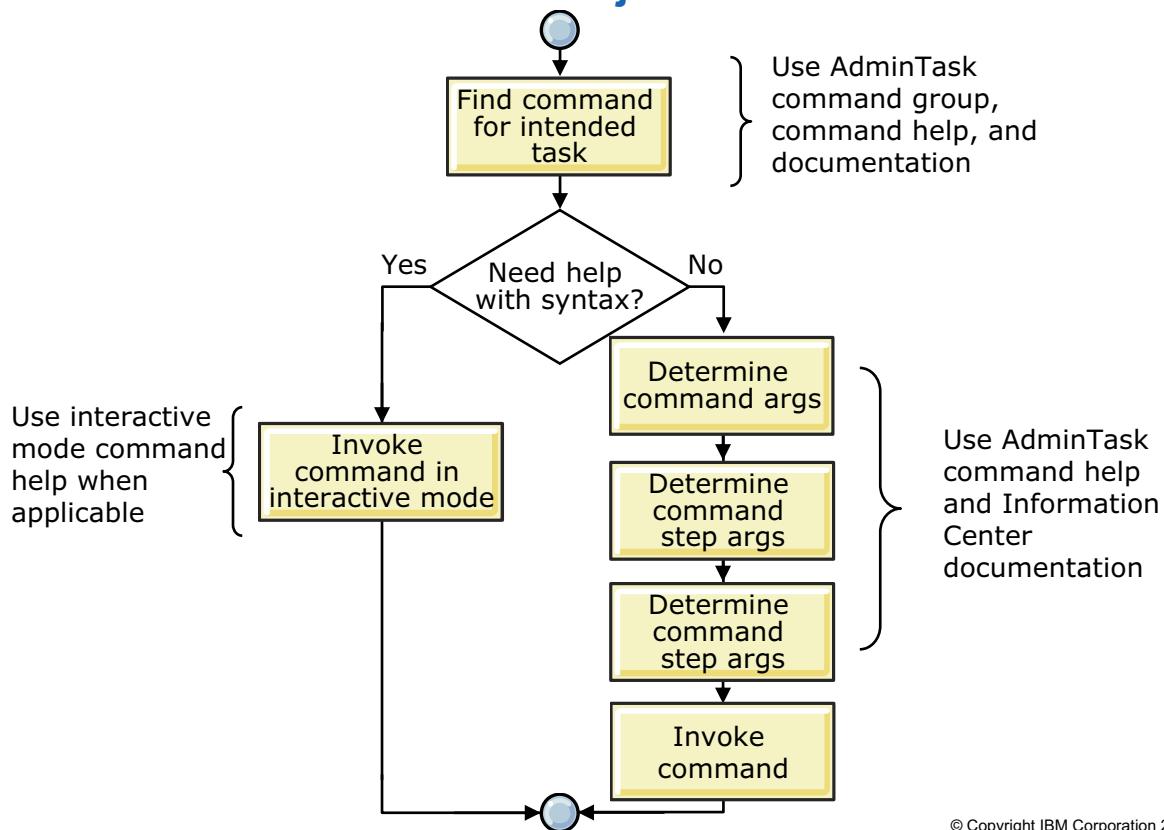
© Copyright IBM Corporation 2013

Figure 10-3. The AdminTask object

WA680 / VA6801.0

Notes:

How to use the AdminTask object



© Copyright IBM Corporation 2013

Figure 10-4. How to use the AdminTask object

WA680 / VA6801.0

Notes:

AdminTask command group reference (1 of 5)

Command group name	Contains commands to
AdminReports	Create configuration reports (inconsistencies and port usage)
AuthorizationGroupCommands	Create and manage authorization groups
AutoGen Commands	Auto-generate LTPA passwords and server Ids
CertificateRequestCommands	Create and manage certificate requests
ChannelFrameworkManagement	Manage the WebSphere Transport Channel service
ClusterConfigCommands	Create and delete application server clusters and cluster members
ConfigArchiveOperations	Export or import server configurations and entire cell configurations
CoreGroupBridgeManagement	Create and manage core group access points, TCP inbound channel port, and bridge interfaces
CoreGroupManagement	Create and manage core groups
CreateWebServerByHostName Commands	Specify the configuration properties for IBM HTTP Server
DescriptivePropCommands	Commands to configure Descriptive Properties

© Copyright IBM Corporation 2013

Figure 10-5. AdminTask command group reference (1 of 5)

WA680 / VA6801.0

Notes:

AdminTask command group reference (2 of 5)

Command group name	Contains commands to
DynamicSSLConfigSelection Commands	Create, delete, and query Dynamic SSL configuration selection objects for managing remote access
EventServiceCommands	Deploy and configure the Event service
EventServiceDBCommands	Configure the Event service databases
GenerateSecurityConfigCommand	Generate security configuration reports
IdMgrConfig	Configure virtual member managers
IdMgrDBSetup	Configure virtual member manager databases, property extension databases, and entry mapping databases
IdMgrRealmConfig	Configure virtual member manager realms
IdMgrRepositoryConfig	Configure virtual member manager repositories
JCAManagement	Configure Java 2 Connector Architecture (J2C) related resources
JDBCProviderManagement	Configure Java Database Connectivity (JDBC) related resources
KeyManagerCommands	Configure Key Managers
KeyReferenceCommands	Manage key references that are associated with key sets

© Copyright IBM Corporation 2013

Figure 10-6. AdminTask command group reference (2 of 5)

WA680 / VA6801.0

Notes:

AdminTask command group reference (3 of 5)

Command group name	Contains commands to
KeySetCommands	Manage key set groups
KeySetGroupCommands	Configure key set groups
KeyStoreCommands	Manage Key Stores
ManagedObjectMetadata	Get managed object metadata properties for a node
ManagementScopeCommands	Create, delete, and list management scopes
NodeConfigCommands	Change the host name
NodeGroupCommands	Administer a Node Group
PersonalCertificateCommands	Manage personal certificates
PortManagement	Manage WebSphere ports
ProfileCommands	Apply security settings that are selected during installation or profile creation time
ProfileCreationCommands	Prepare keys and keystores for a profile creation
ProxyManagement	Create and delete a proxy configuration for a web module
ResourceManagement	Manage resources

© Copyright IBM Corporation 2013

Figure 10-7. AdminTask command group reference (3 of 5)

WA680 / VA6801.0

Notes:

AdminTask command group reference (4 of 5)

Command group name	Contains commands to
SIBAdminBusSecurityCommands	Configure SIB security
SIBAdminCommands	Configure SIB queues and messaging engines
SIBJMSAdminCommands	Configure SIB JMS connection factories, queues, and topics
SIBWebServices	Configure service integration bus web services
SSLConfigCommands	Manage SSL configurations
SSLConfigGroupCommands	Commands for configuring SSL Configuration groups
ServerManagement	Configure servers
SignerCertificateCommands	Manage signer certificates
SpnegoTAICommands	Configure Spnego TAI
TAMConfig	Configure the embedded Tivoli Access Manager
TrustManagerCommands	Configure Trust Managers
UnmanagedNodeCommands	Configure unmanaged nodes
Utility	Perform common utility tasks and queries (useful for scripting)

© Copyright IBM Corporation 2013

Figure 10-8. AdminTask command group reference (4 of 5)

WA680 / VA6801.0

Notes:

AdminTask command group reference (5 of 5)

Command group name	Contains commands to
VariableConfiguration	View and modify variable values
WIMManagementCommands	Support for varying user and group management tasks
WMMMigration	Migrate WMM (WebSphere Member Manager) configuration and data to a virtual member manager
WSCertExpMonitorCommands	Manage the Certificate Expiration Monitor
WSGateway	Configure Web Services Gateway
WSNotificationCommands	Configure the WS-Notification service
WSNotifierCommands	Manage Notifiers
WSScheduleCommands	Create, delete, modify, and list a WS schedule
WizardCommands	Navigate and apply Security Wizard changes

- A number of miscellaneous commands are also available that do not belong to a group.
 - Refer to the WebSphere Application information center for a complete list

© Copyright IBM Corporation 2013

Figure 10-9. AdminTask command group reference (5 of 5)

WA680 / VA6801.0

Notes:

AdminTask help commands

- AdminTask.help()
 - Provides a description of the AdminTask object and lists its commands
 - Equivalent to Help.AdminTask()
- AdminTask.help(" -commandGroups")
 - Lists all of the AdminTask command groups
- AdminTask.help(" -commands")
 - Lists all of the AdminTask commands
- AdminTask.help(*aCommandGroupName*)
 - Displays detailed information for *aCommandGroupName*
- AdminTask.help(*aCommandName*)
 - Displays detailed information for *aCommandName*
- AdminTask.help(*aCommandName*, *aStepName*)
 - Displays detailed information for *aStepName* belonging to *aCommandName*

```
print AdminTask.help("ClusterConfigCommands") ←Command group help request
WASX8007I: Detailed help for command group: ClusterConfigCommands

Description: Commands for configuring application server clusters and
cluster members.
Commands:
createCluster - Creates a new application server cluster.
createClusterMember - Creates a new member of a cluster.
deleteCluster - Delete the configuration of an application server cluster.
```

© Copyright IBM Corporation 2013

Figure 10-10. AdminTask help commands

WA680 / VA6801.0

Notes:

AdminTask help command examples

```
print AdminTask.help("createCluster") ← Command help request
WASX8006I: Detailed help for command: createCluster
Description: Creates a new application server cluster.

Target object: None
Arguments: None ← Types of parameters for an AdminTask command
Steps:
    clusterConfig - Specifies the configuration of the new server cluster.
    replicationDomain - Specifies the configuration of a replication domain
    for this cluster.      Used for HTTP session data replication.
    convertServer - Specifies an existing server will be converted to be
    the first member of cluster.
    eventServiceConfig - Specifies the event service configuration of the
    new server cluster.
    promoteProxyServer - If a proxy server was specified for convertServer,
    apply its proxy settings to the cluster.
```

© Copyright IBM Corporation 2013

Figure 10-11. AdminTask help command examples

WA680 / VA6801.0

Notes:

AdminTask command general syntax

- A command can have an optional *target object* and zero or more *arguments* and *steps*.

- General syntax:

```
commandName ({options})
```

or

```
commandName ({targetObject} {, options})
```

- The *target object* is the configuration ID of the object on which the command operates.
- *options* contains the command's *arguments* and *steps*.
 - An *argument* is a name-value pair of the form:
[-argName, argValue]
 - A *step* represents an activity that is performed during the execution of the command and groups the arguments that it needs with the form:
[-stepName, stepArguments]

© Copyright IBM Corporation 2013

Figure 10-12. AdminTask command general syntax

WA680 / VA6801.0

Notes:

AdminTask command syntax combinations (1 of 2)

- `commandName()`
 - Simplest form, command has no target object, argument, or step.
 - Example:
 - `listServerTypes()`
 - Returns a string that contains all of the server types that are defined in the configuration
- `commandName(targetObject)`
 - Command has a target object.
 - example:
 - `showServerInfo(serverConfigurationID)`
 - Returns a string that contains detailed information for the server that is identified by *serverConfigurationID*

© Copyright IBM Corporation 2013

Figure 10-13. AdminTask command syntax combinations (1 of 2)

WA680 / VA6801.0

Notes:

AdminTask command syntax combinations (2 of 2)

- `commandName (options)`
 - Command has arguments or steps, but no target object.
 - Example:
 - `AdminTask.listServers ([-serverType, "APPLICATION_SERVER"])`
 - Returns a string that contains the configuration names of all servers that are application servers
- `commandName (targetObject, options)`
 - Command has a target object, and arguments or steps.
 - Example:
 - `AdminTask.createClusterMember (clusterID, [[-memberConfig, [[myNode, myClusterMember, "", "", "true", "false]]]])`
 - Creates the cluster member *myClusterMember* and makes it a member of the cluster that is identified by *clusterID*

© Copyright IBM Corporation 2013

Figure 10-14. AdminTask command syntax combinations (2 of 2)

WA680 / VA6801.0

Notes:

AdminTask options syntax

- AdminTask options can be specified in a string or list syntax:
 - Option string syntax:
 - “[{-argName1 argValue1} {-argName2 ...} {-stepName1 [[stepArgValue1 stepArgValue2 ...] ...] –stepName2 ...} {-delete [-collectionStepName [[stepKeyArgValue ...] ...] ...] ...} {-interactive}]”
 - or
 - “[{-argName1 argValue1} {-argName2 ...} {-stepName1 [[stepArgName1 stepArgValue1] [stepArgName2 stepArgValue2] ...] –stepName2 ...} {-delete [-collectionStepName [[stepKeyArgValue ...] ...] ...] ...} {-interactive}]”
- Option list syntax:
 - [{"-argName1", argValue1, "-argName2", ...}, {"-stepName1", [[stepArgValue1, stepArgValue2, ...]], "-stepName2", ...}, {"-delete", [-collectionStepName, [stepKeyArgValue1, ...]], ...}, {"-interactive"}]
- Order of argument values matters and must match documentation.
- The argument values of a step are stored in a list of lists even if there is only one element.

© Copyright IBM Corporation 2013

Figure 10-15. AdminTask options syntax

WA680 / VA6801.0

Notes:

Example: Using AdminTask in batch mode (1 of 3)

- Determine the command parameters.

```
print AdminTask.help("createClusterMember") ← Use command help.
WASX8006I: Detailed help for command: createClusterMember
Description: Creates a new member of an application server cluster.

Target object: Configuration object ID of the server cluster which the new
cluster member will belong to. ← These two parameters are optional; however, one of them is
Arguments: required to identify the containing cluster. You decide to use
clusterName ← the clusterName argument.
[clusterName] - Name of server cluster which the new cluster member will
belong to. ← These steps are optional, but you decide to use the memberConfig step
Steps: ← to configure the new cluster member at the same time that you create it.
      memberConfig - Specifies the configuration of a new member of the
      cluster.
      firstMember - Specifies additional information required to configure
      the first member of a cluster.
      . . .
```

© Copyright IBM Corporation 2013

Figure 10-16. Example: Using AdminTask in batch mode (1 of 3)

WA680 / VA6801.0

Notes:

Example: Using AdminTask in batch mode (2 of 3)

- Determine the step parameters.

```
AdminTask.help("createClusterMember", "memberConfig")  
WASX8013I: Detailed help for step: memberConfig
```

Description: Specifies the configuration of a new member of the cluster.

Collection: No ←Step is not a collection type.

Step has two required arguments and four optional ones Note their order.

Arguments:

*memberNode - Name of node which the new cluster member will belong to.
 *memberName - Name of new cluster member.
 memberWeight - Weight value of new cluster member.
 memberUUID - UUID of new cluster member.
 genUniquePorts - Generates unique port numbers for HTTP transports defined in the server.
 . . .

© Copyright IBM Corporation 2013

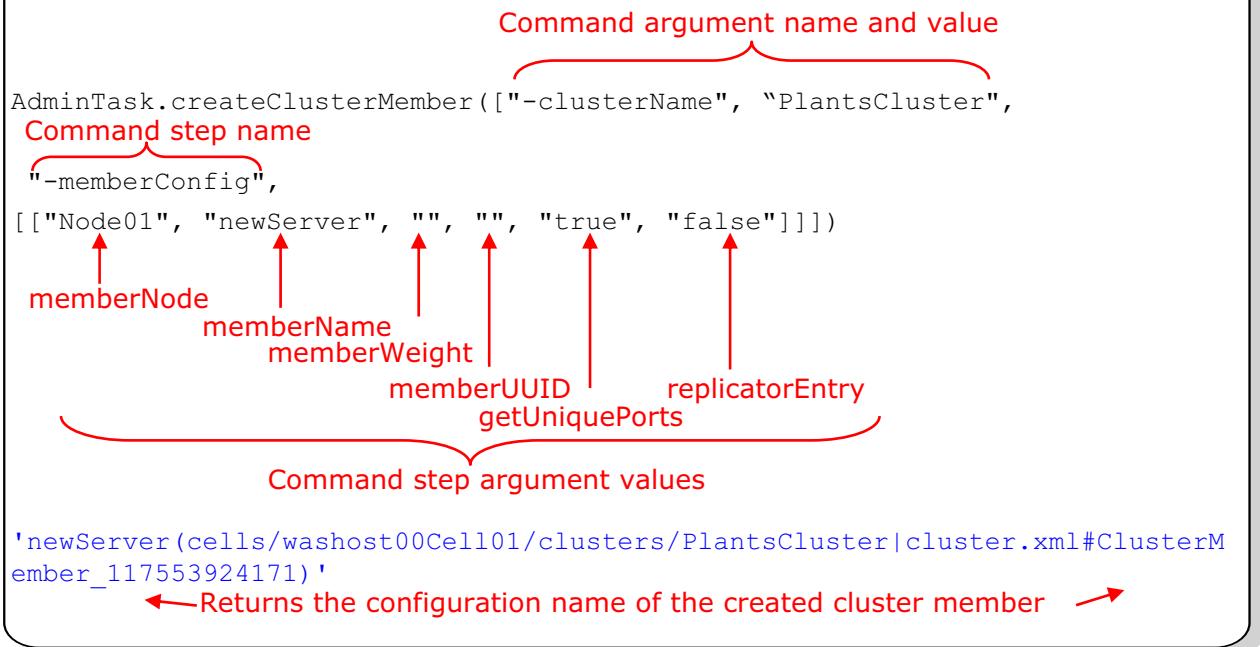
Figure 10-17. Example: Using AdminTask in batch mode (2 of 3)

WA680 / VA6801.0

Notes:

Example: Using AdminTask in batch mode (3 of 3)

- Compose the command invocation syntax.



© Copyright IBM Corporation 2013

Figure 10-18. Example: Using AdminTask in batch mode (3 of 3)

WA680 / VA6801.0

Notes:



AdminTask interactive mode

- AdminTask commands can be run in interactive mode.
 - Allows you to enter the target object and command parameters that are interactively based on prompts
 - Generates the command invocation syntax
 - Started with the –interactive command option
- Benefits of interactive mode
 - Enables you to run commands without knowing the actual syntax
 - Useful if you use wsadmin occasionally
 - Displays the generated command syntax
 - Allows you to learn by example
 - Can be immediately reused in a script

© Copyright IBM Corporation 2013

Figure 10-19. AdminTask interactive mode

WA680 / VA6801.0

Notes:

Example: Using AdminTask in interactive mode (1 of 3)

```
AdminTask.createClusterMember("-interactive") ← Specify interactive option.
```

Create Cluster Member

Creates a new member of an application server cluster.

Cluster Object ID: ← Press **Enter** to skip an optional parameter.

Cluster Name (clusterName): **PlantsCluster** ← No need to put quotation marks around string values.

Creates a new member of an application server cluster.

-> *1. Member Configuration (memberConfig) ← Arrow indicates the current step selection. Asterisk indicates a required step.
 2. First Member Configuration (firstMember)
 3. Configure the event service during cluster member creation.
 (eventServiceConfig)
 4. Promote Proxy Server Settings To Cluster (promoteProxyServer)

S (Select)

N (Next)

C (Cancel)

H (Help)

Select [S, N, C, H]: [S] ← Default selection. Press **Enter** to select the indicated step.

© Copyright IBM Corporation 2013

Figure 10-20. Example: Using AdminTask in interactive mode (1 of 3)

WA680 / VA6801.0

Notes:

Example: Using AdminTask in interactive mode (2 of 3)

```

Member Configuration (memberConfig) ← Specify parameters for required
                                         memberConfig step.

*Node Name (memberNode):
*Member Name (memberName):
Member Weight (memberWeight):
Member UUID (memberUUID):
Generate Unique HTTP Ports (genUniquePorts): [true]
enable data replication (replicatorEntry): [false]
Specific short name of cluster member (specificShortName):.

Select [C (Cancel), E (Edit)]: [E] ← Press Enter to select Edit.
*Node Name (memberNode): Node01 ← Press Enter to ignore optional values
*Member Name (memberName): newServer ← or accept default values.
Member Weight (memberWeight):
Member UUID (memberUUID):
Generate Unique HTTP Ports (genUniquePorts): [true]
enable data replication (replicatorEntry): [false]
Specific short name of cluster member (specificShortName):.
  
```

} Step arguments are displayed along with their default values and requirement condition.

© Copyright IBM Corporation 2013

Figure 10-21. Example: Using AdminTask in interactive mode (2 of 3)

WA680 / VA6801.0

Notes:

Example: Using AdminTask in interactive mode (2 of 3)

Create Cluster Member

Creates a new member of an application server cluster.

1. Member Configuration (memberConfig)
- > 2. First Member Configuration (firstMember) ← Next command step is displayed for selection.
3. Configure the event service during cluster member creation.
(eventServiceConfig)
4. Promote Proxy Server Settings To Cluster (promoteProxyServer)

S (Select)

N (Next)

P (Previous)

F (Finish)

C (Cancel)

H (Help)

Select [S, N, P, F, C, H]: [F] Since the remaining steps are optional, press Enter to finish.

WASX7278I: Generated command line:

'newServer(cells/was61host00Cell01/clusters/PlantsCluster|cluster.xml#ClusterMember_1175563869296)' ← Returns the configuration name of the created cluster member.

© Copyright IBM Corporation 2013

Figure 10-22. Example: Using AdminTask in interactive mode (2 of 3)

WA680 / VA6801.0

Notes:

AdminTask utility commands

- Part of Utility command group
- `isFederated()`
 - Checks if the system is a single server or network deployment
 - Returns *true* or *false*
- `getDmgrProperties()`
 - Returns a list that contains the port number, name, and host name of the deployment manager
 - Returns an empty string if the system is a single server
- `changeHostName(nodeName, hostName)`

```
print AdminTask.isFederated()
True

print AdminTask.getDmgrProperties()
[ [port 8879] [name dmgr] [host washost00] ]
```

© Copyright IBM Corporation 2013

Figure 10-23. AdminTask utility commands

WA680 / VA6801.0

Notes:

Commonly used AdminTask commands

Command name	Description
createApplicationServer	Creates an application server
createCluster	Creates an application server cluster
createClusterMember	Creates a member of an application server cluster
createSIBus	Creates a Service Integration Bus
deleteCluster	Deletes the configuration of an application server cluster
deleteClusterMember	Deletes a member from an application server cluster
deleteServer	Deletes a server configuration
deleteSIBus	Deletes a named bus, including everything on it
getServerType	Returns the type of the specified server
listNodes	Returns the display names of the nodes in the cell or a nodegroup
listServers	Returns the configuration names of the servers in the configuration. Can be filtered by server type or node name
listServerTypes	Returns all of the server types that are defined in the cell or a node
showServerInfo	Shows the detailed information of a specified server

© Copyright IBM Corporation 2013

Figure 10-24. Commonly used AdminTask commands

WA680 / VA6801.0

Notes:

AdminTask usage considerations

- Replace existing and more complex AdminConfig commands in scripts with their AdminTask equivalent
 - Simplifies scripting
 - Lowers maintenance cost
- As more AdminTask commands become available, start to use them in place of the equivalent AdminConfig methods
- AdminTask can also be used to perform operational tasks
 - Not just for making configuration changes
 - Example command:
 - `updateAppOnCluster ([-ApplicationNames", appName1, ...])`
 - Synchronizes the nodes and restarts the cluster members where the specified applications are deployed

© Copyright IBM Corporation 2013

Figure 10-25. AdminTask usage considerations

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe the overall steps that are involved in using the AdminTask object
- Identify the primary methods and resources that help in performing each step
- Explain the general syntax of an AdminTask command
- Explain how to use the AdminTask object to perform administrative tasks in interactive or batch mode

© Copyright IBM Corporation 2013

Figure 10-26. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. True or false: When performing an administrative task, you should always first inquire if the AdminTask can be used to do it.
2. What is the recommended first step in using the AdminTask object?
3. True or false: Only selected AdminTask commands can be run in both batch and interactive mode.
4. In interactive mode, what does an asterisk (*) in front of an option or step name indicate?

© Copyright IBM Corporation 2013

Figure 10-27. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.
- 4.

Checkpoint answers

- True or false: When performing an administrative task, you should always first inquire if the AdminTask can be used to do it.

Answer: True

- What is the recommended first step in using the AdminTask object?

Answer: Search the AdminTask command groups for a command that can be used for the intended task

- True or false: Only selected AdminTask commands can be run in both batch and interactive mode.

Answer: False. All commands can be run in either batch or interactive mode

- In interactive mode, what does an asterisk (*) in front of an option or step name indicate?

Answer: An asterisk indicates that the option or step name is required.

© Copyright IBM Corporation 2013

Figure 10-28. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 6

Using the AdminTask object

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 10-29. Exercise 6

WA680 / VA6801.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Apply the general steps that are required to start a command in batch or interactive mode by using the AdminTask object
- Use the primary methods and resources that help in performing each step
- Use the AdminTask object to perform administrative tasks in interactive or batch mode

© Copyright IBM Corporation 2013

Figure 10-30. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 11. Introduction to the PlantsByWebSphere and messaging applications

What this unit is about

This unit describes the PlantsByWebSphere application architecture and how it is used to demonstrate WebSphere Application Server concepts and functionality.

Messaging producer and consumer applications are also presented.

What you should be able to do

After completing this unit, you should be able to:

- Describe the components and functions of the PlantsByWebSphere application

How you will check your progress

- Checkpoint questions
- Lab exercises

References

WebSphere Application Server Network Deployment V8.5 Information Center

Unit objectives

After completing this unit, you should be able to:

- Describe the components and functions of the PlantsByWebSphere application

© Copyright IBM Corporation 2013

Figure 11-1. Unit objectives

WA680 / VA6801.0

Notes:



Topics

- PlantsByWebSphere application
- WebSphere Default Messaging applications

© Copyright IBM Corporation 2013

Figure 11-2. Topics

WA680 / VA6801.0

Notes:

11.1. PlantsByWebSphere application

PlantsByWebSphere application



© Copyright IBM Corporation 2013

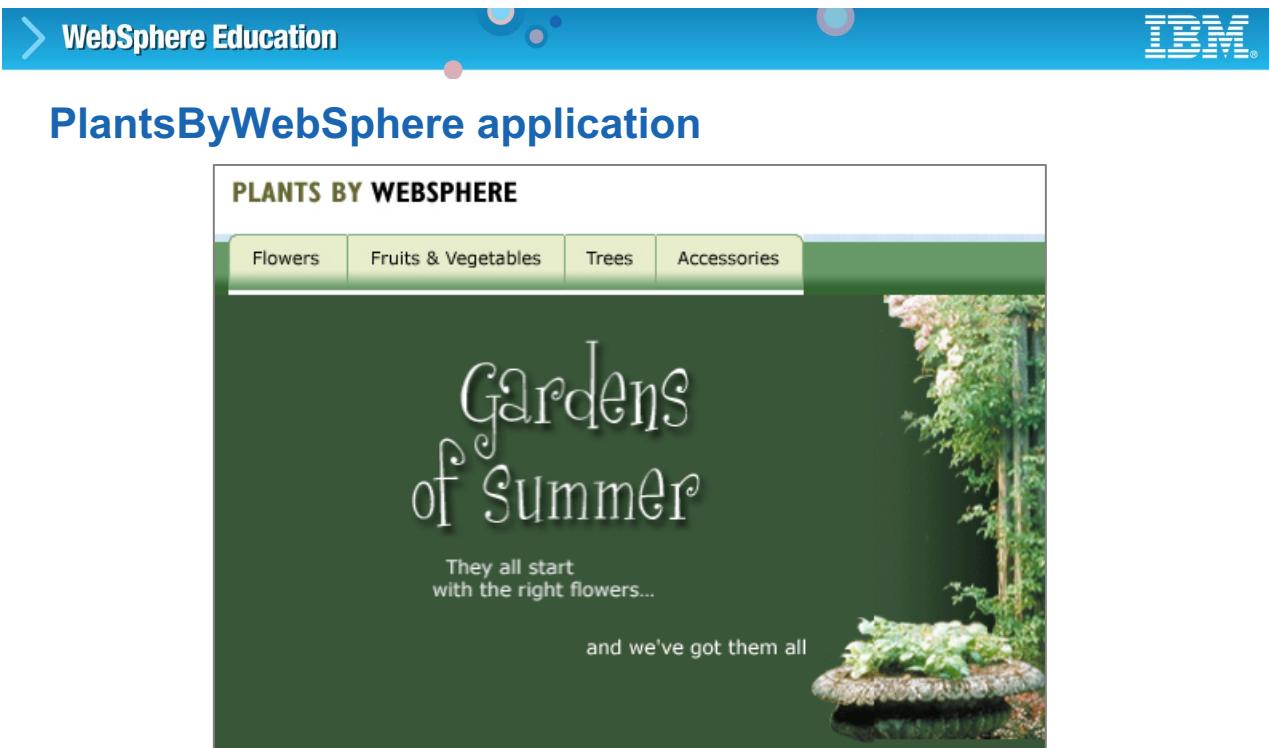
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 11-3. PlantsByWebSphere application

WA680 / VA6801.0

Notes:



- Simple shopping cart application
 - Added server information
 - Uses Derby, but can use DB2
 - Built as an enhanced EAR

© Copyright IBM Corporation 2013

Figure 11-4. PlantsByWebSphere application

WA680 / VA6801.0

Notes:

PlantsByWebSphere is a simple shopping cart application that is available with the WebSphere Application Server distribution. It uses Derby as its back-end database, but can also be configured to work with other databases such as DB2.

WebSphere Education



PlantsByWebSphere sample

- PlantsByWebSphere is available through the **Features > Samples Applications** that come with the WebSphere Application Server
 - Can be found in <was_root>/samples/PlantsByWebSphere
 - More samples are available through the information center
- The version of PlantsByWebSphere used in this course is altered slightly
 - To make the PlantsByWebSphere application more useful for educational purposes, several extra links are added to the bottom of the Help page
[\(View Server Info and Admin Home\)](#)

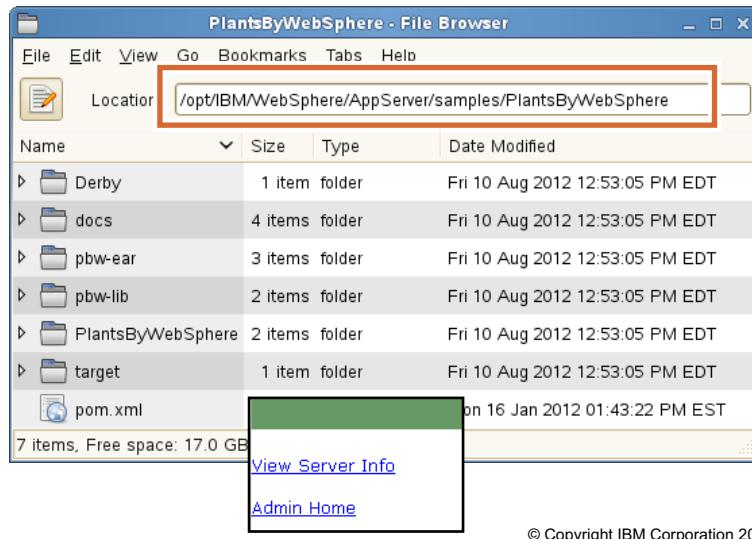


Figure 11-5. PlantsByWebSphere sample

WA680 / VA6801.0

Notes:

PlantsByWebSphere is a sample application. There are numerous other sample applications, which are now available through the information center. In previous versions of WebSphere, these other samples were available directly through the WebSphere Application Server distribution.

For educational purposes, the PlantsByWebSphere application is modified slightly. This modification was done to more easily demonstrate several functional issues that are useful to point out in a class such as this.

To demonstrate session failover in a clustered environment, an extra link was added to the Help page. This link shows server information so that it is easy to understand where the server affinity is mapped.

- To more easily demonstrate application security, an extra link that is called Admin Home was also added to the Help page. This link makes it easier to access the Admin servlet, which is protected through Java EE security. Otherwise, the user must type in the URL (or use a bookmark).

This course also uses DB2 instead of Derby, giving students a chance to configure data sources and JDBC drivers (instead of relying on the embedded definitions as part of the enhanced EAR). To rebuild the DB2 database, run the `CreateDB` script in the `software` directory.

PLANTS BY WEBSPHERE

Your shopping cart is currently empty

Flowers Fruits & Vegetables Trees Accessories

HOME : SHOPPING CART : LOGIN : HELP :

Gardens of Summer

They all start with the right flowers... and we've got them all

Tips	Specials		
Preserve extra grass seed by keeping it dry. Tape boxes and bags closed, or seal them into plastic bags. Be sure to remove extra air from the bags. Store all seed in a cool, dry area such as a garage or basement.	 Bonsai Tree \$30.00 each	 Red Delicious Strawberries \$3.50 (50 seeds)	 Tulips \$17.00 (10 bulbs)

Powered by **IBM WebSphere** e-business software ►

Flowers : Fruits & Vegetables : Trees : Accessories : Home : Shopping Cart : My Account : Login : Help

© Copyright IBM Corporation 2013

Figure 11-6. Home page: <http://<hostname>/PlantsByWebSphere>

WA680 / VA6801.0

Notes:

This screen shows the home screen for the PlantsByWebSphere application.

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories

Home > Sign in

Registration

Enter the information below to set up your account. We will never sell your information without your permission. With your permission, we may share your information with our trusted business partners.

Required fields are denoted with a red asterisk (*).

Login Information

E-mail address: *user@plants.com
 Password: *
 Verify Password: *

Contact Information

First Name: *Ima
 Last Name: *Gardner
 Address Line 1: *123 Main St
 Address Line 2:
 City: *Gotham
 State: *PA
 ZIP Code: *15222
 Phone (daytime): *412-555-1234

Register

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories

Home

Login or Register

If you are a returning customer and previously set up an account, please enter your e-mail address and password below.

E-mail address: sbywebsphere.ibm.com
 Password: *

Sign in

- Log in with
 - User: plants@plantsbywebsphere.ibm.com
 - Password: plants
- Or, register as new user
 - Enter your own data

© Copyright IBM Corporation 2013

Figure 11-7. Login and registration

WA680 / VA6801.0

Notes:

This screen shows the Login and Registration screens for the PlantsByWebSphere application.

There is no need to log in to the application unless the user wants to go through the purchasing screens.

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories

HOME : SHOPPING CART

Home > Sign in

Account Update

Enter the information below to update your account. This information will not be shared without your permission. With your permission we will only share your name and email address with our trusted business partners.

Required fields are denoted with a red asterisk (*).

Contact Information

First Name	* David
Last Name	* Grover
Address Line 1	* 123 Main Street
Address Line 2	Apt. C
City	* Raleigh
State	* NC
ZIP Code	* 27604
Phone (daytime)	* 919-555-1234

Update

- Click **My Account** on the bottom to see your account information
 - If you are not logged in, you are prompted to log in or register

Powered by **IBM WebSphere**

Flowers : Fruits & Vegetables : Trees : Accessories : Home : Shopping Cart **My Account** Login : Help

© Copyright IBM Corporation 2013

Figure 11-8. My Account

WA680 / VA6801.0

Notes:

This screen shows the My Account page for the PlantsByWebSphere application. If this screen shows user information, that means the user is already logged in. If not, a login screen is displayed.

The screenshot displays the WebSphere Education interface with a blue header bar. On the left is a logo with a right-pointing arrow and the text "WebSphere Education". On the right is the IBM logo. Below the header, there are three main sections, each representing a shopping category:

- Trees:** The tab "Trees" is highlighted with a red border. The page title is "PLANTS BY WEBSPHERE" and the sub-page title is "Trees". It shows a small green tree icon and the text "Page 1 of 1".
- Flowers:** The tab "Flowers" is highlighted with a red border. The page title is "PLANTS BY WEBSPHERE" and the sub-page title is "Flowers". It shows a small yellow flower icon and the text "Page 1 of 1".
- Fruits & Vegetables:** The tab "Fruits & Vegetables" is highlighted with a red border. The page title is "PLANTS BY WEBSPHERE" and the sub-page title is "Fruits & Vegetables". It shows a small green vegetable icon and the text "Page 1 of 1".

Each section includes a "Home" link at the top left and a copyright notice at the bottom right: "© Copyright IBM Corporation 2013".

Figure 11-9. Shopping

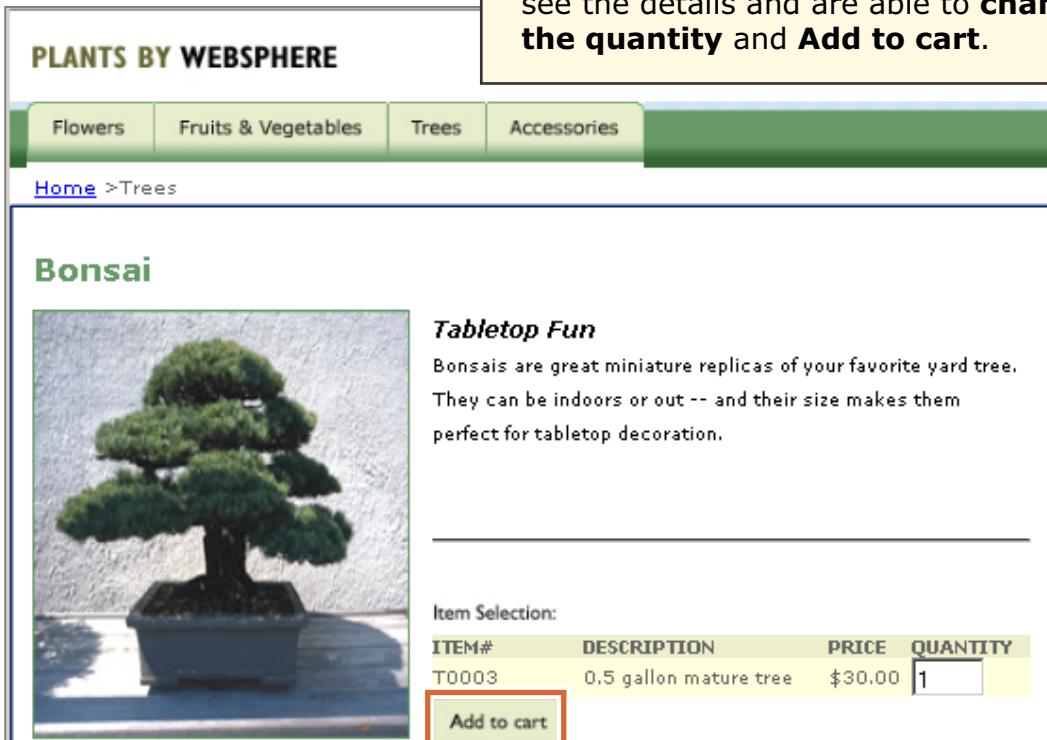
WA680 / VA6801.0

Notes:

This screen shows three of the shopping tabs for the PlantsByWebSphere application. These include Trees, Flowers, and Fruits & Vegetables. From these screens, the user can click the individual items and add them to their cart.

WebSphere Education

Select an item



The screenshot shows a web application titled "PLANTS BY WEBSPHERE". The navigation bar includes categories: Flowers, Fruits & Vegetables, Trees, and Accessories. The "Trees" category is selected. Below the navigation is a breadcrumb trail: Home > Trees. The main content area features a large image of a bonsai tree in a rectangular pot. To the right of the image is a descriptive text block: "Tabletop Fun" followed by a paragraph about bonsais being great miniature replicas of yard trees, suitable for indoor or outdoor use, and perfect for tabletop decoration. Below this is a section titled "Item Selection:" with a table:

ITEM#	DESCRIPTION	PRICE	QUANTITY
T0003	0.5 gallon mature tree	\$30.00	<input type="text" value="1"/>

An "Add to cart" button is located at the bottom of the table row. A callout box with a yellow background and black border points to this button with the text: "When you have an item page open, you see the details and are able to **change the quantity** and **Add to cart**".

© Copyright IBM Corporation 2013

Figure 11-10. Select an item

WA680 / VA6801.0

Notes:

This screen shows the details for an individual item within the PlantsByWebSphere application. Users can click **Add to cart** and continue to shop. Or, if they want, they can choose to check out.

PLANTS BY WEBSPHERE

- The shopping cart can be filled with numerous items
- Click **Checkout Now** when ready

Flowers Fruits & Vegetables Trees Accessories

Home

Shopping Cart

Here are the items you have selected. To recalculate your total after changing the quantity of an item, select the 'Recalculate' button. To remove an item from your cart, enter "0" as the quantity. Select 'Checkout Now' to begin the checkout process.

ITEM #	ITEM DESCRIPTION	PACKAGING	QUANTITY	PRICE	SUBTOTAL
F0002	Baby Breath	2 plants	<input type="text" value="2"/>	\$6.00	\$12.00
V0007	Watermelon	1 pkt. (100 seeds)	<input type="text" value="1"/>	\$2.00	\$2.00

Order Subtotal:\$14.00

[Continue Shopping](#) [Recalculate](#) [Checkout Now](#)

© Copyright IBM Corporation 2013

Figure 11-11. Shopping cart

WA680 / VA6801.0

Notes:

This screen shows the shopping cart screen for the PlantsByWebSphere application. From this screen, users can modify the quantities of their items, click **Checkout Now**, or continue to shop.



Checking out: Billing information

Shipping Information

Check here if the shipping address is the same as the billing address.

Full Name *
 Address Line 1 *
 Address Line 2
 City *
 State *
 Zip Code *
 Phone (daytime) *

- Checkout requires more information, including credit card data
- Click **Continue**

Shipping Method

Select a shipping method below. Your order total will be updated on the next page.

Shipping Method *

Credit Card *
 orderinfo:ccardnum: Credit card numbers must be entered as XXXX XXXX XXXX XXXX.
 Card Number *
 Expiration Month *
 Expiration Year *
 Cardholder Name *

© Copyright IBM Corporation 2013

Figure 11-12. Checking out: Billing information

WA680 / VA6801.0

Notes:

This screen shows the first of the checkout screens for the PlantsByWebSphere application. This screen is where users confirm the billing and shipping information. They also must enter credit card data. Be careful here, as the format of the credit card information must be XXXX XXXX XXXX XXXX (with spaces).



Checking out: Submit

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories

Home > Shopping Cart >

Review Your Order

Review your order below and select 'Submit Order' at the bottom to place your order. You can also add more items to your order by selecting 'Continue Shopping'.

Order Information					
ORDER TOTAL	SHIPPING ADDRESS	BILLING ADDRESS			
\$10.99	David Grover 123 Main Street Apt. C Raleigh, NC 27604 919-555-1234	David Grover 123 Main Street Apt. C Raleigh, NC 27604 919-555-1234			

Order Details

ITEM #	ITEM DESCRIPTION	PACKAGING	QUANTITY	PRICE	SUBTOTAL
V0002	Ornamental Gourd	1 pkt. (100 seeds)	1	\$1.50	\$1.50
A0008	Gloves	3 pairs per pack	1	\$4.50	\$4.50

Order Subtotal: \$6.00

Shipping, Standard Ground (3 to 6 business days) \$4.99: \$4.99

Order Total: \$10.99

[Continue Shopping](#) [Submit Order](#)

© Copyright IBM Corporation 2013

Figure 11-13. Checking out: Submit

WA680 / VA6801.0

Notes:

When the billing information is entered, the user is able to click **Submit Order**.

PLANTS BY WEBSPHERE

Flowers Fruits & Vegetables Trees Accessories

Home >

Help

Plants By WebSphere provides limited help support. See the sample docs directory for documentation on the design, building, and installation of the sample.

Debug mode has been tied to the JSF project stage declaration. Debug messages will be displayed when the web app's javax.faces.PROJECT_STAGE context param is set to either Development or UnitTest. A value of SystemTest or Production will turn off debug output. The current state of debugging is indicated in the check box below.

Debug messages enabled

If the database becomes corrupted for some reason, the button below can be used to delete all data currently in the database and populate it with a fresh set of data. If this does not work, stop the server and repeat the prerequisite steps found in the docs directory to unzip the Derby database.

[Reset database](#)

[View Server Info](#)

[Admin Home](#)

• The Help page provides an entry point to the following links:

- [View Server Info](#)
- [Admin Home](#)

Powered by
IBM WebSphere
e-business software

Flowers : Fruits & Vegetables : Trees : Accessories : Home : Shopping Cart : © Copyright IBM Corporation 2013

Figure 11-14. Help page

WA680 / VA6801.0

Notes:

The Help page provides an entry point to the following links: View Server Info and Admin Home. These links were added to the PlantsByWebSphere application especially for this course. They are not part of the PlantsByWebSphere EAR file that is distributed with the product.

View Server Info page

The **View Server Info** page shows which server is hosting the connection. It also displays the **Session Data** (and time created).

Runtime server information

Cell	Node	Process	Session Data	Session Created
was85host01Node01Cell	was85host01Node01	server1	null	null

Session Data Update Refresh

Show cookies

PLANTS BY WEBSHnERE

Runtime server information

Cell	Node	Process	Session Data	Session Created
was85hostNode01Cell	was85host01Node01	server1	Stephanie	Fri Aug 10 15:18:56 EDT 2012

Session Data Update Refresh

Show cookies

HOME

© Copyright IBM Corporation 2013

Figure 11-15. View Server Info page

WA680 / VA6801.0

Notes:

The View Server Info page shows which server is hosting the current connection. This information is useful for demonstrating server failover, since it is important to know which cluster member to stop.

The session data is used to demonstrate failover of session information through memory-to-memory replication.

The PlantsByWebSphere application does not store the shopping cart in an HTTP session object, so it does not failover correctly upon a server failure. Instead, this session information field was created to demonstrate HTTP session object failover.

PLANTS BY WEBSHIRE

HOME : ADMIN HOME

[Manage BackOrders](#) - View backorder inventory, order from suppliers, add new stock to inventory.

[Supplier Configuration](#) - Configure the Supplier.

Powered by
IBM WebSphere®
e-business software >

PLANTS BY WEBSHIRE

[Admin Home](#)

Supplier Configuration

Enter the Supplier's Configuration Information

Full Name	Greenhouse By WebSphere
Street Address	4205 Miami Blvd.
City	Durham
State	NC
Zip	27709
Phone	919-555-1212
Location URL	<input type="text" value="http://localhost:9080/OrderProcessorEJB/services/FrontGate"/>

Notes:

- The Admin home page is used to demonstrate application security
- When configured, these pages require authentication

© Copyright IBM Corporation 2013

Figure 11-16. Admin home page

WA680 / VA6801.0

Notes:

The administrative pages are mapped to the SampAdmin security role. When application security is enabled, authentication is required to access these pages.



HTML documentation for PlantsByWebSphere (1 of 2)

- For more information about PlantsByWebSphere, there are several HTML files in the sample directory that can be useful

- Overview:**

`<was_root>/samples/PlantsByWebSphere/docs/index.html`



© Copyright IBM Corporation 2013

Figure 11-17. HTML documentation for PlantsByWebSphere (1 of 2)

WA680 / VA6801.0

Notes:

The following files contain potentially useful information about PlantsByWebSphere:

`<was_root>/samples/PlantsByWebSphere/docs/index.html`

`<was_root>/samples/PlantsByWebSphere/docs/techNotes.html`



HTML documentation for PlantsByWebSphere (2 of 2)

- **TechNotes:**

<was_root>/samples/PlantsByWebSphere/docs/techNotes.html

The screenshot shows a Mozilla Firefox window with the title bar "Plants By WebSphere Sample TechNotes - Mozilla Firefox". The address bar displays "file:///C:/Program Files/IBM/WebSphere/AppServer/samples/PlantsByWebSphere/docs/techNotes.html". The page content is titled "Plants By WebSphere Technotes". On the left, there is a sidebar with links: "Overview", "Getting Started", "Development and Build", and "Application Install and Management". The main content area is titled "Getting started". It contains text about creating a unique email account and a table with default login credentials.

E-mail address	plants@plantsbywebsphere.ibm.com
Password	plants

© Copyright IBM Corporation 2013

Figure 11-18. HTML documentation for PlantsByWebSphere (2 of 2)

WA680 / VA6801.0

Notes:

The Technotes page provides information about the default email address and password. Also, the database tables and SQL statement for creating them are shown. There is a description of the Java objects that are used in the application.

The Plants by WebSphere Sample incorporates the following technologies:

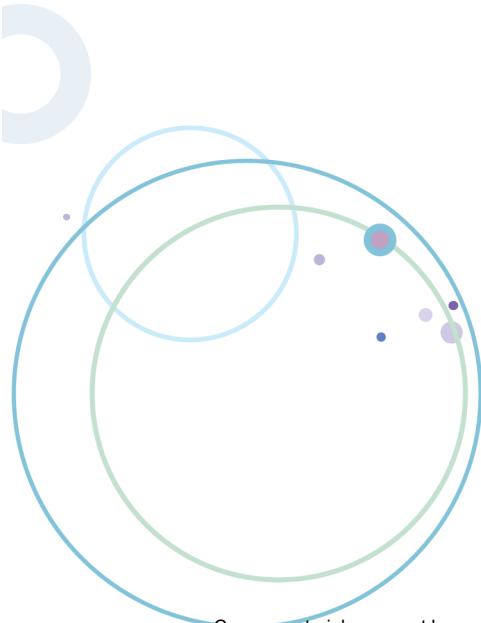
- Java Persistence API (JPA) entity beans
- Stateless session beans
- Stateful session beans
- Servlets
- JavaServer Faces (JSF) files and Facelets
- Java Platform, Enterprise Edition (J2EE) security

The Plants by WebSphere application is supported through a series of JSF pages and HTML pages. These pages communicate with the following servlets: AccountServlet,

ShoppingServlet, ImageServlet, and AdminServlet. The servlets use the various enterprise bean business methods, which in turn, access data from the database as needed.

11.2. WebSphere Default Messaging applications

WebSphere Default Messaging applications



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 11-19. WebSphere Default Messaging applications

WA680 / VA6801.0

Notes:

Message Sender Simulator

- Install to the PlantsCluster the EAR file: **MSGSSenderSimulator.ear**
 - Simulates the function of buying and selling stocks.
 - You can select how many shares to buy or sell.
 - For each operation, a message is placed on a JMS queue.
 - Each message has a transaction number that includes the name of the server that produced it and a sequence number.
 - You can view the action of placing the message on the queue in the SystemOut log file.



© Copyright IBM Corporation 2013

Figure 11-20. Message Sender Simulator

WA680 / VA6801.0

Notes:

This application is the message producer. It is running on both cluster members Plants1 and Plants2. You can send, buy, and sell messages from either cluster member by pressing the **Send messages** button.



Trade processor application

- Install to the PlantsCluster the EAR file: **TPApplication.ear**
 - This application uses a message-driven bean EJB (MDB) which listens on the same queue on which the simulator places messages.
 - As the MDB retrieves the messages, they are listed in a table on a web page and displayed to the user.
 - Every 30 seconds the oldest message is considered processed and is removed from the table.
 - As messages are received and discarded, trace entries are written to the SystemOut log file.

A screenshot of a web-based application titled "Trade Requests To Be Processed On Server: Plants1". The page has a yellow background. At the top, it says "Refresh every: 10 seconds". Below that is a horizontal menu bar with buttons labeled "Account", "Buy/Sell", "Symbol", "Qty", "Total cost", and "Transaction". The main area below the menu is currently empty, showing a large yellow space.

© Copyright IBM Corporation 2013

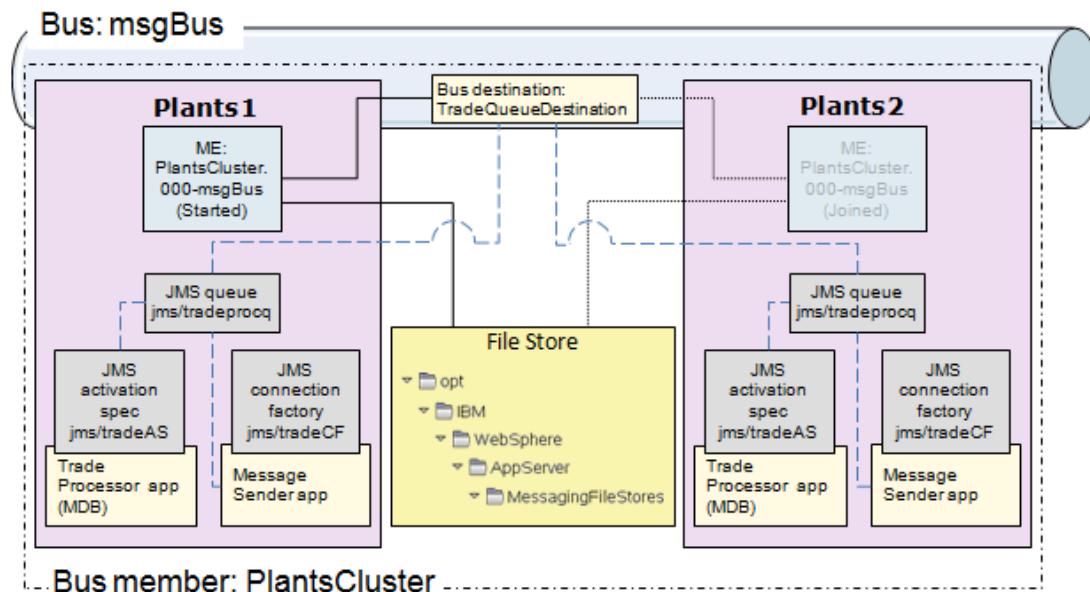
Figure 11-21. Trade processor application

WA680 / VA6801.0

Notes:

This application is the message consumer. Though the application is running on both cluster members, messages are processed only by the server that is currently hosting the messaging engine (ME).

Topology diagram



- Messaging resources include: Bus, Bus destination, Bus member, JMS connection factory, JMS activation spec, JMS queue, and File store

© Copyright IBM Corporation 2013

Figure 11-22. Topology diagram

WA680 / VA6801.0

Notes:

This diagram shows all of the components that are involved in WebSphere Default Messaging.

A service integration bus that is called msgBus is defined first. Then, the PlantsCluster is added as a member of the bus. By default, a messaging engine (ME) is active only on one of the cluster members. The HA manager decides which cluster member host the ME when both are running. A bus destination that is called TradeQueueDestination is defined that holds all of the messages that are produced. A message store is defined for the MEs for persisting messages. In this example, the messages are stored in the file system. JMS resources are defined for each cluster member to support the messaging applications. These resources are the JMS activation specification, JMS connection factory, and the JMS queue.



Monitor web page (1 of 2)

- http://<hostname>/Trade/processor/Monitor.html

Sending messages from server: Plants1

Select number of messages to be sent for each of the **Buy** and **Sell** categories, then click **Send messages**.

Account	Buy/Sell	Symbol	Qty	Total cost	Transaction

Sending messages from server: Plants2

Select number of messages to be sent for each of the **Buy** and **Sell** categories, then click **Send messages**.

Account	Buy/Sell	Symbol	Qty	Total cost	Transaction

- Click **Send messages** button on Plants1 and Plants2

© Copyright IBM Corporation 2013

Figure 11-23. Monitor web page (1 of 2)

WA680 / VA6801.0

Notes:

A monitor web page is used to show the controls for sending messages to each cluster member. Also shown, is a table of messages (if any) that are being processed by the consumer application.

Monitor web page (2 of 2)

- Server Plants1 gets all of the messages. Why?

The screenshot shows two panels of the WebSphere Monitor web page. The left panel, titled 'Sending messages from server: Plants1', contains instructions to select numbers of messages for Buy and Sell categories and click 'Send messages'. It has input fields for 'Buy messages' (set to 1) and 'Sell messages' (set to 1), along with a 'Send messages' button. The right panel, titled 'Trade Requests To Be Processed On Server: Plants1', displays a table of trade requests. The table has columns: Account, Buy/Sell, Symbol, Qty, Total cost, and Transaction. The data is as follows:

Account	Buy/Sell	Symbol	Qty	Total cost	Transaction
23423234 (John Doe)	BUY	PG	10.0	\$657.90	Plants1 - 5
87652289 (Elaine Moose)	SELL	DELL	1.0	\$19.89	Plants1 - 6
23423234 (John Doe)	BUY	PG	10.0	\$657.90	Plants2 - 3
87652289 (Elaine Moose)	SELL	DELL	1.0	\$19.89	Plants2 - 4

The right panel, titled 'Trade Requests To Be Processed On Server: Plants2', is currently empty.

© Copyright IBM Corporation 2013

Figure 11-24. Monitor web page (2 of 2)

WA680 / VA6801.0

Notes:

In this example screen capture, each cluster member sends one buy and one sell message. The messages are queued on the server that is currently hosting the messaging engine. In this example, the cluster member Plants1 is hosting the messaging engine.

Unit summary

Having completed this unit, you should be able to:

- Describe the components and functions of the PlantsByWebSphere application

© Copyright IBM Corporation 2013

Figure 11-25. Unit summary

WA680 / VA6801.0

Notes:

Unit 12. Configuring and managing servers and server resources by using scripting

What this unit is about

This unit describes how to use the Administrative objects to configure and manage servers and their resources.

What you should be able to do

After completing this unit, you should be able to:

- Describe how to perform common server and server resource management tasks by using the administrative objects

Unit objectives

After completing this unit, you should be able to:

- Describe how to perform common server and server resource management tasks by using the administrative objects

© Copyright IBM Corporation 2013

Figure 12-1. Unit objectives

WA680 / VA6801.0

Notes:

Configuring servers

- Use AdminConfig to perform configuration changes such as:
 - Saving or resetting configuration changes
- Use AdminTask to:
 - Create application server
 - Create clusters and cluster members
 - Modify JVM settings
- Use AdminControl to:
 - Stop and start applications servers
 - Stop node agents
 - Start, stop, and ripple start clusters

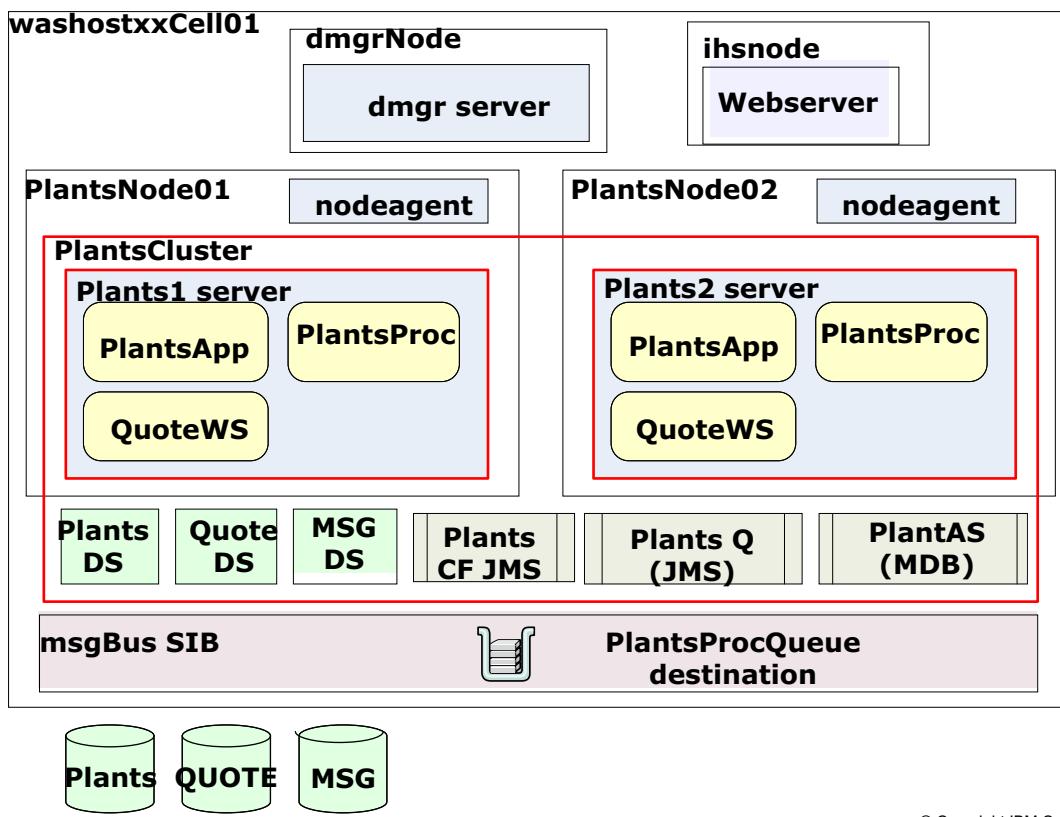
© Copyright IBM Corporation 2013

Figure 12-2. Configuring servers

WA680 / VA6801.0

Notes:

Configuring application servers, clusters, and cluster members



© Copyright IBM Corporation 2013

Figure 12-3. Configuring application servers, clusters, and cluster members

WA680 / VA6801.0

Notes:

AdminTask: ServerManagement command group

- The ServerManagement command group for the AdminTask object includes the following methods:
 - createApplicationServerTemplate
 - createApplicationServer
 - createGenericServer
 - createWebServer
 - deleteServer
 - deleteServerTemplate
 - getServerType
 - listServer
 - listServerTemplates
 - listServerTypes
 - showServerInfo
 - showServerTypeInfo
 - showTempalteInfo

© Copyright IBM Corporation 2013

Figure 12-4. AdminTask: ServerManagement command group

WA680 / VA6801.0

Notes:

Creating an application server template

- `AdminTask.createApplicationServerTemplate()`
 - Creates a template from which to create a server.
 - *templateName* is a string that contains the name of the application server template.
 - *serverName* is a string that contains the name of the server from which to base the template.
 - *nodename* is a string that contains the node that corresponds to the server from which to base the template.
 - *description* is an optional string that contains the description of the template.
 - *templateLocation* is an optional string that contains the configuration ID that represents the location to place the template.

© Copyright IBM Corporation 2013

Figure 12-5. Creating an application server template

WA680 / VA6801.0

Notes:

Example: Creating an application server template

```
AdminTask.createApplicationServerTemplate('[-templateName PlantsTemplate1
-serverName Plants1 -nodeName Node01 -description "This is the Plants
template" -templateLocation Websphere:_Websphere_Config_Data_Display_Name=
APPLICATION_SERVER,_Websphere_Config_Data_Id=templates/servertypes/APPLICA
TION_SERVER|servertype-metadata.xml']')
'PlantsTemplate1(templates/servertypes/APPLICATION_SERVER/servers/PlantsTe
mplate1|server.xml#Server_1183652542187)'

AdminConfig.save() ← Save the configuration changes
```

Create a server template that is based on the Plants1 application server

© Copyright IBM Corporation 2013

Figure 12-6. Example: Creating an application server template

WA680 / VA6801.0

Notes:

Creating an application server

- `AdminTask.createApplicationServer()`
 - Used to create an application server.
 - *nodeName* is a string that contains the name of the node.
 - *name* is a string that contains the name of the server.
 - *templateName* is an optional string that contains the name of the template from which to base the server.
 - *genUniquePorts* is an optional Boolean that specifies that unique ports should be created for the server.
 - *templateLocation* is an optional string that contains the location of a template.

© Copyright IBM Corporation 2013

Figure 12-7. Creating an application server

WA680 / VA6801.0

Notes:

Example: Creating an application server

```
node = 'Node01'           ← Specify the node name

AdminTask.createApplicationServer(node, "[ -name test1 -genUniquePorts true
-templateName default ]")

'test1(cells/washost00Cell01/nodes/Node01/servers/test1|server.xml#Server
_1183989701656)'

AdminConfig.save()          ← Save the configuration changes
```

© Copyright IBM Corporation 2013

Figure 12-8. Example: Creating an application server

WA680 / VA6801.0

Notes:



AdminTask: ClusterConfigCommands command group

The ClusterConfigCommands command group for the AdminTask object includes the following methods:

- createCluster
- createClusterMember
- deleteCluster
- deleteClusterMember

© Copyright IBM Corporation 2013

Figure 12-9. AdminTask: ClusterConfigCommands command group

WA680 / VA6801.0

Notes:

Creating a cluster

- *AdminTask.createCluster()*
 - Creates an application server cluster.
 - *clusterConfig* is the step that specifies the configuration of the new server cluster.
 - *replicationDomain* is the step that specifies the configuration of a replication domain for the cluster.
 - *convertServer* is the step that specifies an existing server to convert to the first member of the cluster.
 - *eventServiceConfig* is the step that specifies the event service configuration of the new server cluster.
 - *promoteProxyServer* is the step that if a proxy server was specified for *convertServer*, apply its proxy settings to the cluster.

© Copyright IBM Corporation 2013

Figure 12-10. Creating a cluster

WA680 / VA6801.0

Notes:

Example: Creating a cluster

```
AdminTask.createCluster(['-clusterConfig', [['PlantsCluster', 'true']], '-replicationDomain', [['true']]])  
↑  
Specifies the prefer local option  
  
'PlantsCluster(cells/washost00Cell01/clusters/PlantsCluster|cluster.xml#ServerCluster_1183993670031)'  
  
AdminConfig.save() ← Save the configuration changes
```

Specifies the creation of the replication domain

Specify the cluster name

© Copyright IBM Corporation 2013

Figure 12-11. Example: Creating a cluster

WA680 / VA6801.0

Notes:

Creating cluster members

- `AdminTask.createClusterMember()`
 - Creates a server cluster member.
 - `clusterName` is the string that contains the name of the server cluster to which the new cluster member belongs.
 - `memberConfig` is the step that specifies the configuration of a new member of the cluster.
 - `firstMember` is the step that specifies additional information that is required to configure the first member of a cluster.
 - `eventServiceConfig` is the step that specifies the event service configuration of a new member of the cluster.
 - `promoteProxyServer` is the step in which, if a proxy server was specified and no other servers exist in the cluster, the proxy settings are applied to the cluster.

© Copyright IBM Corporation 2013

Figure 12-12. Creating cluster members

WA680 / VA6801.0

Notes:

Example: Creating cluster members

```

nodeName='Node01'
clusterName='PlantsCluster'
memberName='Plants1'
serverWeight='2'
memberUUID=''
generateUniquePorts='true'
replicatorEntry='false'
```

Cluster member property variables and values

Create the cluster member with the specified properties



```
AdminTask.createClusterMember(['-clusterName', clusterName, '-memberConfig', [nodeName, memberName, nodegroupName, serverWeght, generateUniquePorts, replicatorEntry]])
```

```
'Plants1(cells/washost00Cell01/clusters/PlantsCluster|cluster.xml#ClusterMember_1183996552062)'
```

```
AdminConfig.save()
```

Save the configuration changes

© Copyright IBM Corporation 2013

Figure 12-13. Example: Creating cluster members

WA680 / VA6801.0

Notes:

Modifying JVM properties

- Application server and cluster members, being a Java process, requires a JVM to run, and to support the Java applications that are running.
- Use the following AdminTask methods from the ServerManagement command group to modify JVM settings:
 - setJVMDebugMode()
 - setGenericJVMArguments()
 - setJVMInitialHeapSize()
 - setJVMMaxHeapSize()
 - setJVMMode()
 - setJVMProperties()
 - setJVMSystemProperties()

© Copyright IBM Corporation 2013

Figure 12-14. Modifying JVM properties

WA680 / VA6801.0

Notes:

Setting JVM properties

- `AdminTask.setJVMProperties()`
 - Sets the specified JVM properties.
 - `serverName` is the string that contains the name of the server for which the JVM properties to be modified.
 - `nodeName` is the string that contains the node name where the server runs.
- Optional parameters, which are listed in the following reference tables, can be specified following the `serverName` and `nodeName` arguments.

© Copyright IBM Corporation 2013

Figure 12-15. Setting JVM properties

WA680 / VA6801.0

Notes:

JVM property reference (1 of 3)

JVM property	Property
classpath	The standard class path in which the Java virtual machine (JVM) code looks for classes.
bootClasspath	Bootstrap classes and resources for JVM code. This option is only available for JVM instructions that support bootstrap classes and resources. You can separate multiple paths by a colon (:) or semi-colon (;), depending on the operating system of the node.
verboseModeClass	Specifies whether to use verbose debug output for class loading. The default is not to enable verbose class loading.
verboseModeGarbageCollection	Specifies whether to use verbose debug output for garbage collection.
verboseModeJNI	Specifies whether to use verbose debug output for native method invocation. The default is not to enable verbose Java Native Interface (JNI) activity.
maximumHeapSize	Specifies the maximum heap size available in megabytes to the JVM code.

© Copyright IBM Corporation 2013

Figure 12-16. JVM property reference (1 of 3)

WA680 / VA6801.0

Notes:

JVM property reference (2 of 3)

JVM property	Property
runHProf	This parameter applies to WebSphere Application Server version. It specifies whether to use HProf profiler support. To use another profiler, specify the custom profiler settings with the hprofArguments parameter. The default is not to enable HProf profiler support.
hprofArguments	This parameter applies to WebSphere Application Server version. It specifies command-line profiler arguments to pass to the JVM code that starts the application server process. You can specify arguments when HProf profiler support is enabled.
debugMode	Specifies whether to run the JVM in debug mode. The default is not to enable debug mode support.
debugArgs	Specifies the command-line debug arguments to pass to the JVM code that starts the application server process. You can specify arguments when the debug mode is enabled.
genericJvmArguments	Specifies the command-line arguments to pass to the JVM code that starts the application server process.

© Copyright IBM Corporation 2013

Figure 12-17. JVM property reference (2 of 3)

WA680 / VA6801.0

Notes:

JVM property reference (3 of 3)

JVM property	Property
<code>executableJarFileName</code>	Specifies a full path name for an executable JAR file that the JVM code uses.
<code>disableJIT</code>	Specifies whether to disable the just in time (JIT) compiler option of the JVM code.
<code>osName</code>	Specifies the JVM settings for an operating system. When started, the process uses the JVM settings for the operating system of the node.
<code>initialHeapSize</code>	Specifies the initial heap size in megabytes that is available to the JVM code.

© Copyright IBM Corporation 2013

Figure 12-18. JVM property reference (3 of 3)

WA680 / VA6801.0

Notes:

Example: Setting JVM properties

```
serverName='Plants1'
nodeName='Node01'
```

Use the `setJVMProperties()` method in batch mode to set the specified property

```
AdminTask.setJVMProperties(['-serverName', serverName, '-nodeName',
    nodeName, '-initialHeapSize','512'])
```

```
→'true'
```

```
AdminTask.setJVMProperties(['-serverName', serverName, '-nodeName',
    nodeName, '-maximumHeapSize','1028', '-  
verboseModeGarbageCollection','false'])
```

Use the `setJVMProperties()` method in batch mode to set multiple properties

```
→'true'
```

The value 'true' is returned upon execution
of the `setJVMProperties()` method

```
AdminConfig.save()
```

Save the configuration changes

© Copyright IBM Corporation 2013

Figure 12-19. Example: Setting JVM properties

WA680 / VA6801.0

Notes:

Managing servers: Starting and stopping server

- AdminControl.startServer()
 - Starts an application server.
- AdminControl.stopServer()
 - Stops an application server.
 - *serverName* is the string that contains the name of the server to start.
 - *nodeName* is the string that contains the node name where the server runs.

```
AdminControl.startServer('Plants1','Node01')           ← Starts the
'WASX7262I: Start completed for server "Plants1" on node   application server
"Node01"'
```



```
AdminControl.stopServer('Plants1','Node01')            ← Stops the
'WASX7264I: Stop completed for server "Plants1" on node "Node01"'   application server
```

© Copyright IBM Corporation 2013

Figure 12-20. Managing servers: Starting and stopping server

WA680 / VA6801.0

Notes:

Managing servers: Stopping node agents

1. Get object name of the *NodeAgent* MBean with AdminControl
2. Use *AdminControl.invoke()* method to start the operation:
 - 惣 stopNode stops the node agent

```
na = AdminControl.queryNames('type=NodeAgent, node=Node01, *')  
'WebSphere:name=NodeAgent,process=nodeagent,platform=common,node=No  
de01,diagnosticProvider=true,version=6.1.0.0,type=NodeAgent,mbeanId  
entifier=NodeAgent,cell=washost00Cell01,spec=1.0'  
  
AdminControl.invoke(na, 'stopNode') ← Stops the node agent  
  
..
```

© Copyright IBM Corporation 2013

Figure 12-21. Managing servers: Stopping node agents

WA680 / VA6801.0

Notes:

Managing clusters: Starting clusters

1. Identify the Cluster MBean and assign it to the cluster variable

2. Use *AdminControl.invoke()* method to start the operation:
 - `start` starts the cluster
 - `rippleStart` ripple starts the cluster

```

cluster =
AdminControl.completeObjectName ('type=Cluster, name=
PlantsCluster, *')
          ↑
          Retrieves the object name of the cluster

AdminControl.invoke(cluster, 'start')
          ↑
          Start the cluster

AdminControl.invoke(cluster, 'rippleStart')
          ↑
          Ripple start the
          cluster
    
```

© Copyright IBM Corporation 2013

Figure 12-22. Managing clusters: Starting clusters

WA680 / VA6801.0

Notes:

Managing clusters: Stopping clusters

1. Identify the Cluster MBean and assign it to the cluster variable
2. Use *AdminControl.invoke()* method to start the operation:
 - stop stops the cluster

```
cluster =  
AdminControl.completeObjectName('type=Cluster,name='  
PlantsCluster,'*')  
  
AdminControl.invoke(cluster, 'stop')  
''  
  
Retrieves the object name of the cluster  
Stop the cluster
```

© Copyright IBM Corporation 2013

Figure 12-23. Managing clusters: Stopping clusters

WA680 / VA6801.0

Notes:



Managing clusters: Query cluster state

1. Identify the Cluster MBean and assign it to the cluster variable
 2. Use `AdminControl.getAttribute()` method to retrieve the `state` attribute
 - If the cluster is started, `websphere.cluster.running` is displayed.
 - If the cluster is stopped, `websphere.cluster.stopped` is displayed.

```
cluster =  
AdminControl.completeObjectName('type=Cluster, name='  
PlantsCluster, '*')  
  
AdminControl.getAttribute(cluster, 'state')  
  
'websphere.cluster.stopped'  
  
Retrieves the object name of the cluster  
Retrieve the
```

© Copyright IBM Corporation 2013

Figure 12-24 Managing clusters: Query cluster state

WA680 / VA6801.0

Notes:

Security: Determining whether security is enabled or disabled

- AdminTask.isAppSecurityEnabled()
 - Used to determine whether application security is enabled
 - 'true' is returned if application security is enabled
 - 'false' is returned if application security is disabled
- AdminTask.isGlobalSecurityEnabled()
 - Used to determine whether administrative security is enabled
 - 'true' is returned if administrative security is enabled
 - 'false' is returned if administrative security is disabled

```
print AdminTask.isAppSecurityEnabled() ← Determine whether  
false  
application security is  
enabled  
  
print AdminTask.isGlobalSecurityEnabled() ← Determine whether  
true  
administrative  
security is enabled
```

© Copyright IBM Corporation 2013

Figure 12-25. Security: Determining whether security is enabled or disabled

WA680 / VA6801.0

Notes:

Security: Enabling and disabling administrative security

- AdminTask.setGlobalSecurity()
 - The administrative security field in the security.xml file is updated based on the user input of true or false.
 - *enabled* is the Boolean parameter that contains the value that specifies whether to enable or disable global security.

```
AdminTask.setGlobalSecurity ('[-enabled true]')  
'true'
```

```
AdminConfig.save()
```



Enables administrative security

© Copyright IBM Corporation 2013

Figure 12-26. Security: Enabling and disabling administrative security

WA680 / VA6801.0

Notes:

Security: Enable and disable Java 2 security

1. Identify the security configuration MBean and assign it to the security variable
2. Use *AdminConfig.modify()* method to start the operation:
 - 'enforceJava2Security', 'true' enables Java 2 security
 - 'enforceJava2Security', 'false' disables Java 2 security

```
security = AdminConfig.list('Security')

AdminConfig.modify(security, [ ['enforceJava2Security',
    'false']])
''

AdminConfig.save()
```



© Copyright IBM Corporation 2013

Figure 12-27. Security: Enable and disable Java 2 security

WA680 / VA6801.0

Notes:

AdminTask: AuthorizationGroupCommands command group

- The AuthorizationGroupCommands command group for the AdminTask object includes the following methods:
 - addResourceToAuthorizationGroup ()
 - createAuthorizationGroup ()
 - deleteAuthorizationGroup ()
 - listAuthorizationGroups ()
 - listAuthorizationGroupsForGroupID ()
 - listAuthorizationGroupsForUserID ()
 - listAuthorizationGroupsOfResource ()
 - listResourcesOfAuthorizationGroup ()
 - listResourcesForGroupID ()
 - listResourcesForUserID ()
 - mapGroupsToAdminRole ()
 - mapUsersToAdminRole ()
 - removeGroupsFromAdminRole ()
 - removeResourceFromAuthorizationGroup ()
 - removeUsersFromAdminRole ()

© Copyright IBM Corporation 2013

Figure 12-28. AdminTask: AuthorizationGroupCommands command group

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe how to perform common server and server resource management tasks by using the administrative objects

© Copyright IBM Corporation 2013

Figure 12-29. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. What are the AdminTask methods used to create clusters and cluster members?
2. How do you query the running status of the cluster?
3. When the AdminTask.setGlobalSecurity() method is started, which configuration file is updated?

© Copyright IBM Corporation 2013

Figure 12-30. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

Checkpoint answers

1. What are the AdminTask methods used to create clusters and cluster members?

Answer: AdminTask.createCluster() and
AdminTask.createClusterMember()

2. How do you query the running status of the cluster?

Answer: Use AdminControl.getAttribute() method to retrieve the state attribute

3. When the AdminTask.setGlobalSecurity() method is started, which configuration file is updated?

Answer: The configuration file that is updated is security.xml.

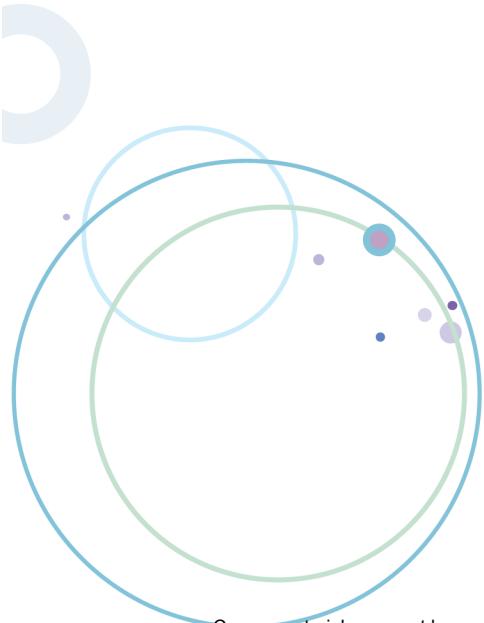
© Copyright IBM Corporation 2013

Figure 12-31. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 7



Creating and configuring the Plants server environment with scripting

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 12-32. Exercise 7

WA680 / VA6801.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Create a deployment manager and a managed node profile by using a response file
- Develop Jython scripts to create a cluster and add cluster members
- Develop Jython scripts to query the state of a cluster, and start or stop a cluster

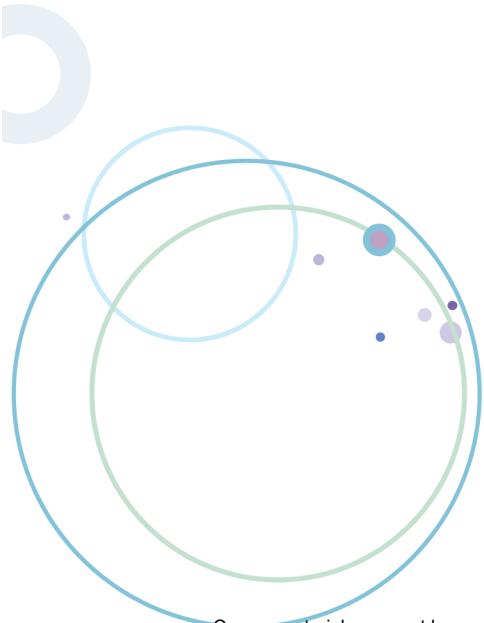
© Copyright IBM Corporation 2013

Figure 12-33. Exercise objectives

WA680 / VA6801.0

Notes:

Exercise 8



Installing and configuring the IBM HTTP Server

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 12-34. Exercise 8

WA680 / VA6801.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Install the IBM HTTP Server and its plug-in by using a silent installation
- Develop Jython scripts to configure the IBM HTTP Server as an unmanaged node

© Copyright IBM Corporation 2013

Figure 12-35. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 13. Deploying and managing the PlantsByWebSphere application by using scripting

What this unit is about

This unit describes how to use the Administrative objects to deploy and manage applications and their resources.

What you should be able to do

After completing this unit, you should be able to:

- Describe how to perform common application and application resource management tasks by using the administrative objects

Unit objectives

After completing this unit, you should be able to:

- Describe how to perform common application and application resource management tasks by using the administrative objects

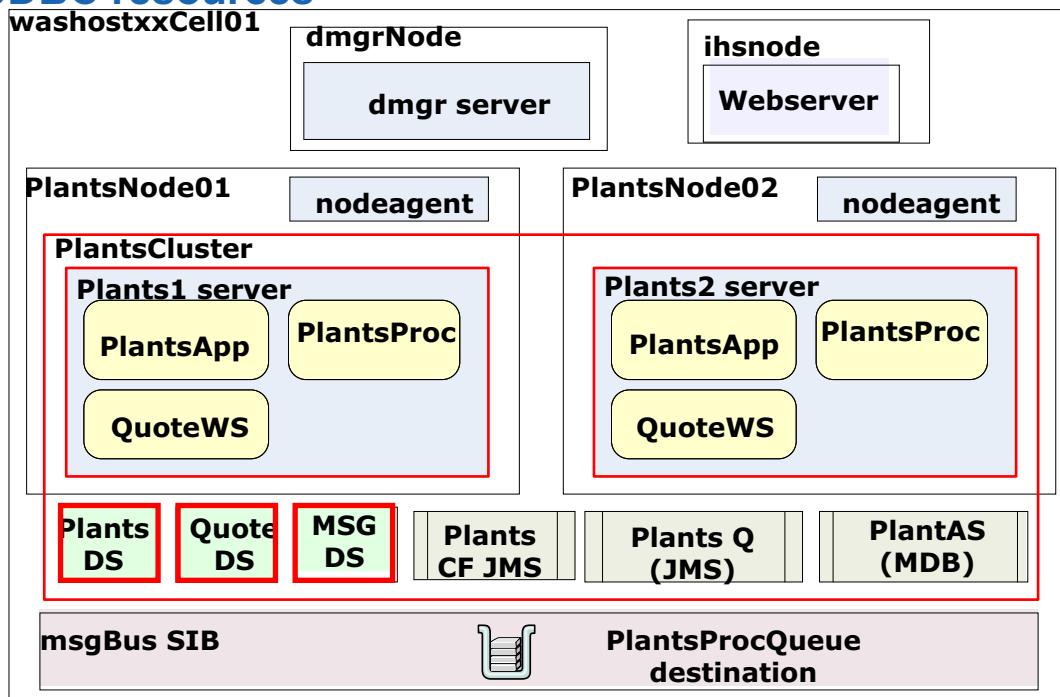
© Copyright IBM Corporation 2013

Figure 13-1. Unit objectives

WA680 / VA6801.0

Notes:

JDBC resources



© Copyright IBM Corporation 2013

Figure 13-2. JDBC resources

WA680 / VA6801.0

Notes:



AdminTask: JDBCProviderManagement command group

- The JDBCProviderManagement command group for the AdminTask object includes the following methods:
 - createDatasource
 - createJDBCProvider
 - listDatasources
 - listJDBCProviders

© Copyright IBM Corporation 2013

Figure 13-3. AdminTask: JDBCProviderManagement command group

WA680 / VA6801.0

Notes:

Create JDBC providers

- `AdminTask.createJDBCProvider`
 - Used to create a JDBCProvider.
 - `scope` is a string that contains the scope for the new JDBC provider.
 - `databaseType` is a string that contains the name of the type of database that is used by the JDBC provider.
 - `providerType` is a string that contains the JDBC provider type that is used by the JDBC provider.
 - `implementationType` is a string that contains the implementation type for this JDBC provider. Use Connection pool data source if your application runs in a single phase or a local transaction. Otherwise, use XA data source to run in a global transaction.
 - `name` is an optional string that contains the name of the JDBC provider.
 - `description` is an optional string that contains the description for the JDBC provider.
 - `implementationClassName` is an optional string that contains the Java class name for the JDBC driver implementation.
 - `classpath` is an optional string that contains the list of paths or JAR file names that form the location for the resource provider classes.
 - `nativePath` is an optional string that contains the list of paths that form the location for the resource provider native libraries.

© Copyright IBM Corporation 2013

Figure 13-4. Create JDBC providers

WA680 / VA6801.0

Notes:



Example: Creating JDBC providers

```
AdminTask.createJDBCProvider(['-scope','Cell=washost00Cell101',
'-databaseType','DB2','-providerType','DB2 Universal JDBC Driver
Provider', '-implementationType', 'XA data source', '-name','DB2 Universal
JDBC Driver Provider (XA)', '-description', 'XA DB2 Universal JDBC Driver-
compliant Provider.', '-classpath', '${DB2UNIVERSAL_JDBC_DRIVER_PATH}/
db2jcc.jar;${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cu.jar;
${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license_cisuz.jar', '-nativePath',
'${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}')
```

```
'''DB2 Universal JDBC Driver Provider (XA) (cells/washost00Cell01|resources.xml#JDBCProvider 1184035713984)'''
```

`AdminConfig.save()` Save the configuration changes

© Copyright IBM Corporation 2013

Figure 13-5. Example: Creating JDBC providers

WA680 / VA6801.0

Notes:

Create data sources

- `AdminTask.createDatasource()`
 - Used to create a data source.
 - *name* is a string that contains the name of the data source.
 - *jndiName* is a string that contains the data source JNDI name.
 - *dataStoreHelperClassName* is a string that contains the name of the DataStoreHelper implementation class that extends the capabilities of the selected JDBC driver implementation class to perform data-specific functions.
 - *description* is a string that contains the description of the data source.
 - *category* is an optional string that contains the category that you can use to classify a group of data sources.
 - *componentManagedAuthenticationAlias* is a string that contains the alias that is used for database authentication at run time.
 - *containerManagedPersistence* is a Boolean specifying whether the data source is used for container managed persistence for enterprise beans.
 - *configureResourceProperties* is a step configures the resource properties that the data source requires.

© Copyright IBM Corporation 2013

Figure 13-6. Create data sources

WA680 / VA6801.0

Notes:

Example: Creating data sources

```

AdminTask.createDatasource(                                     JDBC Provider
'DB2 Universal JDBC Driver Provider(XA) (cell
s/washost00Cell101|resources.xml#JDBCProvider_1184035713984)',

[ '-name', 'DB2 Universal JDBC Driver XA DataSource', '-jndiName',
'jdbc/Plantsds', '-dataStoreHelperClassName',
'com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper', '-
componentManagedAuthenticationAlias', 'myCellManager01/PlantsApp', '-
xaRecoveryAuthAlias', 'myCellManager01/PlantsApp', '-
configureResourceProperties', '[ [databaseName java.lang.String Plants 1]
[driverType java.lang.Integer 4] [serverName java.lang.String dbserver
1] [portNumber java.lang.Integer 50000] ]' ])

                                         Create the data source

'"DB2 Universal JDBC Driver XA DataSource(cells/washost00Cell101|
resources.xml#DataSource_1184036585671)"'

AdminConfig.save()                                     Save the configuration changes

```

© Copyright IBM Corporation 2013

Figure 13-7. Example: Creating data sources

WA680 / VA6801.0

Notes:

Modifying WebSphere variables

1. Get a list of the WebSphere variables using the `AdminConfig.list()` method and locate `VariableSubstitutionEntry` entries

2. Use `AdminControl.modify()` method to update the specified environment variable

© Copyright IBM Corporation 2013

Figure 13-8. Modifying WebSphere variables

WA680 / VA6801.0

Notes:

Example: modifying WebSphere variables

```

varName = "DB2_JDBC_DRIVER_PATH"           ← Specify the variable to change
newValue = "C:/SQLLIB/java"                ← Specify the new value

node = AdminConfig.getid("/Node:Node01/")    ← Specify the scope of the variable

varSubstitutions =
AdminConfig.list("VariableSubstitutionEntry", node).split(linesep)   ← Retrieve the list of VariableSubstitutionEntry entries

for varSubst in varSubstitutions:
    getVarName = AdminConfig.showAttribute(varSubst, "symbolicName")
    if getVarName == varName:
        AdminConfig.modify(varSubst, [{"value": newValue}])
        break
    ↑          Iterate through the list of entries. Next, modify the
              variable to the new variable value

AdminConfig.save()                         ← Save the configuration changes

```

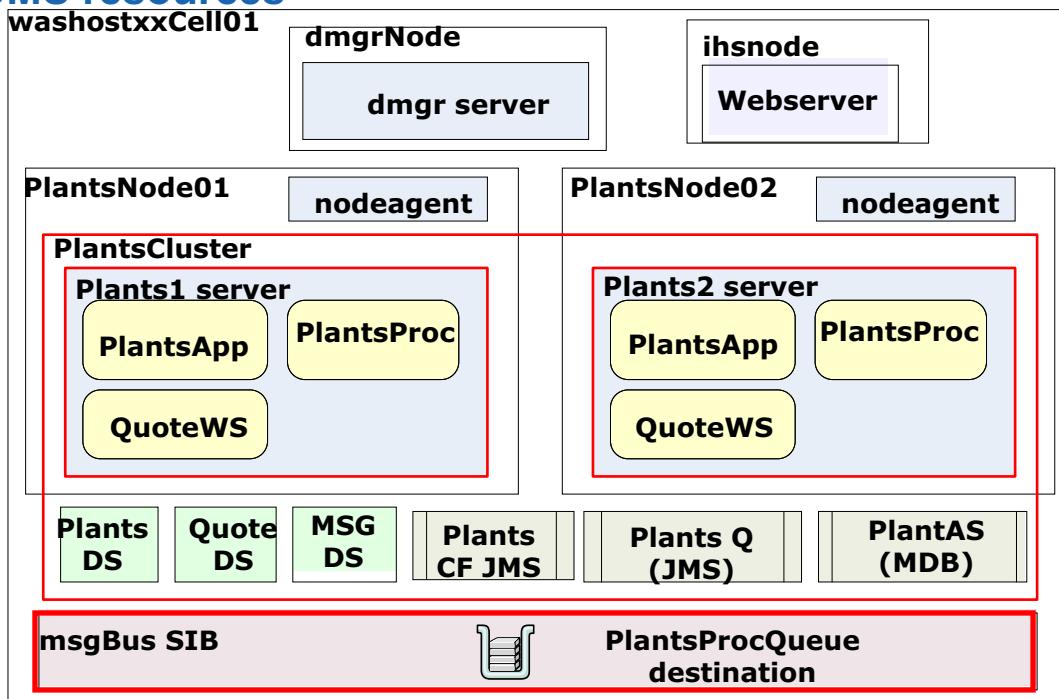
© Copyright IBM Corporation 2013

Figure 13-9. Example: modifying WebSphere variables

WA680 / VA6801.0

Notes:

JMS resources



© Copyright IBM Corporation 2013

Figure 13-10. JMS resources

WA680 / VA6801.0

Notes:

Creating a service integration bus

- `AdminTask.createSIBus()`
 - Creates a service integration bus.
 - *busName* is a string that contains the name of the SIBus.
 - *description* is an optional string that contains the description for the bus.
 - *busSecurity* is an optional Boolean specifying whether authorization policy for the bus is enforced. This option requires administrative security to be enabled.
 - *mediationsAuthAlias* is a string that contains the name of the authentication alias that is used to authorize mediations to access the bus.
 - *protocol* is a string that contains the name of the protocol that is used for communication between messaging engines on this bus.
 - *discardOnDelete* is a Boolean specifying whether messages are deleted if a message point is deleted.

© Copyright IBM Corporation 2013

Figure 13-11. Creating a service integration bus

WA680 / VA6801.0

Notes:

Adding service integration bus members

- `AdminTask.addSIBusMember()`
 - Adds a bus member to the specified service integration bus.
 - *busName* is a string that contains the name by which the service integration bus is known.
 - *wmqServer* is a string that contains the name of the WebSphere MQ server to add to the bus.
 - *host* is a string that contains the overridden value for the WebSphere MQ server bus member host attribute.
 - *port* is a string that contains the overridden value for the WebSphere MQ server bus member port attribute.
 - *channel* is a string that contains the value for the WebSphere MQ server bus member channel attribute.
 - *securityAuthAlias* is a string that contains the value for the WebSphere MQ server bus member securityAuthAlias attribute.
 - *transportChain* is a string that contains the value for the WebSphere MQ server bus member transportChain attribute.
 - *trustUserIds* is a Boolean that specifies whether the user identifiers that are received in messages from WebSphere MQ are propagated into messages or not.

© Copyright IBM Corporation 2013

Figure 13-12. Adding service integration bus members

WA680 / VA6801.0

Notes:

Bus destinations

- AdminTask.createSIBDestination()
 - Creates an SIB bus destination.
 - *bus* is a string that contains the name of the service integration bus on which to create the bus destination.
 - *name* is a string that contains the identifier by which this destination is known for administrative purposes.
 - *type* is a string that contains the type of bus destination that you want to create.
 - *wmqServer* is the string that specifies the name of the WebSphere MQ server bus member where the destination is assigned.
 - *wmqQueueName* is the string that specifies the name of the WebSphere MQ queue that is used to store messages that are sent to the destination.

© Copyright IBM Corporation 2013

Figure 13-13. Bus destinations

WA680 / VA6801.0

Notes:

Example: creating an SIBus, bus member, and JMS destinations

```

AdminTask.createSIBus('[-bus msgBus -description "Message bus" -
busSecurity true]')
    ← Create the SIB bus

'msgBus(cells/washost00Cell01/buses/msgBus|sib-
bus.xml#SIBus_1184077610906)'

AdminTask.addSIBusMember('[-bus msgBus -cluster PlantsCluster -
datasourceJndiName "jdbc/mesengds"]')
    "
        ↑
        Add the cluster as the SIBus
        member

AdminTask.createSIBDestination('[-bus msgBus -name PlantsProcQueue -type
Queue -cluster PlantsCluster]')
    ← Create a queue destination

'(cells/washost00Cell01/buses/msgBus|sib-destinations.xml
#SIBQueue_1184082745515)'

AdminConfig.save()    ← Save the configuration changes

```

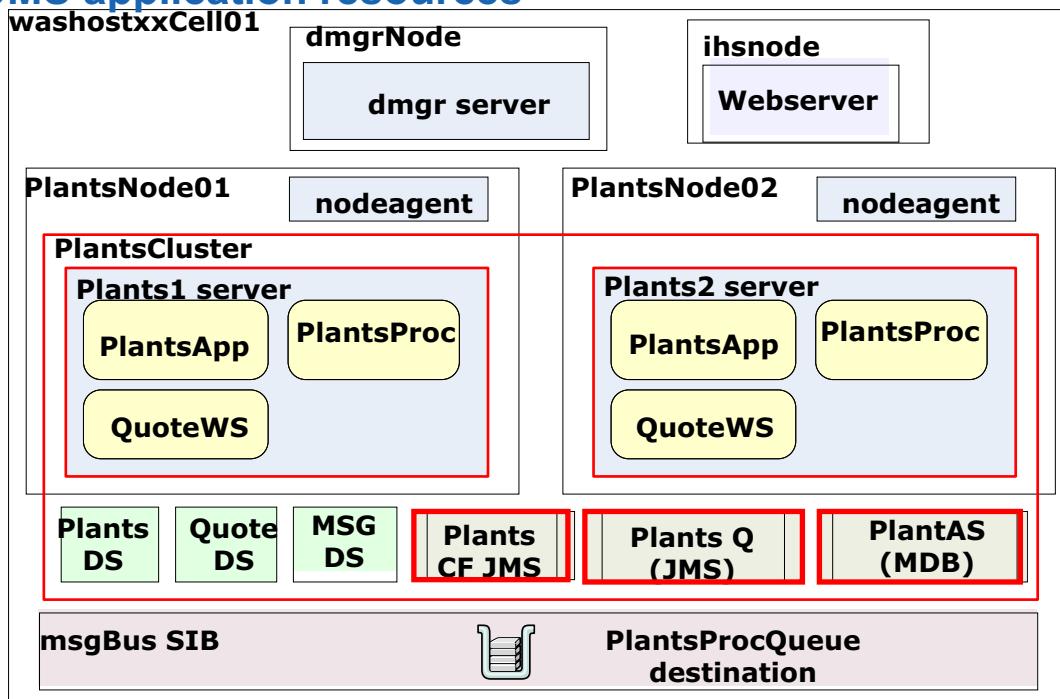
© Copyright IBM Corporation 2013

Figure 13-14. Example: creating an SIBus, bus member, and JMS destinations

WA680 / VA6801.0

Notes:

JMS application resources



© Copyright IBM Corporation 2013

Figure 13-15. JMS application resources

WA680 / VA6801.0

Notes:

JMS connection factory

- `AdminTask.createSIBJMSCConnectionFactory()`
 - Creates a JMS connection factory.
 - *name* is a string that contains the name of the JMS connection factory.
 - *jndiName* is a string that contains the JNDI name of the JMS connection factory
 - *busName* is a string that contains the name of the SIBus.
 - *type* is a string that contains the type of destination, either *queue* or *topic*.
 - *authDataAlias* is a string that contains the name of the J2C authentication alias.
 - *xaRecoveryAuthAlias* is a string that contains the name of the J2C authentication alias for two-phase commit data sources.
 - *description* is a string that contains the description of the JMS connection factory.

© Copyright IBM Corporation 2013

Figure 13-16. JMS connection factory

WA680 / VA6801.0

Notes:

Example: creating JMS connection factory

```
cn=AdminConfig.getid('/ServerCluster:PlantsCluster/')

AdminTask.createSIBJMSSConnectionFactory(cn, '[ -name PlantsCF -jndiName
"jms/PlantsJMSCF" -busName msgBus ]')

PlantsCF(cells/washost00Cell01/clusters/PlantsCluster|resources.xml#J2CCon
nectionFactory_1184083869750)

AdminConfig.save()
```

Retrieve the configuration id for the cluster

Create the JMS connection factory

Save the configuration changes

© Copyright IBM Corporation 2013

Figure 13-17. Example: creating JMS connection factory

WA680 / VA6801.0

Notes:

JMS queue destination

- `AdminTask.createSIBJMSQueue()`
 - Used to create a JMS queue destination.
 - *name* is a string that contains the name of the SIB JMS queue.
 - *jndiName* is a string that contains the JNDI name for the SIB JMS queue.
 - *queueName* is a string that contains the name of the underlying SIB queue to which the queue points.
 - *description* is an optional string that contains the description of the JMS queue.
 - *deliveryMode* is an optional string that contains the delivery mode for messages.
 - *timeToLive* is an optional integer value that represents the time in milliseconds to be used for message expiration.
 - *priority* is an optional whole number 0 - 9 representing the message priority.
 - *readAhead* is an optional string that contains the read-ahead value.
 - *busName* is an optional string that contains the name of the bus on which the queue exists.

© Copyright IBM Corporation 2013

Figure 13-18. JMS queue destination

WA680 / VA6801.0

Notes:

Example: creating JMS queue

```
cn=AdminConfig.getid('/ServerCluster:PlantsCluster/')

AdminTask.createSIBJMSQueue(cn, ['-name "Plants Processor Queue" -jndiName
"jms/PlantsProcJMSQueue" -queueName PlantsProcQueue -busName msgBus'])

'"Plants Processor Queue(cells/washost00Cell01/clusters/
PlantsCluster|resources.xml#J2CAdminObject_1184087982937)"'

AdminConfig.save()
```

Retrieve the configuration id for the cluster

Create the JMS Queue

Save the configuration changes

© Copyright IBM Corporation 2013

Figure 13-19. Example: creating JMS queue

WA680 / VA6801.0

Notes:

JMS activation specification

- `AdminTask.createSIBJMSActivationSpec()`
 - Used to create a JMS activation specification.
 - *name* is a string that contains the name of the activation specification.
 - *jndiName* is a string that contains the JNDI name of the activation specification.
 - *destinationJndiName* is a string that contains the JNDI name of the JMS destination.
 - *busName* is a string that contains the name of the SIBus.
 - *description* is an optional string that contains the description of the activation specification.
 - *acknowledgeMode* is an optional string that contains the method that session acknowledges any messages it receives.
 - *authenticationAlias* is a string that contains the authentication alias.
 - *clientId* is a string that contains the client identifier that is required for durable topic subscriptions.
 - *destinationType* is a string that specifies whether the message-driven bean uses a queue or topic destination.
 - *durableSubscriptionHome* is a string that contains the name of the durable subscription home.

© Copyright IBM Corporation 2013

Figure 13-20. JMS activation specification

WA680 / VA6801.0

Notes:

Example: creating JMS activation specification

```
cn=AdminConfig.getid('/ServerCluster:PlantsCluster/')

AdminTask.createSIBJMSActivationSpec (cn, '[-name PlantsProcessorAS - 
jndiName "eis/PlantsProcAS2" -destinationJndiName "jms/PlantsProcJMSQueue" 
-busName msgBus]')

'PlantsProcessorAS(cells/washost00Cell01/clusters/PlantsCluster|resources.
xml#J2CActivationSpec_1184090317750)'

AdminConfig.save()
```

Retrieve the configuration id for the cluster

Create the JMS activation specification

Save the configuration changes

© Copyright IBM Corporation 2013

Figure 13-21. Example: creating JMS activation specification

WA680 / VA6801.0

Notes:

Adding users and groups to bus connector roles

- AdminTask.addUserToBusConnectorRole()
 - Used to assign a user permission to connect to the bus specified
 - *bus* is a string that contains the bus name.
 - *user* is a string that contains the user that is being assigned to the bus.

- AdminTask.addGroupToBusConnectorRole()
 - Used to assign a group permission to connect to the bus specified
 - *bus* is a string that contains the bus name.
 - *group* is a string that contains the group that is being assigned to the bus.

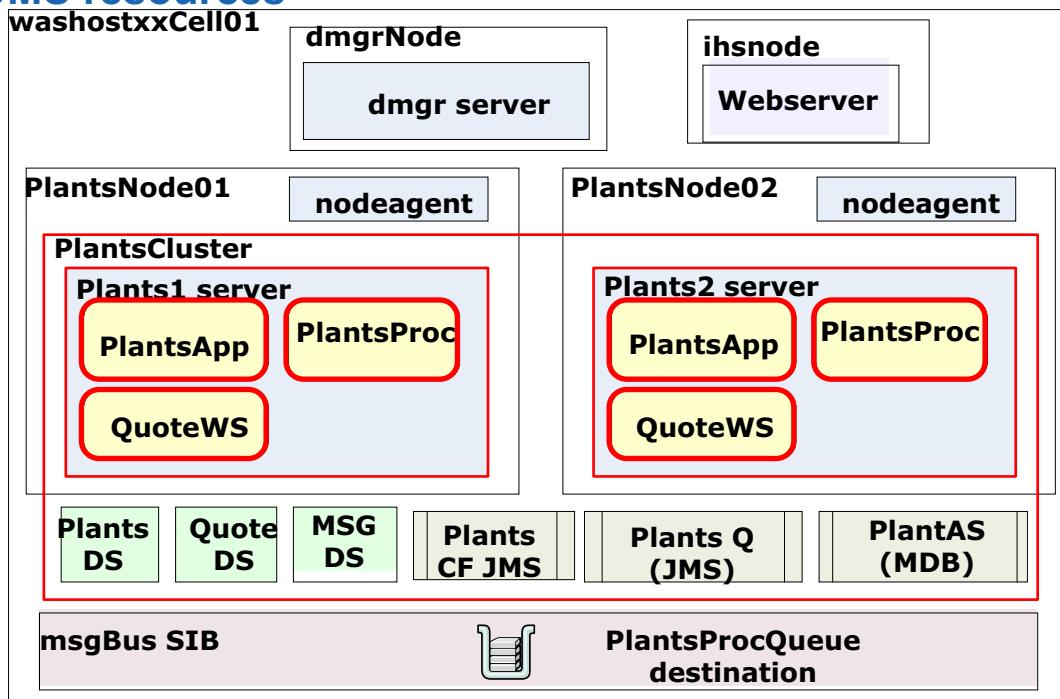
© Copyright IBM Corporation 2013

Figure 13-22. Adding users and groups to bus connector roles

WA680 / VA6801.0

Notes:

JMS resources



© Copyright IBM Corporation 2013

Figure 13-23. JMS resources

WA680 / VA6801.0

Notes:

Managing applications

- Use the AdminApp administrative object to:
 - Install, uninstall, list, edit, and export applications
 - Update application metadata
 - Map web modules to virtual hosts
 - Map modules to servers
- Use AdminConfig to perform configuration changes such as:
 - Specifying a library that the application uses
 - Setting session management configuration properties
- Use AdminControl to:
 - Start and stop applications

© Copyright IBM Corporation 2013

Figure 13-24. Managing applications

WA680 / VA6801.0

Notes:

Installing and uninstalling applications

- Steps for installing an application:
 1. Determine which options to use to install your application using AdminApp.options()
 2. Install the application
 3. Save the configuration change.

```
AdminApp.install("C:/MyApps/PlantsApplication.ear", [ ["-cluster",
"PlantsCluster"]])
AdminConfig.save()
```

- Uninstalling an application:

- Use AdminApp.uninstall()

```
AdminApp.uninstall("PlantsApplication")
AdminConfig.save()
```

Display name

© Copyright IBM Corporation 2013

Figure 13-25. Installing and uninstalling applications

WA680 / VA6801.0

Notes:

Starting and stopping applications

1. Get object name of *ApplicationManager* MBean with AdminControl
2. Use *AdminControl.invoke()* method to start the operation:
 - `startApplication` starts the application
 - `stopApplication` stops the application

```
appManager =  
AdminControl.queryNames ("cell=washost00Cell01,node=Node01,  
type=ApplicationManager,process=Plants1,*")  
  
AdminControl.invoke (appManager, "stopApplication",  
"PlantsApplication")
```

Display name

© Copyright IBM Corporation 2013

Figure 13-26. Starting and stopping applications

WA680 / VA6801.0

Notes:



Other application management tasks

- Listing application modules
 - AdminApp.listModules(*appDisplayName*)
- Updating an application
 - AdminApp.update(*appDisplayName*, *contentType*, *options*)
- Exporting an application
 - AdminApp.export(*appDisplayName*, *fileName*)

© Copyright IBM Corporation 2013

Figure 13-27. Other application management tasks

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe how to perform common application and application resource management tasks by using the administrative objects

© Copyright IBM Corporation 2013

Figure 13-28. Unit summary

WA680 / VA6801.0

Notes:



Checkpoint questions

1. Which AdminTask command group is used for the creation of JDBC providers and data sources?
2. True or false: You can add application servers, clusters, and WebSphere MQ servers to a service integration bus?
3. Which administrative object is used to install enterprise applications?

© Copyright IBM Corporation 2013

Figure 13-29. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

Checkpoint answers

1. Which AdminTask command group is used for the creation of JDBC providers and data sources?

Answer: The JDBCProviderManagement command group

2. True or false: You can add application servers, clusters, and WebSphere MQ servers to a service integration bus?

Answer: True

3. Which administrative object is used to install enterprise applications?

Answer: AdminApp

© Copyright IBM Corporation 2013

Figure 13-30. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 9



Deploying the PlantsByWebSphere application

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

WA680 / VA6801.0

Figure 13-31. Exercise 9

Notes:

Exercise objectives

At the end of this exercise, you should be able to:

- Set WebSphere environment variables
- Create database resources, including J2C authentication aliases, JDBC providers, and data sources
- Install an enterprise application to a cluster
- Map application modules to servers

© Copyright IBM Corporation 2013

Figure 13-32. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 14. Scripting methodologies and recommendations

What this unit is about

This unit gives recommendations and methodologies that should be considered when working on real world scripting effort.

What you should be able to do

After completing this unit, you should be able to:

- Describe multiple approaches to a scripting project
- Identify a process for subdividing a scripting effort

How you will check your progress

- Checkpoint questions

Unit objectives

After completing this unit, you should be able to:

- Describe multiple approaches to a scripting project
- Identify a process for subdividing a scripting effort

© Copyright IBM Corporation 2013

Figure 14-1. Unit objectives

WA680 / VA6801.0

Notes:



Topics

- Programming discipline
- Team development effort
- Asynchronous programming

© Copyright IBM Corporation 2013

Figure 14-2. Topics

WA680 / VA6801.0

Notes:

Why have such formal guidelines? (1 of 2)

- A good story grows in the “telling”
 - I am creating a “one time” or “quick and dirty” script
 - Plan for success
 - Scripts provide reliability, repeatability, and promote reuse
 - A single user effort might expand to a team; success brings more work
- Scripts always have a longer life than you expect
 - When a script does what is required, it works
 - When a script works, it is used, and copied
 - When a script does not work, it must be fixed
 - “There is always time to do it over”

© Copyright IBM Corporation 2013

Figure 14-3. Why have such formal guidelines? (1 of 2)

WA680 / VA6801.0

Notes:

Why have such formal guidelines? (2 of 2)

- What happens when “good scripts go bad?”
 - Plan for finding problems
 - Use a framework for tracing
 - Other people need to easily understand your work
- Habits are hard to break
 - You should form good habits, not bad habits

© Copyright IBM Corporation 2013

Figure 14-4. Why have such formal guidelines? (2 of 2)

WA680 / VA6801.0

Notes:

14.1. Programming discipline for writing scripts

This topic provides information about using the IBM Installation Manager.

Programming discipline for writing scripts



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 14-5. Programming discipline for writing scripts

WA680 / VA6801.0

Notes:

Use source control

- Developers must feel secure in making changes
- Source control is critical part of team development
- Source control provides the ability:
 - To save work. (revisions, snapshots, and so on)
 - To roll back changes
 - To merge modifications that are made by different members (or over time)
 - To capture a set of resources at a point in time (version, release, and so on)
- These capabilities are important to team development

© Copyright IBM Corporation 2013

Figure 14-6. Use source control

WA680 / VA6801.0

Notes:

Unit Tests

- Part two of the safety net
 - Unit tests are functional tests
 - These tests are the definition of what works and what does not work
- These tests must grow as functionality grows
 - If there is no test, the functionality is not defined to work
 - As bugs are discovered, new unit tests are added to capture that case
- Regression testing requires unit tests
 - Capabilities should not regress
 - Working capabilities should not break when new features are added
 - “What works?” should not be in doubt

© Copyright IBM Corporation 2013

Figure 14-7. Unit Tests

WA680 / VA6801.0

Notes:

Coding standards (1 of 2)

- Comments
 - Should have a template for a prolog
 - Tell what is expected
 - Tell what should happen
 - Tell what is happening
 - Your code is not easily understood
- Trace or logging framework
 - Messages tell what is happening
 - Be able to show more or fewer messages
- Error handling
 - Expect and anticipate problems
 - Always log problems and errors
 - Error handling should improve as testing increases

© Copyright IBM Corporation 2013

Figure 14-8. Coding standards (1 of 2)

WA680 / VA6801.0

Notes:

Coding standards (2 of 2)

- Comments, Trace, and Error Handling
 - Each of these topics can take an entire chapter
 - Each of these topics can be an entire book
- The team should set coding standards
 - These standards grow out of a team process that is well run
 - Standards make the code easier to understand
 - Might be inherited from an IT governance process

© Copyright IBM Corporation 2013

Figure 14-9. Coding standards (2 of 2)

WA680 / VA6801.0

Notes:



Peer review

- Code belongs to the team, not any one individual
 - Rarely does one person do code maintenance
 - Do not wait until there is a problem to get more people to look at the code
- Some organizations use “pair programming”
 - Two people at a keyboard
 - One types and one thinks
 - Actually both are thinking
- Maintenance is easier
 - Code is easier to understand if more than one person creates the code
 - Less risk of losing a key person

© Copyright IBM Corporation 2013

Figure 14-10. Peer review

WA680 / VA6801.0

Notes:

Refactor the code

- Refactoring takes time
- An important part of coding standards
 - Good ideas are propagated across the team
- Refactoring requires confidence by developers
 - Units tests provide confidence that it still works
 - Source control provides confidence that the team can go back
 - Peer review and coding standards help ensure that the team understands the code

© Copyright IBM Corporation 2013

Figure 14-11. Refactor the code

WA680 / VA6801.0

Notes:

Keep it simple

- Avoid the temptation to create unrequested features
 - Require more time
 - Require more tests
 - Create more bugs
- Unit tests keep the focus on small units
 - Better to have many iterations
 - Do not try to build everything for the first release
 - Shorter build cycles with testing produce better results

© Copyright IBM Corporation 2013

Figure 14-12. Keep it simple

WA680 / VA6801.0

Notes:



Continuous integration

- Part of “Keeping it simple”
 - Create smaller modules, which are based on units tests
 - Each part of code should do one thing well
- Critical for team development
 - A successful integration is a release (or version) in source control
 - A good time for peer review
 - Peer review often leads to code refactoring
 - Code refactoring and peer review leads to code standards
- All parts work together
 - The team code
 - The development process comes together

© Copyright IBM Corporation 2013

Figure 14-13. Continuous integration

WA680 / VA6801.0

Notes:

Extreme Programming

- The principals that described are based on Extreme Programming
 - Having a process is more important than which process
 - A process must not become more important than the work
- Extreme programming is a lightweight process
 - It is based on simple principals
 - As a process, it can grow as a project grows
- The operation role uses scripting
 - An administrative activity that should use a development discipline
 - Avoid a “one time” and “quick and dirty” mentality

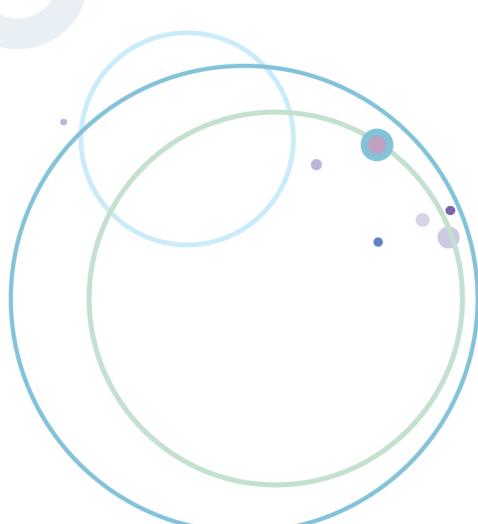
© Copyright IBM Corporation 2013

Figure 14-14. Extreme Programming

WA680 / VA6801.0

Notes:

Scripting and asynchronous programming



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 14-15. Scripting and asynchronous programming

WA680 / VA6801.0

Notes:

Why asynchronous scripting?

- What does asynchronous mean?
 - One section of code starts an activity
 - A different section of code waits for a notification
- Some administrative activities cannot be done in sequence
 - Ripple start or stop of servers
 - Some activities take too long for the script to wait
- It is beyond the scope of this course
- Look at MBeans and the JMX framework
 - MBeans allow for asynchronous programming
 - MBeans provide Notification Listener interfaces

© Copyright IBM Corporation 2013

Figure 14-16. Why asynchronous scripting?

WA680 / VA6801.0

Notes:

References (1 of 2)

- Asynchronous administration

<http://www14.software.ibm.com/webapp/wsbroker/redirect?version=phil&product=was-nd-iseries&topic=tjmxUpdateWholeApp>

- Gibson, Robert. *WebSphere Application Server Administration Using Jython*. ISBN-13: 978-0137009527 IBM-Press, October 28, 2009.

- “Extreme Programming”

<http://www.extremeprogramming.org>

© Copyright IBM Corporation 2013

Figure 14-17. References (1 of 2)

WA680 / VA6801.0

Notes:



References (2 of 2)

- Van Sickel, Peter. "Using the latest Jython with a WebSphere Application Server wsadmin thin client." *Developer Works* July 2012
http://www.ibm.com/developerworks/websphere/library/techarticles/1207_vansickel/1207_vansickel.html
- Van Sickel, Peter. "WebSphere Application Server wsadmin script development using the PyDev plug-in for Eclipse." *Developer Works* September 2012
http://www.ibm.com/developerworks/websphere/techjournal/1209_vansickel/1209_vansickel.html

© Copyright IBM Corporation 2013

Figure 14-18. References (2 of 2)

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe multiple approaches to a scripting project
- Identify a process for subdividing a scripting effort

© Copyright IBM Corporation 2013

Figure 14-19. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. (True/False) Developing administrative scripts is best done as a solitary activity.
2. Name two important coding standards to consider in writing scripts?
3. What scenario might require an asynchronous programming model?

© Copyright IBM Corporation 2013

Figure 14-20. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.



Checkpoint answers

1. (True/False) Developing administrative scripts is best done as a solitary activity.
FALSE: It is often a team writing scripts.
2. Name two important coding standards to consider in writing scripts?
Comments, Error Handling, Trace
3. What scenario might require an asynchronous programming model?
Updating an application, ripple starting/stopping servers, where there is the potential for a long wait time.

© Copyright IBM Corporation 2013

Figure 14-21. Checkpoint answers

WA680 / VA6801.0

Notes:

Unit 15. Using ws_ant

What this unit is about

This unit describes how to use ws_ant in automating a WebSphere Application Server runtime environment.

What you should be able to do

After completing this unit, you should be able to:

- Describe the features and usage of ws_ant

How you will check your progress

- Checkpoint questions
- Lab exercises

References

WebSphere Application Server V8.5 Information Center

<http://ant.apache.org>

<http://ant.apache.org/manual/tasksoverview.html>

Unit objectives

After completing this unit, you should be able to:

- Describe the features and usage of ws_ant

© Copyright IBM Corporation 2013

Figure 15-1. Unit objectives

WA680 / VA6801.0

Notes:

What is Ant?

- Apache Ant is a Java based build tool
- Provides a simple open source scripting engine that can run scripts that are written in XML format
- WebSphere Application Server provides a set of Ant-based tasks that you can use to manage some common administrative operations
- Allows you to automate the task of compiling, packaging, installing, and testing applications on the application server

© Copyright IBM Corporation 2013

Figure 15-2. What is Ant?

WA680 / VA6801.0

Notes:

In theory, Ant it is similar to Make, but Ant is different. Instead of a model that is extended with shell-based commands, you can extend Ant by using Java classes. Instead of writing shell commands, XML-based configuration files are used. These files reference a target tree in which various tasks are run. Each task is run by an object that implements a particular Task interface.

Ant features

- Cross operating system build utility
- Improves the reliability of enterprise application construction and deployment
- Ant can be extended by using Java classes
 - An object that implements a Task interface runs each task
- Ant uses XML to describe the build process and its dependencies

© Copyright IBM Corporation 2013

Figure 15-3. Ant features

WA680 / VA6801.0

Notes:

The latest release of Ant was March 2013, Apache Ant 1.9.0. It is available for download at <http://ant.apache.org/bindownload.cgi>.

Features of the latest release include:

Uses Java 1.5

Now supports @Ignore annotation in JUnit 4 test cases.

The compressed file, bzip2, and tar tasks are improved.

Why use Ant?

- Allows you to automate tasks such as compiling, packaging, installing, and testing applications
- Accelerates the construction of Java projects
- Ideal for tasks that involve application deployment and server operations
- Create reusable and reliable scripts

© Copyright IBM Corporation 2013

Figure 15-4. Why use Ant?

WA680 / VA6801.0

Notes:

In the Ant environment, you can create platform-independent scripts that compile, package, install, and test your application on WebSphere Application Server. It integrates with **wsadmin** scripts that use Ant as their invocation mechanism.

Ant buildfiles

- Ant buildfiles are XML text files
 - By default, Ant searches for an XML file named `build.xml`.
 - Use the `-buildfile` command-line option to specify another XML file
- Each buildfile contains one project and at least one target
- Targets contain task elements
- Each task element of the buildfile can have an ID attribute
 - The ID attribute must be unique and can later be referred to by other elements

© Copyright IBM Corporation 2013

Figure 15-5. Ant buildfiles

WA680 / VA6801.0

Notes:

From the Apache Ant documentation: “Apache Ant’s buildfiles are written in XML. Each buildfile contains one project and at least one (default) target. Targets contain task elements. Each task element of the buildfile can have an id attribute and can later be referred to by the value that is supplied to this attribute. The value has to be unique.”
Reference: <http://ant.apache.org/manual/using.html>

Ant: buildfile example

```

<project name="AntScripting" default="init"> ← Define project and
    <!-- Read a properties file --> default target
        <property file="ant.properties"/> ← Load properties file
            <target name="init"> ← Define target definition
                <tstamp/>
                <echo message="- Build of ${ant.project.name} started at ${TSTAMP} on
${TODAY} ---"/>
                <echo message="--- Using ant version: ${ant.version} "/>
                <echo message="build.xml"/>
            </target> Start wsadmin with command options
            <target name="stopApplication" description="Stops the specified
application.">
                <echo message="- Stopping the ${applicationName} application. "/>
                <wsadmin script="${softwareDir}\stopApplication.py" lang="jython" ←
                    profile="${softwareDir}\profile.py" port="${soapPort}">
                    conntype="soap" user="${adminID}" password="${adminPasswd}">
                        <arg value="${applicationName}"/>
                        <arg value="${clusterName}"/>
                </wsadmin>
            . . .
        
```

© Copyright IBM Corporation 2013

Figure 15-6. Ant: buildfile example

WA680 / VA6801.0

Notes:

This Ant buildfile example shows the definition of a project that contains two targets. The default target in this example is “init” which is specified by the “default” property. A properties file called “ant.properties” is loaded. The definition of the target named “init” prints messages including the date and time and the name of the buildfile. The definition of the target named “stopApplication” uses wsadmin to run a Jython script named stopApplication.py. Two arguments are passed to the Jython script: the application name, and the cluster name where the application is running. The values of these arguments are read from the properties file, in this example. Finally, a task is defined called “wsadmin” that points to the Java class that implements the functions of wsadmin.

Ant: projects

- Projects represent a set of Ant targets and tasks to be run within a buildfile
- Each project can define:
 - Properties files
 - Tasks
 - Targets

```
<project name="AntScripting" default="init"> ← Project definition
  <property file="ant.properties"/>
  <taskdef name="wsadmin"
    classname="com.ibm.websphere.ant.tasks.WsAdmin"/>
  <target name="init">
    <tstamp/>
    <echo message="- Build ${ant.project.name} started at ${TSTAMP} "/>
      <echo message="- Using ant version: ${ant.version} "/>
      <echo message="build.xml"/>
    </target>
</project>
```

© Copyright IBM Corporation 2013

Figure 15-7. Ant: projects

WA680 / VA6801.0

Notes:

Each Project has a build file. The default build file name is build.xml. Use the -buildfile command-line option to load another buildfile.

Each project defines one or more targets. A target is a set of tasks you want to run. When Ant starts, the targets to run can be included as command options.

Ant: properties

- A project can have a set of properties
- Properties have name and value pairs
 - The name is case-sensitive
- Properties can be set in the following ways:
 - Using the property name and value attributes


```
<property name="earLocation"
            value="/opt/IBM/WebSphere/InstallableApps"/>
```
 - Using the property file attribute


```
<property file="ant.properties"/>
```
- Use the following syntax to include the value of a property:
 - \${propertyName}
- For example:
 - <echo>All EAR files are installed from the \${earLocation} directory</echo>

© Copyright IBM Corporation 2013

Figure 15-8. Ant: properties

WA680 / VA6801.0

Notes:

To include the property construct literally without property substitutions, escape the '\$' character with a second '\$'. For example:

```
<echo> The $$\{earLocation\} property is set to ${earLocation} </echo>.
```

Ant provides access to all system properties. For example, \${os.name} returns the name of the operating system. For more information about system properties, refer to the System.getProperties() JavaDoc. Additionally, Ant has the following built-in properties defined:

- **basedir** — The absolute path of the project base directory
- **ant.file** — The absolute path of the buildfile
- **ant.project.name** — the name of the project that is running
- **ant.java.version** — The JVM version Ant detected
- **ant.home** — home directory of Ant

Ant: property file example

```
# Administration ID and password for use by scripts
adminID=wasadmin
adminPasswd=web1sphere
softwareDir=/usr/Software/Scripts/Exercise10/
wasHome=/opt/IBM/WebSphere/AppServer/

# Location to install applications on cluster
earLocation=/opt/IBM/WebSphere/AppServer/profiles/Dmgr/installableApps/

# Port number for the Dmgr's SOAP connector address
soapPort=8879
connType=soap

# Application name and server cluster name
applicationName=PlantsByWebSphere
clusterName=PlantsCluster
```

© Copyright IBM Corporation 2013

Figure 15-9. Ant: property file example

WA680 / VA6801.0

Notes:

Properties are a powerful way to make your build scripts behave dynamically. Ant properties are immutable. After they are set, they cannot be overridden. Since the calling Ant script sets the properties, the behavior of an Ant script can change. You can use this technique to change the directory structures of build scripts so that you can move the scripts from one environment to another.

It is possible when using the ant and antcall tasks, to input properties by the caller.

The antcall task is used to call another target within the same buildfile and optionally specify some properties (parameters in this context).

Ant: tasks

- A task is Java code that Ant runs
- Tasks are defined using the **<taskdef>** element
 - **name**: defines the name of the task that can be called within targets
 - **classname**: defines the Java class that is implemented when the task is run

```

<taskdef name="wsadmin" classname="com.ibm.websphere.ant.tasks.WsAdmin"/>

<target name="stopApplication">
    <wsadmin script="${softwareDir}\stopApplications.py"
        lang="jython"
        profile="${softwareDir}\profile.py" port ="${soapPort}"
        conntype="soap">
        <arg value ="${applicationNames} "/>
    </wsadmin>
</target>

```

© Copyright IBM Corporation 2013

Figure 15-10. Ant: tasks

WA680 / VA6801.0

Notes:

The basic procedure to create a custom Ant task includes:

- Creating a Java class that subclasses the org.apache.tools.ant.Task and overrides the execute() method
- Packaging the class files into a JAR file
- Using **taskdef** to register the custom task in the Ant buildfile

A **<task>** is executable Java code. A task can have multiple attributes. The value of an attribute might contain references to a property. These references are resolved before the task is run. Tasks have a common structure: **<name attribute1="value1" attribute2="value2" ... />** where *name* is the name of the task, *attributeN* is the attribute name, and *valueN* is the value for this attribute. Each task element of the buildfile can have an id attribute and can later be referred to by the value that is supplied to this. The value must be unique. Each Task is bound to a Java class file that Ant runs, passing to it any arguments or sub-elements that are defined with that task. The Ant tool is extensible and it allows you to create your own tasks.

Ant: task types (1 of 2)

Task type	Description
Archive	Used to start zip, jar, EAR, and other archiving utilities
Audit	Starts JDepend parse to traverse a set of Java source files to generate design-quality metrics for each Java package
Compile	Used to start JspC, Rmic, javac and other compilation utilities
Deployment	Task to run a deployment tool for vendor-specific J2EE servers
Documentation	Uses javadoc and stylebook tasks to generate documentation
EJB	Starts tasks to compile EJBs, create deployment descriptors, build EJB jar files, and many other EJB-related operations
Execution	Starts execution tasks such as exec, Java, sleep, and waitfor
File	Used to call file system operations such as move, copy, and delete
Logging	Starts listeners to record logging output of the build process to a file

© Copyright IBM Corporation 2013

Figure 15-11. Ant: task types (1 of 2)

WA680 / VA6801.0

Notes:

For more information about the specific tasks, refer to
<http://ant.apache.org/manual/tasksoverview.html>.

The task types that are listed in this table are fully documented. There can be several tasks that are associated with a task type. For example, here are some of the file tasks:

- Chmod
- Chown
- Delete
- Filter
- Get
- Mkdir
- Replace
- Sync

Touch

Ant: task types (2 of 2)

Task type	Description
Mail	Used to send SMTP emails
Miscellaneous	A collection of tasks to echo text to a file or the console, generate keys in a keystore, play sound files at the end of a build, validate XML and many more
Pre-process	Starts tasks to import other build files, generate documentation, create JNI headers from Java classes, and so on
Property	Used to set properties, load property files, locate classes, and so on
Remote	Starts tasks to connect and use FTP, telnet, rexec sessions, ssh, Java web proxy clients
SCM	Used to connect to client repositories to extract code and upload changes
Testing	Starts Junit and JunitReport tasks

© Copyright IBM Corporation 2013

Figure 15-12. Ant: task types (2 of 2)

WA680 / VA6801.0

Notes:

For more information about the specific tasks, see the Ant manual at
<http://ant.apache.org/manual/tasksoverview.html>

Ant: targets

- A **target** is a set of tasks that can run
- When starting Ant, you select the targets that you want to run
 - When no target is provided, the default target is run

```
<target name="stopApplication" description="Stops the specified application.">
  <echo message="--- Stopping the ${applicationName} application. ---"/>

  <wsadmin script="${softwareDir}\stopApplication.py" lang="jython"
    profile="${softwareDir}\profile.py" port="${soapPort}" conntype="soap"
    user ="${adminID}"
    password ="${adminPasswd}">
    <arg value ="${applicationName}"/>
    <arg value ="${clusterName}"/>
  </wsadmin>
</target>
```

© Copyright IBM Corporation 2013

Figure 15-13. Ant: targets

WA680 / VA6801.0

Notes:

The default target is specified in the *default* attribute of the `<project>` element. For example:

```
<project name="AntScript" default="init">
```

The `<target>` element has the following attributes: `name`, `depends`, `if`, `unless`, `description`, `default`, and `basedir`.

name - the target a name

depends - comma-separated list of names of targets on which this target depends

if - name of the property that must be set in order for this target to run.

unless = name of the property that must not be set in order for this target to run.

description - short description of this target function.

Targets must have a name and might have several more attributes that determine when, and if, the target gets ran. The target is made up of one or more Tasks that can start a command or another program. Targets can have dependencies. For example: “install”

depends on “compile”. Targets can handle cascading dependencies; however, each dependency is handled only once. The dependency is run only when required.

It should be noted, however, that the Ant “depends” attribute specifies only the order in which targets must be run. It does not affect whether the target that specifies the dependency gets run if the dependent target did not run.

Ant tries to run the targets in the “depends” attribute in the order that they are listed (from left to right). Keep in mind that it is possible that a target can get run earlier when an earlier target depends on it. A target gets run only once, even when more than one target depends on it.

Ant: target dependencies

- A target can depend on other targets
 - To build a target, you use the `depends` attribute to specify the order in which targets must be run
 - Ant runs targets in the order that is listed from left to right on the command line

```
<target name= "stopApplication">
    <wsadmin script="${softwareDir}\stopApplication.py" lang="jython"
        profile="${softwareDir}\profile.py" port ="${soapPort}"
        conntype="soap">
        <arg value ="${applicationName}" />
    </wsadmin>
</target>
<target name="uninstallApplication" depends="stopApplications">
    <wsadmin script="${softwareDir}\uninstallApplication.py" lang="jython"
        profile="${softwareDir}\profile.py" port ="${soapPort}"
        conntype="soap">
        <arg value ="${applicationName}" />
    </wsadmin>
</target>
```

© Copyright IBM Corporation 2013

Figure 15-14. Ant: target dependencies

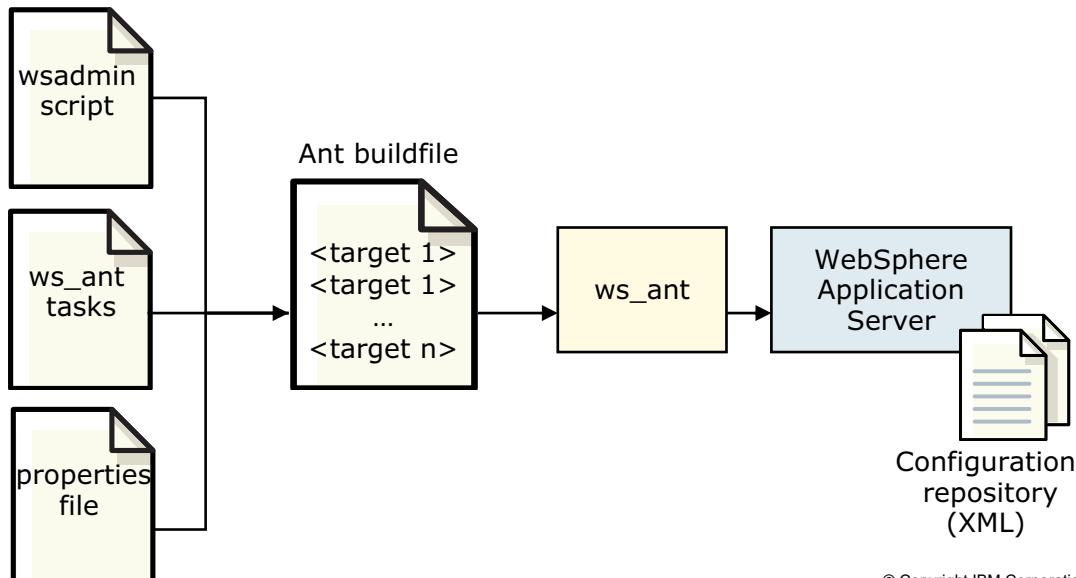
WA680 / VA6801.0

Notes:

In a chain of dependencies stretching back from a particular target, each target gets run only once, even when more than one target depends on it.

What is ws_ant?

- **ws_ant** is the WebSphere utility provided to use Apache Ant with Java Platform, Enterprise Edition applications
- A set of custom WebSphere Ant tasks extends the existing capabilities to include product-specific functions



© Copyright IBM Corporation 2013

Figure 15-15. What is ws_ant?

WA680 / VA6801.0

Notes:

The Apache Ant tasks for the WebSphere Application Server are in the Java package `com.ibm.websphere.ant.tasks`. The API documentation for this pack contains detailed information about all of the Ant tasks that are provided and how to use them.

By combining the following tasks with those tasks provided by Ant, you can create build scripts that compile, package, install, and test your application on the application server:

- Install and uninstall applications
- Start and stop servers in a base configuration
- Run administrative scripts or commands
- Run the Enterprise JavaBeans (EJB) deployment tool
- Run the JavaServer Pages (JSP) file precompilation tool

See the documentation for `com.ibm.websphere.ant.tasks` API in the Reference section of the WebSphere Application Server Information Center.

ws_ant: predefined tasks (1 of 2)

ws_ant task name	Description
DefaultBindings	Generates default IBM WebSphere Bindings for the specified EAR file
endptEnabler	Enables a set of web services within an Enterprise application archive
InstallApplication	Installs a new application to a single server or managed cell
Java2WSDL	Maps a Java class to a Web Services Description Language (WSDL) file
JspC	Compiles a directory of JSP files into .class files
ListApplications	Lists all the applications installed on a single server or managed cell
Messages	Used by other wsanttasks to retrieve their National Language Messages
ModuleValidator	Performs validation of the deployment descriptor, extensions, and bindings documents of an EAR, WAR, EJB Jar, or Application client jars
NLSEcho	Echo extension task to display translated messages

© Copyright IBM Corporation 2013

Figure 15-16. ws_ant: predefined tasks (1 of 2)

WA680 / VA6801.0

Notes:

The classes for the ws_ant tasks are defined in the <was_root>/plugins/com.ibm.ws.runtime.jar archive.

ws_ant: predefined tasks (2 of 2)

ws_ant task name	Description
ServerControl	The abstract base class of StartServer, StopServer, and ServerStatus
ServerStatus	Test the running status of an application server
StartApplication	Used to start an enterprise application
StartServer	Used to start a stand-alone server instance
StopApplication	Used to stop an existing or newly installed application
StopServer	Used to stop a stand-alone server instance
UninstallApplication	Uninstalls an existing application
WsAdmin	Runs the WebSphere command-line administration tool with the specified arguments
WSDeploy	Runs the WebSphere Web Services Deploy tool
WSDL2Java	Creates Java classes and deployment descriptor templates from a WSDL file
WsEjbDeploy	Runs the WebSphere EJB Deploy tool on the specified Jar file with the specified options

© Copyright IBM Corporation 2013

Figure 15-17. ws_ant: predefined tasks (2 of 2)

WA680 / VA6801.0

Notes:

The Javadoc for the WsAdmin class is in the WebSphere Application Server Information Center.

<http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.javadoc.doc/web/apidocs/com/ibm/websphere/ant/tasks/WsAdmin.html>

ws_ant: common task attributes

Attribute name	Description
wasHome	Contains the location of the WebSphere Installation Directory
properties	A properties file containing attributes to set in the JVM System properties
profile	Specifies a script file to be run before the main command or file
profileName	The name of the server profile to be used
conntype	Specifies the type of connection to be used. The Default type is SOAP. The valid values are SOAP, RMI, JMS, and NONE.
script	Contains a set of commands in a file to be passed to the script processor.
host	The host name of the machine to connect to and specified whether the conntype is specified
port	The port on the host to connect to and specified whether the conntype is specified
user	The user ID to authenticate with
password	The password to authenticate with

© Copyright IBM Corporation 2013

Figure 15-18. ws_ant: common task attributes

WA680 / VA6801.0

Notes:

Depending on the task, some of these attributes are required and others are optional, and have default values. For example, for WsAdmin, the following attributes are required: command or script.

ws_ant: defining tasks

- Use the `<taskdef>` element to define the ws_ant tasks in the buildfile
 - Required attributes are `name` and `classname`.
- For example: `<taskdef name="wsJspC" classname="com.ibm.websphere.ant.tasks.JspC"/>`
 - Makes a task that is called `wsJspC` available to Apache Ant. The class `com.ibm.websphere.ant.tasks.JspC` implements the task.

```
<taskdef name="wsInstallApp"
         classname="com.ibm.websphere.ant.tasks.InstallApplication" />

<taskdef name="wsUninstallApp"
         classname="com.ibm.websphere.ant.tasks.UninstallApplication" />

<taskdef name="wsStartApp"
         classname="com.ibm.websphere.ant.tasks.StartApplication" />

<taskdef name="wsStopApp"
         classname="com.ibm.websphere.ant.tasks.StopApplication" />
```

© Copyright IBM Corporation 2013

Figure 15-19. ws_ant: defining tasks

WA680 / VA6801.0

Notes:

This slide shows how to define tasks with a buildfile by using the `<taskdef>` element. The required attributes are the task name and the Java class name that implements the task. The tasks that are shown in the examples are some of the wsadmin tasks that are presented on the next few slides.

ws_ant: installing and uninstalling applications

- The **wsInstallApp** task enables you to install a new application into a WebSphere server or cell.
 - This task is a wrapper for the `AdminApp.install()`.

```
<target name="install">
  <wsInstallApp ear="${earDirectory}/${applicationName}.ear" options="-
    appname ${applicationName} -usedefaultbindings" port="${connPort}"
    conntype="${connType}" user="${adminId}" password="${adminPassword}" />
</target>
```

- The **wsUninstallApp** task enables you to uninstall an application from a WebSphere server or cell.
 - This task is a wrapper for the `AdminApp.uninstall()`.

```
<target name="uninstall">
  <wsUninstallApp application="${applicationName}" port="${connPort}"
    conntype="${connType}" user="${adminId}" password="${adminPassword}" />
</target>
```

© Copyright IBM Corporation 2013

Figure 15-20. ws_ant: installing and uninstalling applications

WA680 / VA6801.0

Notes:

The example target definitions on this slide illustrate task wrappers for the wsadmin Jython commands `AdminApp.install()` and `AdminApp.uninstall()`.

The targets call different tasks, `wsInstallApp` and `wsUninstallApp`, but have similar task attributes, such as `port`, `conntype`, `user`, and `password` for connecting to the application server.

ws_ant: starting and stopping applications

- The **wsStartApp** task enables you to start an application on a WebSphere server or cell
 - This task is a wrapper for the `ApplicationManager.startApplication()` command

```
<target name="startApp">
  <wsStartApp application="${applicationName}" port="${connPort}"
    conntype="${connType}" user="${adminId}" password="${adminPassword}" />
</target>
```

- The **wsStopApp** task enables you to stop an application on a WebSphere server or cell
 - This task is a wrapper for the `ApplicationManager.stopApplication()` command

```
<target name="stopApp">
  <wsStopApp application="${applicationName}" port="${connPort}"
    conntype="${connType}" user="${adminId}" password="${adminPassword}" />
</target>
```

© Copyright IBM Corporation 2013

Figure 15-21. ws_ant: starting and stopping applications

WA680 / VA6801.0

Notes:

The example target definitions on this slide illustrate task wrappers for the wsadmin Jython commands `ApplicationManager.startApplication()` and `ApplicationManager.stopApplication()`.

The targets call different tasks, `wsStartApp` and `wsStopApp`, but have identical task attributes. Recall that the values of the attributes are read from the properties file that is referenced in the buildfile or are explicitly defined within the buildfile.

ws_ant: Running the JSP precompilation tool

- The **JspC** task compiles JSP files into **.class** files by using the WebSphere JSP Batch Compiler

```
<target name="compileJSPs">
  <wsJspC wasHome="${wasHome}" enterpriseappName="${applicationName}"
    forcecompilation="true" keepgenerated="true" />
</target>
```

- Only one of the following four attributes is required to define the source for the JSPs

Attribute name	Description
earPath	The full path to single compressed or expanded enterprise application archive
warPath	The full path to single compressed or expanded web application archive
src	The same as warPath - for compatibility with an earlier version
enterpriseAppName	The name of a deployed Enterprise Application

© Copyright IBM Corporation 2013

Figure 15-22. ws_ant: Running the JSP precompilation tool

WA680 / VA6801.0

Notes:

The JspC task compiles a directory that contains JSP files into **.class** files. Normally, JSP files are converted into servlets the first time they are started. When a JSP is started, a servlet is generated and then compiled into a **.class** file. Although this compilation occurs only once, some applications might not accept the first-time invocation's demand on system resources. Pre-compiling JSPs can remove the cost of initially starting JSP files. The JspC task can make it easier to automate this part of your build process.

ws_ant: build.xml file example (1 of 2)

```

<project name="MyProject" default="info">
    <taskdef name="wsInstallApp"
        classname="com.ibm.websphere.ant.tasks.InstallApplication" />
    <taskdef name="wsUninstallApp"
        classname="com.ibm.websphere.ant.tasks.UninstallApplication" />
    <taskdef name="wsStartApp"
        classname="com.ibm.websphere.ant.tasks.StartApplication" />
    <taskdef name="wsStopApp"
        classname="com.ibm.websphere.ant.tasks.StopApplication" />
    <!--All of the targets run scripts which take their arguments from the
    command line. The arguments passed to the scripts are read from a
    properties file.
    -->
    <properties file= "/usr/Software/myConfig.properties" />

    <target name="info">
        <echo>
            Possible targets are: install uninstall startApp, stopApp
        </echo>
    </target>

```

© Copyright IBM Corporation 2013

Figure 15-23. ws_ant: build.xml file example (1 of 2)

WA680 / VA6801.0

Notes:

The next two slides show all of the WebSphere task definitions and and targets that were previously described assembled into one buildfile. This buildfile can be started with ws_ant, and any of the target names can be used on the command line. For example, ws_ant.sh -f build.xml install.

ws_ant: build.xml file example (2 of 2)

```

<target name="install">
    <wsInstallApp ear="${earDirectory}/${applicationName}.ear" options="-
        appname ${applicationName} -usedefaultbindings" port="${connPort}"
        conntype="${connType}" user="${adminId}" password="${adminPassword}" />
    </target>

<target name="uninstall">
    <wsUninstallApp application="${applicationName}" port="${connPort}"
        conntype="${connType}" user="${adminId}" password="${adminPassword}" />
    </target>

<target name="startApp">
    <wsStartApp application="${applicationName}" port="${connPort}"
        conntype="${connType}" user="${adminId}" password="${adminPassword}" />
    </target>

<target name="stopApp">
    <wsStopApp application="${applicationName}" port="${connPort}"
        conntype="${connType}" user="${adminId}" password="${adminPassword}" />
    </target>
</project>
```

© Copyright IBM Corporation 2013

Figure 15-24. ws_ant: build.xml file example (2 of 2)

WA680 / VA6801.0

Notes:

The targets in this part of the buildfile can be started as follows: `ws_ant.sh -f build.xml install`, `ws_ant.sh -f build.xml uninstall`, `ws_ant.sh -f build.xml startApp`, and `ws_ant.sh -f build.xml stopApp`.

ws_ant: script execution

- The standard syntax to run a specified target in a build file is:
 - `ws_ant.sh -buildfile build.xml targetName1`
- The ws_ant utility is located under the `<was_root>/bin` directory
- Property files that any Ant scripts load can be placed in the `<was_root>/properties` directory
- Default buildfiles should also be added to the `<was_root>/bin` directory or the buildfile can be specified when ws_ant is run

© Copyright IBM Corporation 2013

Figure 15-25. ws_ant: script execution

WA680 / VA6801.0

Notes:

When no arguments are specified, Ant looks for a `build.xml` file in the current directory. If Ant finds that file, it uses that file as the build file and runs the target that is specified in the default attribute of the `<project>` tag. To make Ant use a build file other than `build.xml`, use the command-line option `-buildfile file`, where `file` is the name of the build file that you want to use (or a directory that contains a `build.xml` file).

ws_ant: script execution examples (1 of 2)

```
<was_root>/bin/ws_ant.sh
```

Starts ws_ant and attempts to locate a buildfile named build.xml. If the buildfile does not exist, an exception is thrown. If buildfile is found, the default target of the project is run.

```
<was_root>/bin/ws_ant.sh -buildfile mybuildfile.xml
```

Starts ws_ant and loads the buildfile named mybuildfile.xml. The default target of the project is run.

```
<was_root>/bin/ws_ant.sh -buildfile mybuildfile.xml  
install
```

Starts ws_ant and loads the buildfile named mybuildfile.xml. The target named install is run.

© Copyright IBM Corporation 2013

Figure 15-26. ws_ant: script execution examples (1 of 2)

WA680 / VA6801.0

Notes:

The examples on this slide show different ways that ws_ant is started and the behavior that results. These examples show how to make ws_ant use the default buildfile or another buildfile.

ws_ant: script execution examples (2 of 2)

```
<was_root>/bin/ws_ant.sh -buildfile mybuildfile.xml  
install startApp compileJSPs
```

Starts ws_ant and loads the buildfile named `mybuildfile.xml`. The targets run in the order that is specified from left to right

```
<was_root>/bin/ws_ant.sh -propertyfile  
my.properties
```

Starts ws_ant and loads the buildfile named `build.xml`. The properties file named `my.properties` is loaded and takes precedence over any defined in the Ant script `<property>` element.

© Copyright IBM Corporation 2013

Figure 15-27. ws_ant: script execution examples (2 of 2)

WA680 / VA6801.0

Notes:

The first example on this slide shows how to have multiple targets that are run in a particular order as specified on the command line.

The second example on this slide shows how to override a properties file that is defined within the buildfile by specifying an alternative properties file on the command line.

ws_ant: startup options (1 of 2)

```
<was_root>/bin>ws_ant.sh -help
ant [options] [target [target2 [target3] ...]]
Options:
  -help, -h           print this message
  -projecthelp, -p    print project help information
  -version            print the version information and exit
  -diagnostics        print information to diagnose or report problems.
  -quiet, -q          be extra quiet
  -verbose, -v         be extra verbose
  -debug, -d          print debugging information
  -emacs, -e          produce logging information without adornments
  -lib <path>          specifies a path to search for jars and classes
  -logfile <file>     use given file for log
  -logger <classname> the class which is to perform logging
```

© Copyright IBM Corporation 2013

Figure 15-28. ws_ant: startup options (1 of 2)

WA680 / VA6801.0

Notes:

The syntax of the ws_ant script is:

```
ws_ant.sh [-buildfile buildfile] [ target ] [ -instance instance ] [-find
file] [-listener listenerclass] [-logger loggerclass] [-logfile logfile]
[-emacs] [-debug] [-verbose] [-quiet] [-version] [-projecthelp] [-help]
```

The following are some valuable options for debugging and troubleshooting scripts.

-debug

This parameter is optional. If you specify this parameter, the script displays debugging information.

-verbose

This optional parameter turns on verbose messages, which can be helpful if you must debug the script.

-logger loggerclass

This parameter is optional. The value *loggerclass* specifies a class name to be used for logging. For example, `ws_ant.sh -logger org.apache.tools.ant.DefaultLogger`

-logfile *logfile*

This parameter is optional. The value *logfile* specifies the name of the file that the script writes logging information.

ws_ant: startup options (2 of 2)

```

-listener <classname> add an instance of class as a project listener
-noinput do not allow interactive input
-buildfile <file> use given buildfile
-D<property>=<value> use value for given property
-keep-going, -k execute all targets that do not depend on failed
target(s)
-propertyfile <name> load all properties from file with -D properties
precedence
-inputhandler <class> the class which will handle input requests
-find <file> search for buildfile towards the root of
the filesystem and use it
-nice number A niceness value for the main thread: 1 (lowest)
to 10 (highest); 5 is the default
-nouserlib Run ant without using the jar files from
${user.home}/.ant/lib
-noclasspath Run ant without using CLASSPATH

```

© Copyright IBM Corporation 2013

Figure 15-29. ws_ant: startup options (2 of 2)

WA680 / VA6801.0

Notes:

-listener *listenerclass*

This parameter is optional. The value *listenerclass* specifies a class name to be added as a project listener. You can use Log4jListener, which passes build events to Log4j.

For example, `ws_ant.sh -listener org.apache.tools.ant.listener.Log4jListener`

-find *file*

This parameter is optional. The value *file* tells ws_ant to search towards the root of the file system for the buildfile to use.

If you use the `-find [file]` option, ws_ant searches for a build file first in the current directory, then in the parent directory, and so on. It searches until either a build file is found or the root of the filesystem is reached. By default, it looks for a build file called `build.xml`. To have it search for a build file other than `build.xml`, specify the *file* argument.



IBM Assembly and Deploy Tools: Ant support

- Allows you to create and run Ant buildfiles from the workbench
- The buildfiles can operate on resources in the file system and resources in the workspace
- Output from an Ant buildfile is displayed in the Console view in the same hierarchical format that is seen when running Ant from the command line

© Copyright IBM Corporation 2013

Figure 15-30. IBM Assembly and Deploy Tools: Ant support

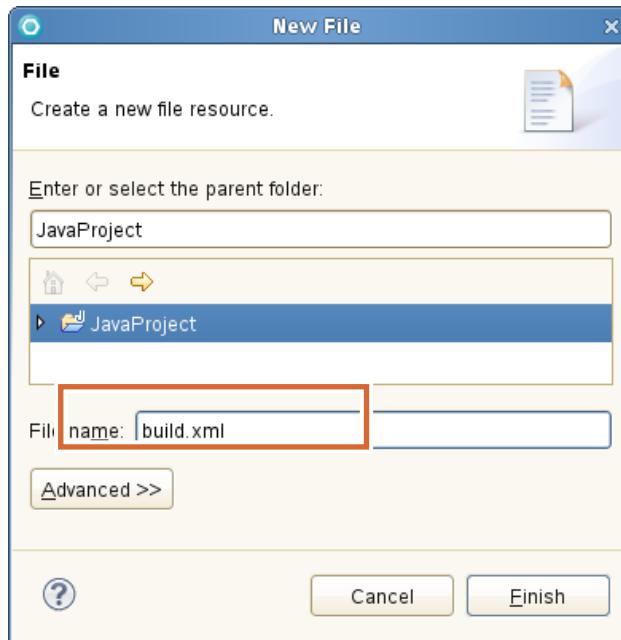
WA680 / VA6801.0

Notes:

In the Console view, Ant tasks are hyper-linked to the associated Ant buildfile. If a javac error occurs, reports are hyper-linked to the associated Java source file and line number.

IBM Assembly and Deploy Tools: creating buildfiles

- To create a buildfile
 - Select **File > New > File**
 - Enter the file name with an xml extension
 - Click **Finish**



© Copyright IBM Corporation 2013

Figure 15-31. IBM Assembly and Deploy Tools: creating buildfiles

WA680 / VA6801.0

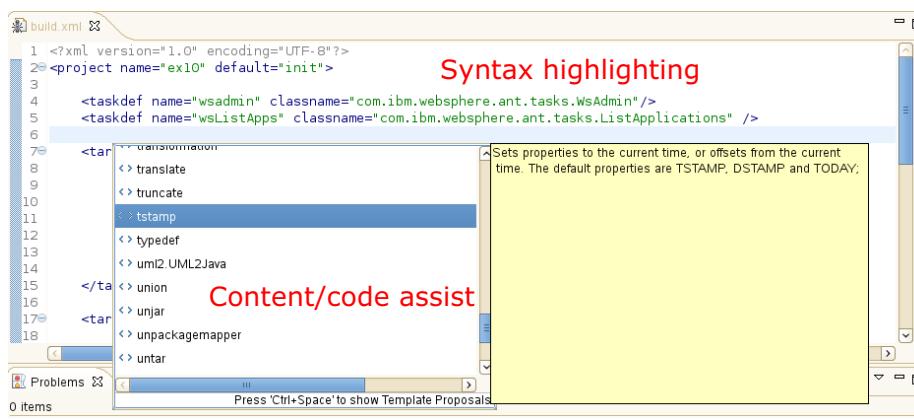
Notes:

When the buildfile has an extension of **.xml**, IADT considers it to be a possible Ant buildfile, and enables Ant-related actions when it is selected.



IBM Assembly and Deploy Tools: Ant editor

- The Ant editor provides specialized features for editor Ant buildfiles
- Associated with the editor is an Ant buildfile specific Outline view that shows the structure of the Ant build file
- The editor includes the following features:
 - Syntax highlighting
 - Content/code assist (including Ant-specific templates)
 - Annotations



© Copyright IBM Corporation 2013

Figure 15-32. IBM Assembly and Deploy Tools: Ant editor

WA680 / VA6801.0

Notes:

The most common way to start the Ant editor is to open an Ant buildfile from one of the navigation views or **Package Explorer** by using menus or by clicking the file (single or double-click depending on your preference settings). The Ant view (which is presented next), is updated as the buildfile is edited.

IBM Assembly and Deploy Tools: Ant view

- The Ant view is used to review Ant buildfiles
 - Build files are added to the view and reveal targets that are defined within them
 - Menu support to open an Ant editor for a buildfile, run selected targets or the entire buildfile
- To add the Ant view to the current perspective, click **Window > Show View > Other... > Ant > Ant**.
- Allows execution of the individual targets or the entire buildfile

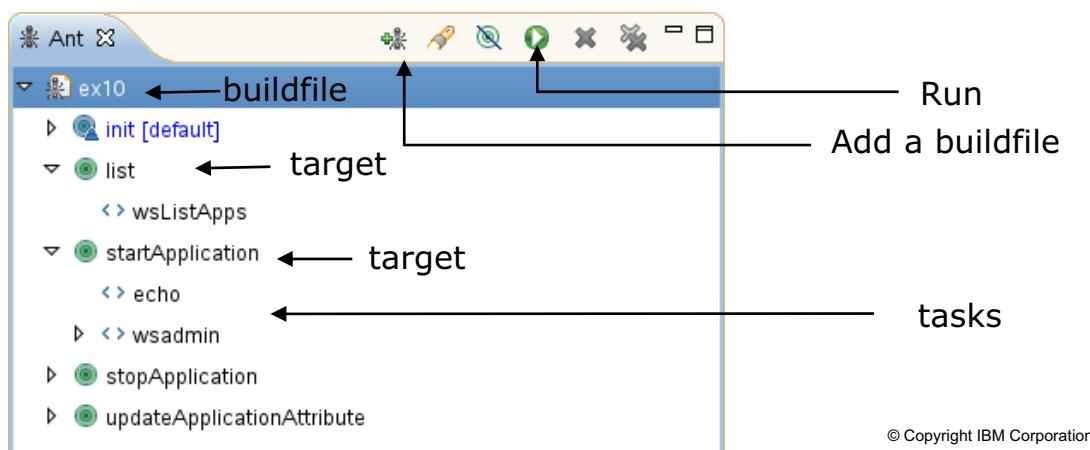


Figure 15-33. IBM Assembly and Deploy Tools: Ant view

WA680 / VA6801.0

Notes:

To add the **Ant view** to the current perspective, click **Window > Show View > Other... > Ant > Ant**

The Ant view allows you to select buildfiles in the workspace, search for buildfiles, hide internal targets, run the default target of the selected buildfile, and remove all, or selected buildfiles.

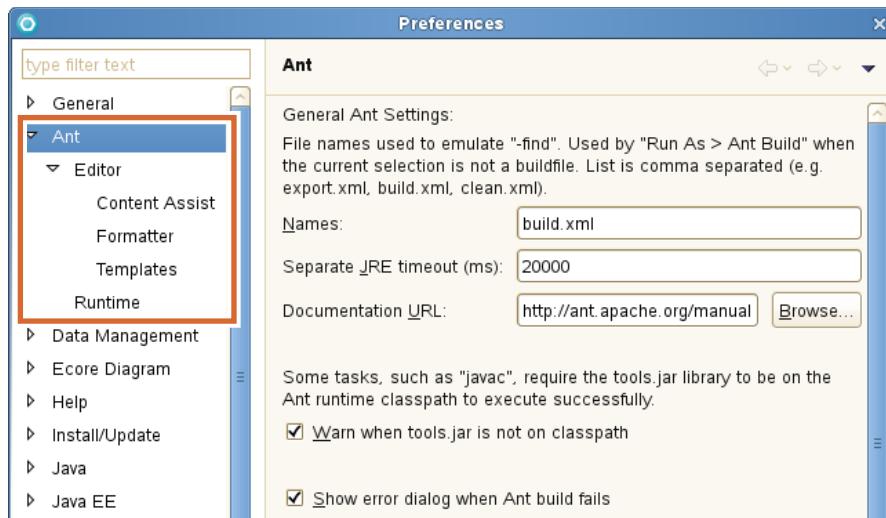
The menu contains actions to run the selected buildfile or a specific target of the buildfile. The purpose is to start the configuration dialog to edit the configuration that is associated with the selected buildfile to refresh buildfiles, and to select an editor for a buildfile.

IBM Assembly and Deploy Tools: Ant preferences

- To modify the Ant preferences:
 - Click **Windows > Preferences > Ant**
- Ant preference settings

include:

- Editor
 - Content Assist
 - Formatter
 - Templates
- Runtime



© Copyright IBM Corporation 2013

Figure 15-34. IBM Assembly and Deploy Tools: Ant preferences

WA680 / VA6801.0

Notes:

General Ant settings include:

Names: Configures the buildfiles that Ant's "-find" emulation searches for.

Separate JRE Timeout: The time in milliseconds to wait for communication with the Java virtual machine when running an Ant build in a separate Java runtime environment. The default value is 20000.

Documentation URL: Enables you to browse for the applicable documentation URL.

Warn when tools.jar is not on the classpath: Displays a warning when tools.jar is not on the classpath.

Show error dialog when Ant build fails: Displays an error message when the Ant build fails.

Create problem markers from javac results: Create warning and error markers from the build results of an Ant javac task. Marker creation requires that a console is allocated for the build and that the listfiles attribute of the javac task is set to true.

Ant Color Options: Configures the colors of the Ant build output.



IBM Assembly and Deploy Tools: Ant formatter

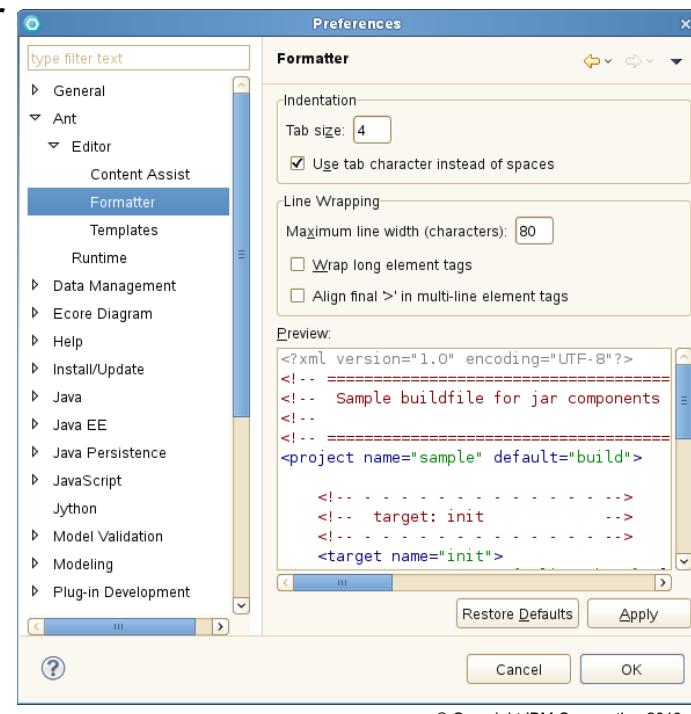
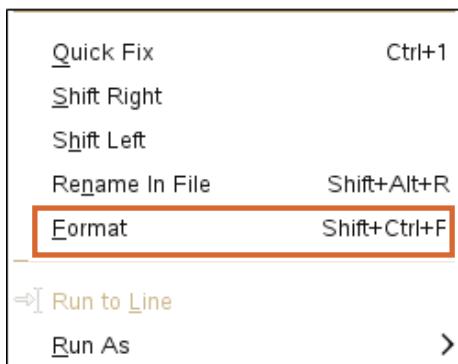
- To access the Ant formatting preferences:

- Click **Ant > Editor > Formatter**

- To format the Ant file:

- Use the menu and select

Format



© Copyright IBM Corporation 2013

Figure 15-35. IBM Assembly and Deploy Tools: Ant formatter

WA680 / VA6801.0

Notes:

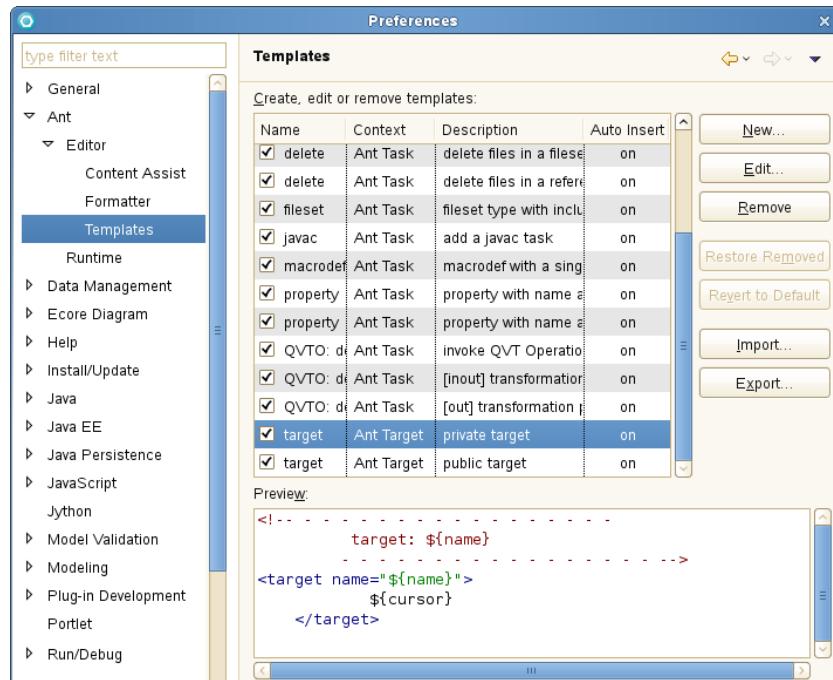
Formatter preferences include:

Indentation (Tab) size: This option controls how many spaces are used to display tabs in the Ant editor.

Line Wrapping Maximum line width: This option allows you to set the maximum line width for the Ant editor.

IBM Assembly and Deploy Tools: Ant templates

- Included by using the Ant editor content assist.
- Templates can be created, edited, or removed.
- Templates can be imported to restore removed templates and revert to the default templates.
- The Preview pane displays what the template generates.



© Copyright IBM Corporation 2013

Figure 15-36. IBM Assembly and Deploy Tools: Ant templates

WA680 / VA6801.0

Notes:

Some of the buttons on the Templates preference pane include:

New...: Allows you to add a template.

Edit...: Allows you to edit an existing template. Changes to a template can be viewed in the Preview Pane before they are applied.

Remove: Allows you to remove an existing template. A removed template can be restored by using Restore Removed or by reimporting the template (if it was imported initially).

IBM Assembly and Deploy Tools: ws_ant support

- To use ws_ant tasks with the IADT Ant runtime, you must import the <was_root>/plugins/com.ibm.ws.runtime.jar archive into the Ant runtime.
- To import the com.ibm.ws.runtime.jar
 - Click **Windows > Preferences > Ant > Runtime**
 - Click **Ant Home Entries**
 - Click **Add External JARs**
 - Browse for the JAR file

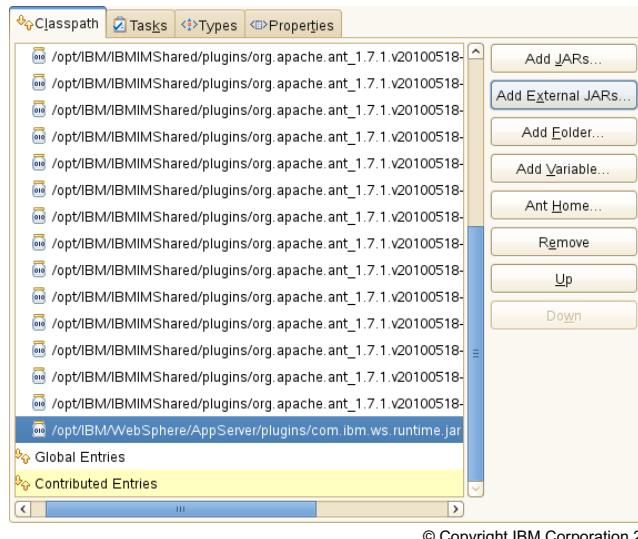


Figure 15-37. IBM Assembly and Deploy Tools: ws_ant support

WA680 / VA6801.0

Notes:

On the Classpath tab, you can add more classes that define tasks and types to the Ant classpath. New tasks and types are added to either the Ant Home or the Global Entries sections. You cannot modify the Contributed Entries section.

IBM Assembly and Deploy Tools: defining tasks

- Tasks are defined in a buildfile or through the IADT.
- Click **Window > Preferences > Ant > Runtime**.
- Click the **Tasks** tab.

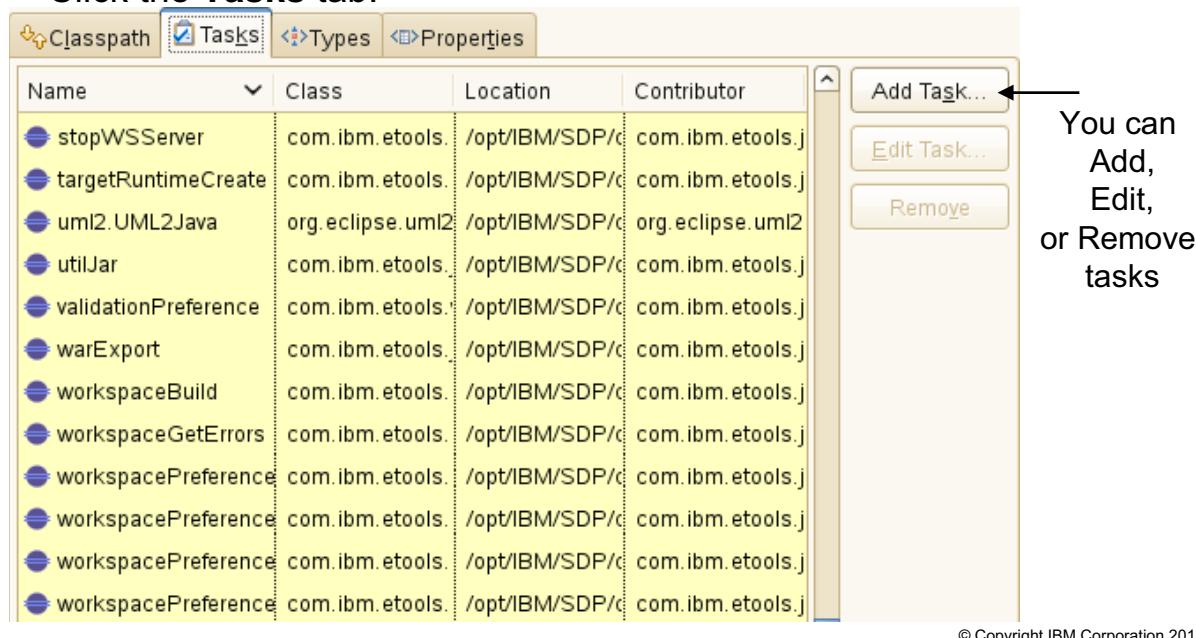


Figure 15-38. IBM Assembly and Deploy Tools: defining tasks

WA680 / VA6801.0

Notes:

On the Tasks tab, you can add tasks that are defined in one of the classes on the Ant classpath.

Add Task...: Allows you to add an Ant task definition to the runtime.

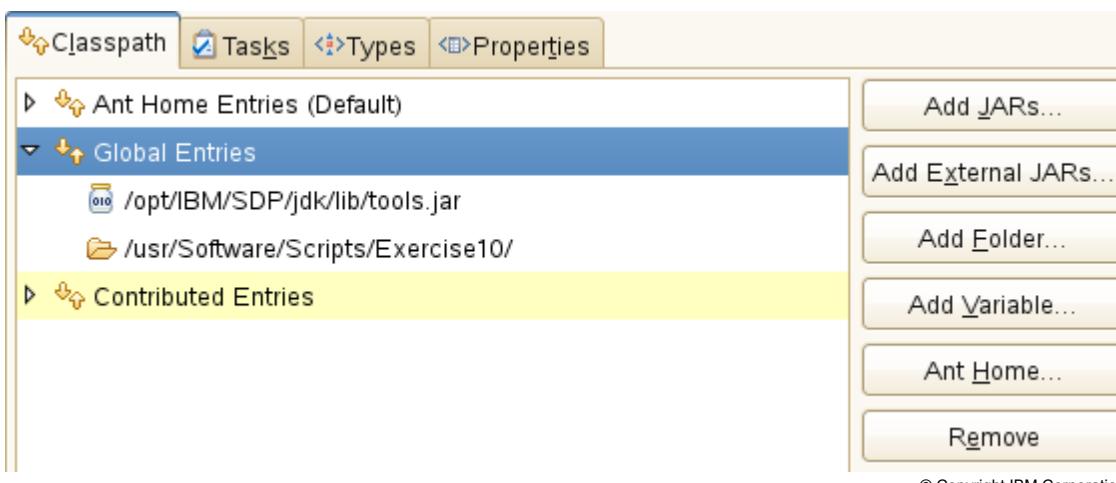
Edit Task...: Allows you to edit an existing Ant task. Note: you can edit only tasks that are not contributed (yellow decorated tasks are contributed and cannot be edited).

Remove: Allows you to remove an existing Ant task from the runtime. Note: you can remove only tasks that are not contributed (yellow decorated tasks are contributed and cannot be removed).



IBM Assembly and Deploy Tools: Ant properties files

- To load a properties file, you must configure the Ant run time to locate the folder that contains the properties file.
 - Click **Window > Preferences > Ant > Runtime**
 - Select Global Entries, and click **Add Folder**
 - Browse for, and select the folder that contains the Ant properties file
 - Click **OK**



© Copyright IBM Corporation 2013

Figure 15-39. IBM Assembly and Deploy Tools: Ant properties files

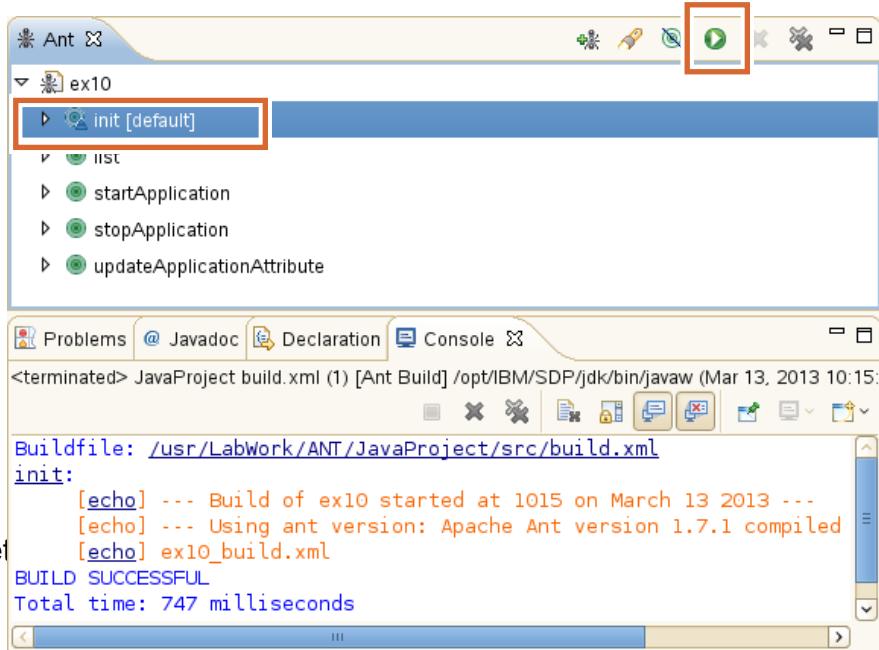
WA680 / VA6801.0

Notes:

To load a properties file, you must configure the Ant runtime to locate the folder that contains the properties file. You can tell Ant where to find the properties file by adding its folder to the Global Entries section of the Classpath.

IBM Assembly and Deploy Tools: Running buildfiles (1 of 2)

- To add the Ant view to the current perspective:
 - Click **Window > Show View > Other... > Ant > Ant**.
- To run an Ant buildfile by using the Ant view:
 - Double-click the buildfile or the specified target
 - Or select the target and click the **Run** icon



© Copyright IBM Corporation 2013

Figure 15-40. IBM Assembly and Deploy Tools: Running buildfiles (1 of 2)

WA680 / VA6801.0

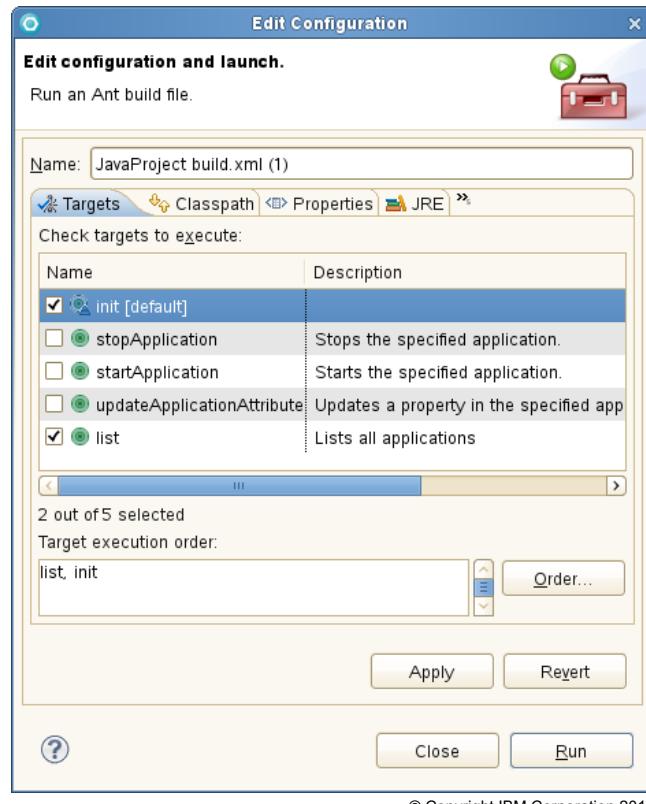
Notes:

The resulting execution is displayed in the Console view.



IBM Assembly and Deploy Tools: Running buildfiles (2 of 2)

- To run an Ant buildfile in the workbench:
 - Select the XML buildfile
 - From the menu, select **Run As > Ant Build...**
 - Select one or more targets from the **Targets** tab
 - Click **Run**



© Copyright IBM Corporation 2013

Figure 15-41. IBM Assembly and Deploy Tools: Running buildfiles (2 of 2)

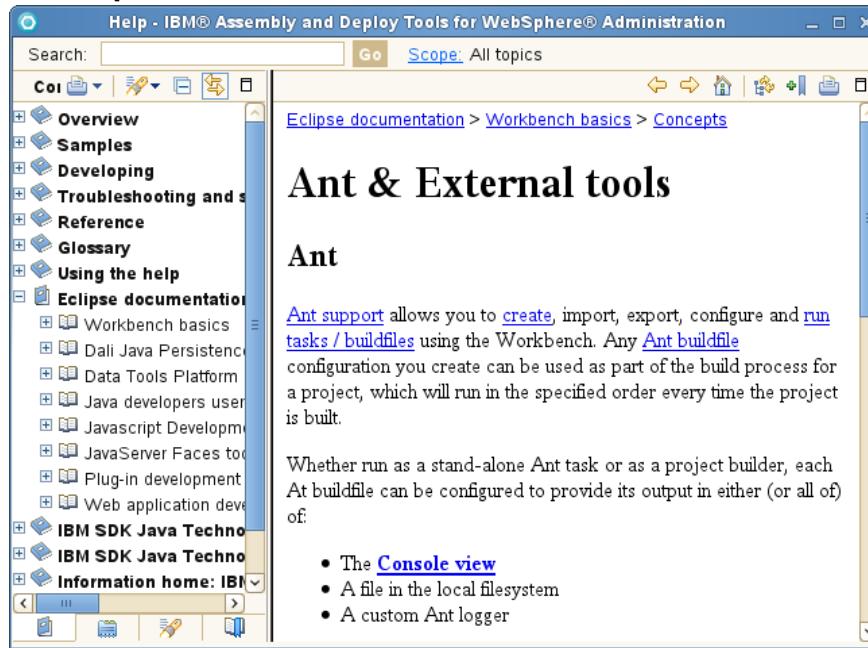
WA680 / VA6801.0

Notes:

On the configuration pane, you can select multiple targets, and specify their order of execution. The resulting execution is displayed in the Console view when you click **Run**.

IBM Assembly and Deploy Tools: accessing help

- To access Ant help topics from the IADT, select:
 - Help > Help Contents > Eclipse documentation > Workbench basics > Concepts > Ant & External tools



© Copyright IBM Corporation 2013

Figure 15-42. IBM Assembly and Deploy Tools: accessing help

WA680 / VA6801.0

Notes:

Help is also available in the WebSphere Application Server and Rational Application Developer Information Centers.

Unit summary

Having completed this unit, you should be able to:

- Describe the features and usage of ws_ant

© Copyright IBM Corporation 2013

Figure 15-43. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. True or False: Ant allows you to automate the task of compiling, packaging, installing, and testing applications on the application server.
2. How is an Ant target different from an Ant task?
3. What is the WebSphere utility that is provided to support the use of Apache Ant?

© Copyright IBM Corporation 2013

Figure 15-44. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. True or False: Ant allows you to automate the task of compiling, packaging, installing, and testing applications on the application server.

Answer: True

2. How is an Ant target different from an Ant task?

Answer: Ant tasks represent Java classes and Ant targets start a set of tasks

3. What is the WebSphere utility provided to support the use of Apache Ant?

Answer: ws_ant

© Copyright IBM Corporation 2013

Figure 15-45. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 10



ws_ant scripting and configuring the service integration bus

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 15-46. Exercise 10

WA680 / VA6801.0

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Combine individual Jython scripts into one Ant script with multiple tasks
- Use Ant property files
- Use Ant tasks for building application code
- Use Ant tasks for deployment and server operation
- Use Ant tasks to configure a service integration environment

© Copyright IBM Corporation 2013

Figure 15-47. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 16. Properties file based configurations and Jython script library

What this unit is about

This unit covers how to use property files with wsadmin to configure or update the WebSphere Application Server environment.

What you should be able to do

After completing this unit, you should be able to:

- Describe a scenario for administration by using property files
- Perform an application update with property files
- Define what a Jython script library is
- Explain where to find the required resources
- Describe the libraries that are available
- Explain how to combine the Jython script library into custom scripts

Unit objectives

After completing this unit, you should be able to:

- Describe a scenario for administration by using property files
- Perform an application update with property files
- Define what a Jython script library is
- Explain where to find the required resources
- Describe the libraries that are available
- Explain how to combine the Jython script library into custom scripts

© Copyright IBM Corporation 2013

Figure 16-1. Unit objectives

WA680 / VA6801.0

Notes:

Configuration repository: The issues

- The repository consists of multiple files in XML and other formats
- The configuration files are spread across many directories
- Configuration objects are complex
- Some configuration objects repeatedly stored in multiple files
- Example: properties for a JDBC provider

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1183122153343" name="Derby JDBC Provider">
  <classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</classpath>
  <factories xmi:type="resources.jdbc:DataSource" xmi:id="DataSource_1183122153625" name="D
    <propertySet xmi:id="J2EEResourcePropertySet_1183122153625">
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153625" name="databaseName" ty
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153626" name="shutdownDatabase"
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153627" name="dataSourceName"
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153628" name="description" typ
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153629" name="connectionAttribut
      <resourceProperties xmi:id="J2EEResourceProperty_1183122153630" name="createDatabase"
    </propertySet>
    <connectionPool xmi:id="ConnectionPool_1183122153631">
  </factories>
</resources.jdbc:JDBCProvider>
```

© Copyright IBM Corporation 2013

Figure 16-2. Configuration repository: The issues

WA680 / VA6801.0

Notes:

Properties file based configuration: A solution

- Properties files are more human readable
- Properties files consist of name and value pairs
- Decouples configuration data from changes in the underlying configuration model between releases
- Can be used with configuration archives
- Differences between configuration environments are easier to identify

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName
    jdbcprovider.props -configData Server=server1 filterMechanism
    • SELECTED_SUBTYPES -selectedSubTypes [JDBCProvider]')
```

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_118
<classpath>${DERBY_JDBC_DRIVER_PATH}/derby.jar</cla
<factories xmi:type="resources.jdbc:DataSource" xmi
<propertySet xmi:id="J2EEResourcePropertySet_1183
    <resourceProperties xmi:id="J2EEResourcePropert
    <resourceProperties xmi:id="J2EEResourcePropert
    <resourceProperties xmi:id="J2EEResourcePro
    <resourceProperties xmi:id="J2EEResourcePro
    <resourceProperties xmi:id="J2EEResourcePropert
    <resourceProperties xmi:id="J2EEResourcePropert
    <resourceProperties xmi:id="J2EEResourcePropert
    <resourceProperties xmi:id="J2EEResourcePropert
    <connectionPool xmi:id="ConnectionPool_1183122153
</factories>
</resources.jdbc:JDBCProvider>
```

```
ResourceType=JDBCProvider
ImplementingResourceType=JDBCProvider
ResourceId=Cell=!(cellName):Node=!(nodeName):Server
#
#Properties
classpath=${DERBY_JDBC_DRIVER_PATH}/derby.jar
name=Derby JDBC Provider (XA)
implementationClassName=org.apache.derby.jdbc.EmbeddedDriver
nativepath={}
description=Built-in Derby JDBC Provider (XA)
providerType=Derby JDBC Provider (XA) #readonly
xa=true #boolean
```

© Copyright IBM Corporation 2013

Figure 16-3. Properties file based configuration: A solution

WA680 / VA6801.0

Notes:

Properties file configuration content

- Each object is defined in a separate section:
 - Resource type and identifier
 - Configuration information
- Example: properties for a JDBC provider

```
# SubSection 1.0 # JDBCProvider attributes
#
# ResourceType=JDBCProvider
# ImplementingResourceType=JDBCProvider
# ResourceId=Cell!=!{cellName}:Node!=!{nodeName}:Server!=!
#   {serverName}:JDBCProvider=ID#JDBCProvider_1183122153343
#
# Properties
#
#   classpath={$DERBY_JDBC_DRIVER_PATH}/derby.jar}
#   name=Derby JDBC Provider
#   implementationClassName=
#     org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource
#   nativepath={}
#   description=Derby embedded non-XA JDBC Provider
#   providerType=Derby JDBC Provider #readonly
#   xa=false #boolean
```

1

Resource type and identifier

2

Configuration information

© Copyright IBM Corporation 2013

Figure 16-4. Properties file configuration content

WA680 / VA6801.0

Notes:

Properties file configuration commands

- **extractConfigProperties:** extracts configuration data into a properties file

```
wsadmin>AdminTask.extractConfigProperties('-propertiesFileName  
server1.props -configData Server=server1')
```

- **validateConfigProperties:** verifies that the properties in the properties file are valid

```
wsadmin>AdminTask.validateConfigProperties('-propertiesFileName  
server1.props')
```

- **applyConfigProperties:** applies properties in a specific properties file

```
wsadmin>AdminTask.applyConfigProperties('-propertiesFileName  
app.props')
```

- **deleteConfigProperties:** deletes objects in your configuration

```
wsadmin>AdminTask.deleteConfigProperties('-propertiesFileName  
thread.props')
```

- **createPropertiesFileTemplates:** creates template properties files

```
wsadmin>AdminTask.createPropertiesFileTemplates('-propertiesFileName  
app.props -configType Application')
```

© Copyright IBM Corporation 2013

Figure 16-5. Properties file configuration commands

WA680 / VA6801.0

Notes:

Benefits

- Allows configuration options to be specified with name-value pairs in a file, rather than as arguments to a script
 - Easy to see what is being set
 - There is no need to write a script that parses parameters and starts low-level configuration methods
- Can be used with configuration archives to replicate configuration information
- Offers a convenient way to compare multiple configurations to see differences

© Copyright IBM Corporation 2013

Figure 16-6. Benefits

WA680 / VA6801.0

Notes:

Format of a properties file

- Divided into subsections.
- Within each subsection, there are resource identifiers.
- For each resource identifier, name-value pairs are specified.
- The properties file does not display the cell, node, server, cluster, application, core group, or node group names.
 - Variables such as `!{cellName}` are included in the EnvironmentVariables section at the bottom of the properties file.
 - The Environment Variables section contains each variable in the properties file.

© Copyright IBM Corporation 2013

Figure 16-7. Format of a properties file

WA680 / VA6801.0

Notes:

Configuration properties file content

- Sample properties for JDBC provider
 - Contains resource identifier and name-value pairs

```
# SubSection 1.0 # JDBCProvider attributes
#
ResourceType=JDBCProvider
ImplementingResourceType=JDBCProvider
ResourceId=Cell={!cellName}:Node={!nodeName}:Server={!serverName}:JDBCProvider=ID#JDBCProvider_1183122153343
#
#Properties
classpath=${DERBY_JDBC_DRIVER_PATH}/derby.jar
name=Derby JDBC Provider
implementationClassName=org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource
nativepath={}
description=Derby embedded non-XA JDBC Provider
providerType=Derby JDBC Provider #readonly
xa=false #boolean
```

© Copyright IBM Corporation 2013

Figure 16-8. Configuration properties file content

WA680 / VA6801.0

Notes:

Format of a properties file

- Default values are shown as
`propertyName=PropertyValue #default (defaultValue)`
 - Example: `enable=true #default(false)`
- Required properties are shown as
`propertyName=PropertyValue #required;`
 - Example: `jndiName=myJndi #required`
- Properties that cannot be changed are shown as `propertyName=PropertyValue #readonly;`
 - Example: `providerType=stdProviderType #readonly`
- Valid values are shown as `propertyName=PropertyValue #type(range)`
 - The range is a list of values with a vertical bar (|) delimiter;
 - Example: `state=START #ENUM(START | STOP)`
- A single property can specify more than one of the tags `readonly`, `type`, `required`, and `default` after the `#` character.
 - Separate the tags by a comma (,)
 - Example: `enable=true #boolean,required,default(false)`
- For basic types such as string, int, or short, the format `name=value #type` is used.
 - Example: `port=9090 #int`
 - If the type is not specified, the string type is used.
- For a list or array types, the format `name={val1, val2, val3} #type` is used.
 - `type` is the type of each object in the list or array.

© Copyright IBM Corporation 2013

Figure 16-9. Format of a properties file

WA680 / VA6801.0

Notes:

Properties file configuration commands

- Properties file configuration is available with wsadmin, with five new methods of the AdminTask object:
 - extractConfigProperties**: extracts configuration data in the form of a properties file
 - validateConfigProperties**: verifies that the properties in the properties file are valid and can be safely applied to the new configuration
 - applyConfigProperties**: applies properties in a specific properties file to the configuration
 - deleteConfigProperties**: deletes properties in your configuration as designated in a properties file
 - createPropertiesFileTemplates**: creates template properties files to use to create or delete specific object types

© Copyright IBM Corporation 2013

Figure 16-10. Properties file configuration commands

WA680 / VA6801.0

Notes:

Details of extractConfigProperties

- Exports the most commonly used configuration data and attributes
- Converts the attributes to properties and saves the data to a file
- Specify the resources to extract with the target object or the configData parameter
 - The target object has format: `Node=nodeName:Server=serverName`
 - The `configData` parameter specifies a server, node, cluster, policy set, or application instance.
 - If no resource is specified, the command extracts the profile configuration data.
- Required parameters
 - `-propertiesFileName` - specifies the name of the properties file to extract to
- Optional parameters
 - `-configData` - specifies the configuration object instance
 - `-options` - specifies extra configuration options
 - `-filterMechanism` - specifies filter information for extracting properties
 - `-selectedSubTypes` - specifies the configuration properties to include or exclude when the command extracts the properties
 - Specified when the filterMechanism parameter is set to `NO_SUBTYPES` or `SELECTED_SUBTYPES`

© Copyright IBM Corporation 2013

Figure 16-11. Details of extractConfigProperties

WA680 / VA6801.0

Notes:

Details of validateConfigProperties

- Verifies that the properties in the properties file are valid and can be successfully applied to the new configuration
- Returns true if the system successfully validates the properties file
- Required parameters:
 - **-propertiesFileName** - specifies the name of the properties file to validate
- Optional parameters:
 - **-variablesMapFileName** - specifies the name of the variables map file
 - **-variablesMap** - specifies the values of the variables to use
 - **-reportFileName** - specifies the name of a report file
 - Changes to property values
 - No change to property values when value is same as defined in the properties file
 - No change to read-only property values
 - Exceptions
 - **-reportFilterMechanism** - specifies the type of report filter mechanism
 - All
 - Errors
 - Errors_And_Changes
 - **-zipFileName** - specifies the name of the .zip file that contains the policy sets that you want applied to the cell

© Copyright IBM Corporation 2013

Figure 16-12. Details of validateConfigProperties

WA680 / VA6801.0

Notes:

Details of applyConfigProperties

- **applyConfigProperties** - applies properties in a specific properties file to the configuration
 - Adds attributes if they do not exist. If the properties exist, the system sets new values.
- Required parameters:
 - **-propertiesFileName** - specifies the name of the properties file to apply
- Optional parameters:
 - **-variablesMapFileName** - specifies the name of the variables map file
 - This file contains values for variables that the system uses from the properties file.
 - **-variablesMap** - specifies the values of the variables to use with the properties file
 - **-reportFileName** - specifies the name of a report file
 - **-reportFilterMechanism** - specifies the type of report filter mechanism
 - All
 - Errors
 - Errors_And_Changes
 - **-validate** - specifies whether to validate the file before applying the changes
 - By default, the command validates the properties file.
 - **-zipFileName** - specifies the name of the .zip file that contains the policy sets that you want applied to the cell

© Copyright IBM Corporation 2013

Figure 16-13. Details of applyConfigProperties

WA680 / VA6801.0

Notes:

Details of deleteConfigProperties

- **deleteConfigProperties** - deletes properties in your configuration as designated in a properties file
- Required parameters
 - **propertiesFileName** - specifies the name of the properties file to apply
- Optional parameters
 - **variablesMapFileName** - specifies the name of the variables map file
 - This file contains values for variables that the system uses from the properties file
 - **variablesMap** - specifies the values of the variables to use with the properties file
 - **reportFileName** - specifies the name of a report file
 - **reportFilterMechanism** - specifies the type of report filter mechanism
 - All
 - Errors
 - Errors_And_Changes
 - **validate** - specifies whether to validate the file before applying the changes
 - By default, the command validates the properties file
 - **zipFileName** - specifies the name of the .zip file that contains the policy sets that you want applied to the cell

© Copyright IBM Corporation 2013

Figure 16-14. Details of deleteConfigProperties

WA680 / VA6801.0

Notes:

Details of `createPropertiesFileTemplates`

- **`createPropertiesFileTemplates`**: creates template properties files to use to create or delete specific object types
- Required parameters
 - **`-propertiesFileName`**
 - Specifies the name of the properties file where the template is stored
 - **`-configType`**
 - Specifies the resource type for the template to create
 - Specify **Server** to create a server type properties file template.
 - Specify **ServerCluster** to create a server cluster type properties file template.
 - Specify **Application** to create an application type properties file template.
 - Specify **AuthorizationGroup** to create an authorization group type properties file template.

© Copyright IBM Corporation 2013

Figure 16-15. Details of `createPropertiesFileTemplates`

WA680 / VA6801.0

Notes:

Comparing properties files and archives

- Properties file based configuration is not a replacement for configuration archives.
 - Properties file based configuration commands do not extract all configuration attributes, only the most commonly used.
 - Configuration archive contains a complete copy of the configuration and can be applied to another system to get an exact replica.
- Use properties file based configuration tool with configuration archives:
 - Use configuration archive tool to import a configuration archive.
 - Follow with properties file based configuration to make customizations.

© Copyright IBM Corporation 2013

Figure 16-16. Comparing properties files and archives

WA680 / VA6801.0

Notes:

Troubleshooting

- Properties file commands can generate reports that help identify problems.
- Use the **-reportFile** flag to create a report for any of the properties file configuration commands.
- Example:

```
AdminTask.validateConfigProperties  
  ('-propertiesFileName myprops.props -reportFile  
  myreport.txt')
```

© Copyright IBM Corporation 2013

Figure 16-17. Troubleshooting

WA680 / VA6801.0

Notes:

Modify existing server configuration

Description	Command
Extract the properties file for the configuration	<code>AdminTask.extractConfigProperties ('-propertiesFileName server1.props -configData Server=server1')</code>
Modify the file	Use your favorite text editor to change properties in <code>server1.props</code> and save the changes to the file
Validate the updated properties file	<code>AdminTask.validateConfigProperties ('-propertiesFileName server1.props')</code>
Apply the updated properties file to the configuration	<code>AdminTask.applyConfigProperties ('-propertiesFileName server1.props')</code>
Save changes	<code>AdminConfig.save()</code>

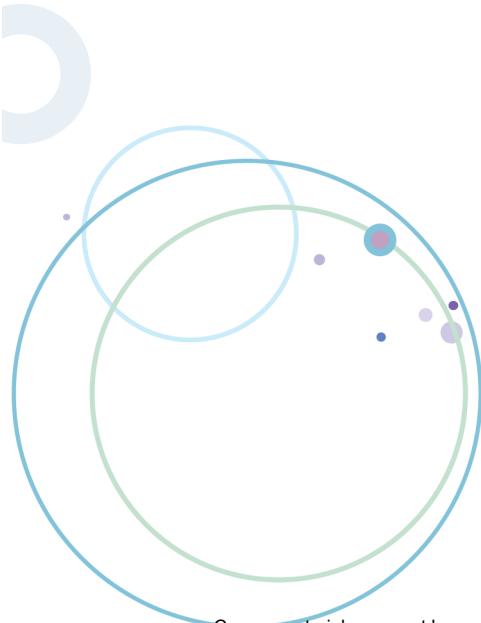
© Copyright IBM Corporation 2013

Figure 16-18. Modify existing server configuration

WA680 / VA6801.0

Notes:

Properties file based configuration examples



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 16-19. Properties file based configuration examples

WA680 / VA6801.0

Notes:

Delete a configuration object

Description	Command
Extract the properties file for a specific object type	<pre>AdminTask.extractConfigProperties (' -propertiesFileName threadPool.props -configData Server=server1 -filterMechanism SELECTED_SUBTYPES - selectedSubTypes [ThreadPool] ')</pre>
Modify the file to add the DELETE=true flag in the initial attribute section	<pre># # SubSection 1.0.1.4 # Thread pools # ResourceType=ThreadPool ImplementingResourceType=Server ResourceId=Cell=!{cellName}:Node=!{nodeName}:Server =!{serverName}:ThreadPoolManager=ID#ThreadPoolManager_1:ThreadPool=myThreadPool DELETE=true #</pre>
Delete the properties	<pre>AdminTask.deleteConfigProperties (' -propertiesFileName thread.props')</pre>
Save changes	<pre>AdminConfig.save()</pre>

© Copyright IBM Corporation 2013

Figure 16-20. Delete a configuration object

WA680 / VA6801.0

Notes:




Create a configuration object

Description	Command
Extract the properties file for a specific object type	AdminTask.extractConfigProperties('-propertiesFileName ds.props -configData Server=server1 -filterMechanism SELECTED_SUBTYPES -selectedSubTypes [DataSource]')
An existing data source section as a template, modify the file to contain: 1. A new resource ID 2. Updated properties for the data source	# SubSection 1.0.1.0 # DataSource attributes # ResourceType=DataSource ImplementingResourceType=JDBCProvider ResourceId=Cell={!{cellName}}:Node={!{nodeName}}:Server={!{serverName}}:JDBCProvider=ID#JDBCProvider_1183122153343:DataSource=ID# DataSource_99999 # #Properties # name= My DataSource ...
Apply the properties file	AdminTask.applyConfigProperties('-propertiesFileName thread.props')
Save changes	AdminConfig.save()

© Copyright IBM Corporation 2013

Figure 16-21. Create a configuration object

WA680 / VA6801.0

Notes:

Install an application

Description	Command
Create a properties file template for installing an application	<pre>AdminTask.createPropertiesFileTemplates('-propertiesFileName app.props -configType Application')</pre>
Update that the file app.props contains the require values for the application you are installing	<pre># ResourceType=Application ImplementingResourceType=Application ResourceId=Deployment= SKIP=true CreateDeleteCommandProperties=true # #Properties # EarFileLocation=location of earfile #required Name=appName #required TargetNode=targetnodeName #required TargetServer=targetServerName #required</pre>
Apply the properties file	<pre>AdminTask.applyConfigProperties('-propertiesFileName app.props')</pre>
Save changes	<pre>AdminConfig.save()</pre>

© Copyright IBM Corporation 2013

Figure 16-22. Install an application

WA680 / VA6801.0

Notes:

Jython script library



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 16-23. Jython script library

WA680 / VA6801.0

Notes:



Disadvantage of Jython scripting language

- Jython is a powerful scripting language; however, it can be difficult for new users.
- Similar to learning a new programming language.
- Script developer must be familiar with the Java libraries to take full advantage.
- Multiple syntaxes are used to start the same activities.
- Presents a barrier to learning WebSphere.

© Copyright IBM Corporation 2013

Figure 16-24. Disadvantage of Jython scripting language

WA680 / VA6801.0

Notes:



Script library

- Provides the full administrative capability to manage and control the WebSphere Application Server installation:
 - Application management
 - Configuration
 - Runtime operations
- Available as is
- Reduces the requirement to learn Jython scripting language
- Can be run in batch, command line, or interactive mode
- Reduces common errors in scripts
- Can be modified to work with WebSphere Application Server 6 and higher
- Supports rapid development of new scripts by combining procedures from the wsadmin script library with custom code developed by the user

© Copyright IBM Corporation 2013

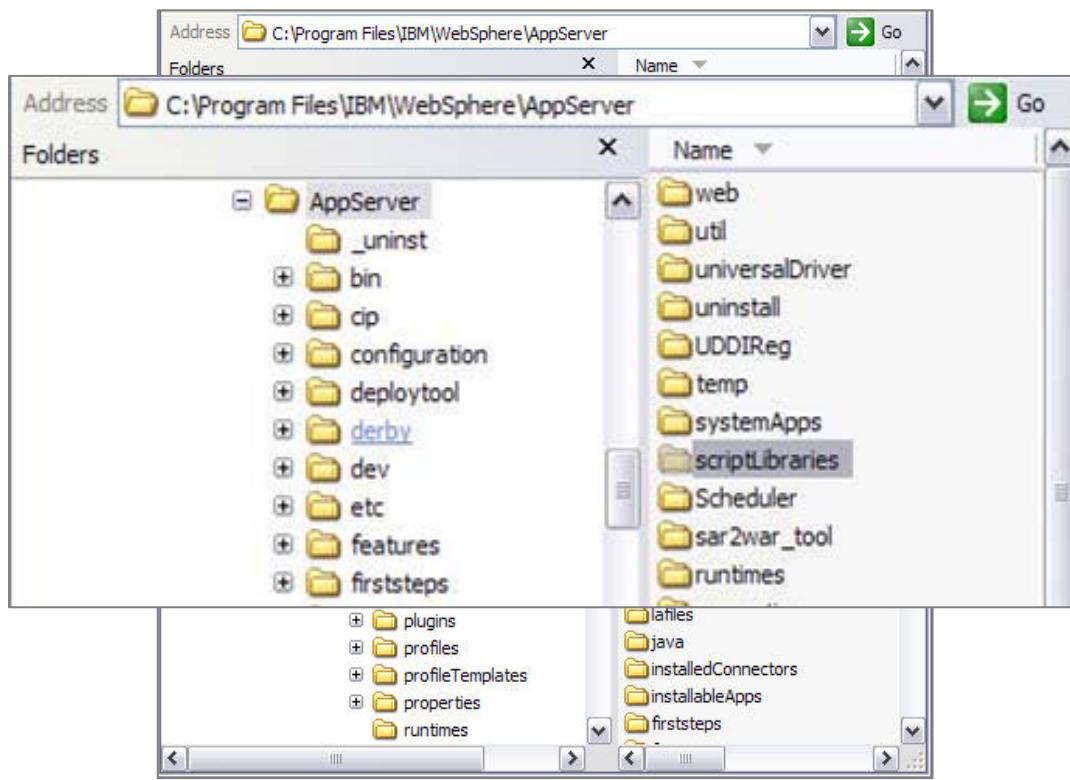
Figure 16-25. Script library

WA680 / VA6801.0

Notes:



Script library location



© Copyright IBM Corporation 2013

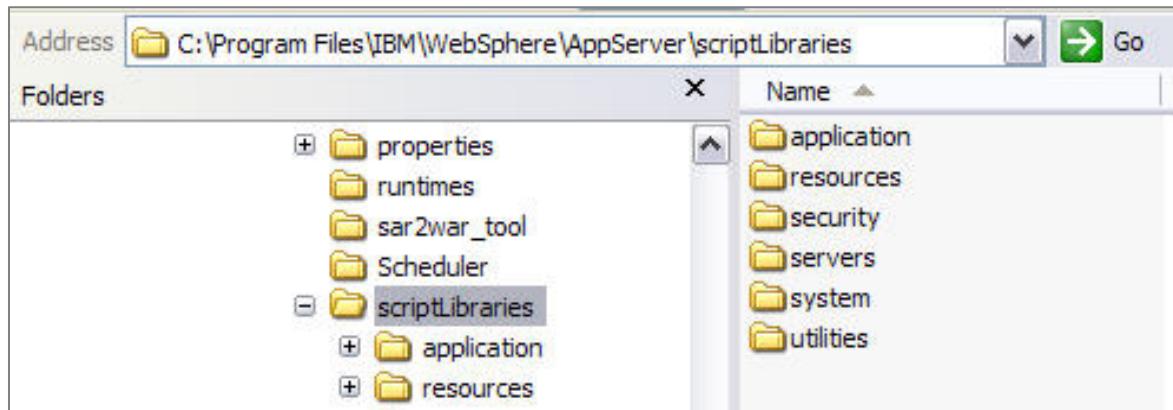
Figure 16-26. Script library location

WA680 / VA6801.0

Notes:

WebSphere Education

Script libraries



© Copyright IBM Corporation 2013

Figure 16-27. Script libraries

WA680 / VA6801.0

Notes:



Application script library

- Contains scripts to:
 - Install
 - Uninstall
 - Update
 - Modify
 - Export
 - Start/Stop
- Also contains scripts to work with business level applications
- Examples:
 - `installAppWithDefaultBindingOption`
 - `installAppWithNodeAndServerOptions`
 - `uninstallApplication`
 - `listApplications`
 - `updateApplicationUsingDefaultMerge`

© Copyright IBM Corporation 2013

Figure 16-28. Application script library

WA680 / VA6801.0

Notes:



Resources script library

- Can be used to list, create, and install
 - J2C
 - JDBC
 - JMS
 - Mail
 - JAAS authentication alias
 - Shared libraries
- Examples:
 - `createJ2CActivationSpec`
 - `createJDBCProvider`
 - `createDataSource`
 - `createJMSPublisher`
 - `createWASQueue`
 - `createMailProvider`

© Copyright IBM Corporation 2013

Figure 16-29. Resources script library

WA680 / VA6801.0

Notes:



Security script library

- Can be used to list, create, and delete:
 - Application users
 - Administrative users
- Examples:
 - `createAuthorizationGroup`
 - `mapUsersToAdminRole`
 - `mapGroupsToAdminRole`
 - `listResourcesForUserID`
 - `removeUserFromAllAdminRoles`

© Copyright IBM Corporation 2013

Figure 16-30. Security script library

WA680 / VA6801.0

Notes:



Servers script library

- Contains two scripts:
 - Admin Server Management
 - Admin Cluster Management
- Admin Server Management
 - Contains scripts to list, create, delete, start, and stop the application servers
 - It also contains scripts to configure the JVM properties of the application server
- Examples:
 - `createApplicationServer`
 - `startAllServers`
 - `setJVMProperties`
- Admin Cluster Management
 - Contains scripts to list, create, delete, start, ripple start, and stop the application server cluster
- Examples:
 - `createClusterWithFirstMember`
 - `startSingleCluster`
 - `immediateStopSingleCluster`

© Copyright IBM Corporation 2013

Figure 16-31. Servers script library

WA680 / VA6801.0

Notes:

System script library

- System script library
 - Provides scripts for managing nodes and node groups
 - Provides the capabilities to manage the infrastructure remotely without having to create complex Jython scripts
- Examples:
 - `listNodes`
 - `doesNodeExist`
 - `isNodeRunning`
 - `stopNodeAgent`

© Copyright IBM Corporation 2013

Figure 16-32. System script library

WA680 / VA6801.0

Notes:



Utilities script library

- Utilities script library:
 - Two scripts are provided: AdminLibHelp and AdminUtilities
 - Provides scripts for obtaining help on the script library, and miscellaneous scripts
- Examples:
 - **Save**
 - **getScriptLibraryList**
 - **getScriptLibraryPath**
 - **help**

© Copyright IBM Corporation 2013

Figure 16-33. Utilities script library

WA680 / VA6801.0

Notes:

Script library help

- **AdminLibHelp.help()**
 - The **AdminLibHelp** provides general help information for the Jython script libraries for the wsadmin tool
- **<Script Lib>.help()**
 - Example: **AdminServerManagement.help()**
 - Provides a detailed explanation of the script library
 - Provides all of the procedures available in the script library
- **<Script Lib>.help("procedure name")**
 - Example: **AdminServerManagement.help("configureThreadPool")**
 - Provides:
 - Required arguments
 - Optional arguments
 - Description
 - Usage examples

© Copyright IBM Corporation 2013

Figure 16-34. Script library help

WA680 / VA6801.0

Notes:

Sample script procedures

```
#-----  
# AdminServerManagementr.py - Jython procedures for performing server management tasks.  
#-----  
#  
# This script includes the following server management procedures:  
#  
# Group 1: ServerConfiguration  
#   Ex1: listServers  
#   Ex2: listServerTypes  
#   Ex3: listServerTemplates  
#   EX4: createApplicationServer  
#   Ex5: createAppServerTemplate  
#   Ex6: createGenericServer  
#   Ex7: createWebServer  
#   Ex8: deleteServer  
#   Ex9: deleteServerTemplate  
#   Ex10: startAllServers  
#   Ex11: startSingleServer  
#   Ex12: stopAllServers  
#   Ex13: stopSingleServer  
#   Ex14: listJVMProperties  
#   Ex15: showServerInfo  
#   Ex16: getJavaHome  
#   Ex17: setJVMProperties
```

© Copyright IBM Corporation 2013

Figure 16-35. Sample script procedures

WA680 / VA6801.0

Notes:

Sample script (1 of 2)

```
## Example 4 Create a new application server ##
def createApplicationServer( nodeName, serverName, templateName, failonerror=AdminUtilities._BLANK_ ):
    if (failonerror==AdminUtilities._BLANK_):
        failonerror=AdminUtilities._FAIL_ON_ERROR_
    #endif
    msgPrefix = "createApplicationServer(\""+nodeName+"\"," + " "+serverName+", "+templateName+", "+failo

try:
    #
    # Create a new application server
    #
    print "-----"
    print " AdminServerManagement: Create an application server on a given node"
    print " Node name:           "+nodeName
    print " New Server name:     "+serverName
    print " Optional parameter: "
    print "      Template name: "+templateName
    print " Usage: AdminServerManagement.createApplicationServer(\""+nodeName+"\"," + " "+serverName+
    print "-----"
    print " "
    print " "
```

© Copyright IBM Corporation 2013

Figure 16-36. Sample script (1 of 2)

WA680 / VA6801.0

Notes:

Sample script (2 of 2)

```
# Check the required parameters
if (nodeName == ""):
    raise AttributeError(AdminUtilities._formatNLS(resourceBundle, "WASL6041E", ["nodeName", nc

if (serverName == ""):
    raise AttributeError(AdminUtilities._formatNLS(resourceBundle, "WASL6041E", ["serverName", nc

# Check if node exists
node = AdminConfig.getId("/Node:" + nodeName + "/")
if (len(node) == 0):
    # Node does not exist
    raise AttributeError(AdminUtilities._formatNLS(resourceBundle, "WASL6040E", ["nodeName", nod
#endif

# Check if server exists
server = AdminConfig.getId("/Node:" + nodeName + "/Server:" + serverName)
if (len(server) > 0):
    # Server already exists
    raise AttributeError(AdminUtilities._formatNLS(resourceBundle, "WASL6046E", [serverName]))
#endif

# Construct required parameters
requiredParamList = ['-name', serverName]
```

© Copyright IBM Corporation 2013

Figure 16-37. Sample script (2 of 2)

WA680 / VA6801.0

Notes:

Combining script library with Jython scripts

- Use a text editor to combine several scripts from the Jython script library, as the following example displays:

```
#  
# My Custom Jython Script - file.py  
#  
AdminServerManagement.createApplicationServer("myNode", "Server1", "default")  
AdminServerManagement.createApplicationServer("myNode", "Server2", "default")  
  
# Use one of them as the first member of a cluster  
AdminClusterManagement.createClusterWithFirstMember("myCluster",  
"APPLICATION_SERVER", "myNode", "Server1")  
  
# Add a second member to the cluster  
AdminClusterManagement.createClusterMember("myCluster", "myNode", "Server3")  
  
# Install an application  
AdminApplication.installAppWithClusterOption("DefaultApplication",  
"..\\installableApps\\DefaultApplication.ear",  
"myCluster")  
  
# Start all servers and applications on the node  
AdminServerManagement.startAllServers("myNode")
```

© Copyright IBM Corporation 2013

Figure 16-38. Combining script library with Jython scripts

WA680 / VA6801.0

Notes:



Custom script library

- To deploy a custom script library:
 1. Create custom script with any number of required procedures.
 2. Create a subfolder under <was_root>/scriptLibraries/
 3. Place new custom script in newly created folder.
 4. Start wsadmin. This action automatically loads a new script library.
 5. Start new procedures.

Example: `XYZServerManagement.startXYZservers()`

© Copyright IBM Corporation 2013

Figure 16-39. Custom script library

WA680 / VA6801.0

Notes:

Recommendations for using the script library

- Do not modify the original script library; instead copy the content from the script library and modify the script to suit your environment.
- Try to use commands in the script library first before using one of commands from the administrative objects such as AdminTask, and AdminConfig.

© Copyright IBM Corporation 2013

Figure 16-40. Recommendations for using the script library

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Describe a scenario for administration by using property files
- Perform an application update with property files
- Define what a Jython script library is
- Explain where to find the required resources
- Describe the libraries that are available
- Explain how to combine the Jython script library into custom scripts

© Copyright IBM Corporation 2013

Figure 16-41. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. True or False: Editing XML files is a preferred method for updating the WebSphere repository.
2. True or False: Property files can be compared to highlight changes to the WebSphere repository.
3. Describe one process for use of the WebSphere Jython script library.

© Copyright IBM Corporation 2013

Figure 16-42. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.

Checkpoint answers

1. True or False: Editing XML files is a preferred method for updating the WebSphere repository.

Answer: False. Editing property files and applying is preferred to editing XML files directly.

2. True or False: Property files can be compared to highlight changes to the WebSphere repository.

Answer: True. Property files can be used to compare changes over points in time or in different environments.

3. Describe one process for use of the WebSphere Jython script library.

Answer: There are several recommended practices for using the script library.

1. Copy a script and then modify it as needed.
2. Read the scripts for a guide to using Jython scripts for WebSphere administration.
3. Find a script that can be used "as is" from the script library.

© Copyright IBM Corporation 2013

Figure 16-43. Checkpoint answers

WA680 / VA6801.0

Notes:

Unit 17. Using scripts with the job manager and CIM

What this unit is about

This unit describes how to use scripting commands to submit jobs to a job manager.

What you should be able to do

After completing this unit, you should be able to:

- Describe the flexible management topology
- Describe the function of the job manager and the administrative agent
- Use scripts to create job manager and administrative agent profiles
- Use scripts to configure target hosts for the job manager
- Use scripts to configure CIM jobs for the job manager
- Use scripts to submit a job to install the Installation Manager on a remote host
- Use scripts to install the WebSphere Application Server and create an application server profile
- Use scripts to monitor the status of a job manager job
- Automate the installation of WebSphere Application Server

How you will check your progress

- Checkpoint questions

References

WebSphere Application Server V8.5 Information Center

Unit objectives

After completing this unit, you should be able to:

- Describe the flexible management topology
- Describe the function of the job manager and the administrative agent
- Use scripts to create job manager and administrative agent profiles
- Use scripts to configure target hosts for the job manager
- Use scripts to configure CIM jobs for the job manager
- Use scripts to submit a job to install the Installation Manager on a remote host
- Use scripts to install the WebSphere Application Server and create an application server profile
- Use scripts to monitor the status of a job manager job
- Automate the installation of WebSphere Application Server

© Copyright IBM Corporation 2013

Figure 17-1. Unit objectives

WA680 / VA6801.0

Notes:

Topics

- Job manager profile and target hosts
- Centralized Installation Manager (CIM) in the job manager
- Creating a flexible management environment
- Automate the installation of WebSphere Application Server

© Copyright IBM Corporation 2013

Figure 17-2. Topics

WA680 / VA6801.0

Notes:

17.1.Job manager profile and target hosts

Job manager profile and target hosts



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 17-3. Job manager profile and target hosts

WA680 / VA6801.0

Notes:

Flexible management

- The flexible management model addresses scalability requirements with the introduction of the job manager
 - A single job manager can manage many WebSphere Application Server base edition and Network Deployment topologies that are registered with it as managed nodes
- In this model, rather than starting administrative operations directly against stand-alone servers and deployment managers, they are instead submitted as administrative jobs to the job manager
- The managed nodes then fetch these administrative jobs at predefined intervals
- A job manager supports
 - Configuration and control (start/stop) operations for servers, clusters, and applications
 - Execution of `wsadmin` scripts as administrative jobs

© Copyright IBM Corporation 2013

Figure 17-4. Flexible management

WA680 / VA6801.0

Notes:

The flexible management model addresses scalability requirements with the introduction of the job manager. A single job manager can manage several WebSphere Application Server Base edition and Network Deployment cell topologies that are registered with it as managed nodes. In this model, rather than using administrative operations directly against stand-alone servers and deployment managers, they are instead submitted as administrative jobs to the job manager. The managed nodes then fetch these administrative jobs at predefined intervals. A job manager supports configuration and control (start and stop) operations for servers, clusters, and applications. It also supports execution of `wsadmin` scripts as administrative jobs.

Flexible management characteristics

- Offers asynchronous job-queuing mechanism for administration purposes
- An alternative to the Network Deployment cell model, not a replacement
- Offers administrators more management options not available before WebSphere Application Server V7
- Management of multiple base servers
 - Can manage a server farm that contains hundreds of base servers
- Coordinate management actions across multiple deployment managers
 - Can manage multiple cells
 - Geographically distributed cells

© Copyright IBM Corporation 2013

Figure 17-5. Flexible management characteristics

WA680 / VA6801.0

Notes:

Flexible management environments rely on asynchronous processing of work units (known as jobs) from the job manager. This approach supports large scaling and can support many application servers without degrading performance. It also reduces latency and bandwidth requirements on the network; even dialup lines to remote sites can work well without slowing down the overall system. Additionally, configuration information does not exist beyond the node level, so no bottleneck is associated with accessing a master configuration repository.

Flexible management is not a replacement for the network deployment model but can be used as an alternative to it. The two models can be combined by having a job manager coordinate management activities across multiple deployment managers.

Job manager

- Profile type that supports Centralized Installation Manager (CIM) and flexible management
- Targets or host computers can be registered with the job manager
- To manage stand-alone application servers, use administrative agent to register servers with the job manager
- To manage multiple cells, register deployment managers with job manager directly
- Use job manager to queue and schedule jobs for registered targets, stand-alone application servers, or cells

© Copyright IBM Corporation 2013

Figure 17-6. Job manager

WA680 / VA6801.0

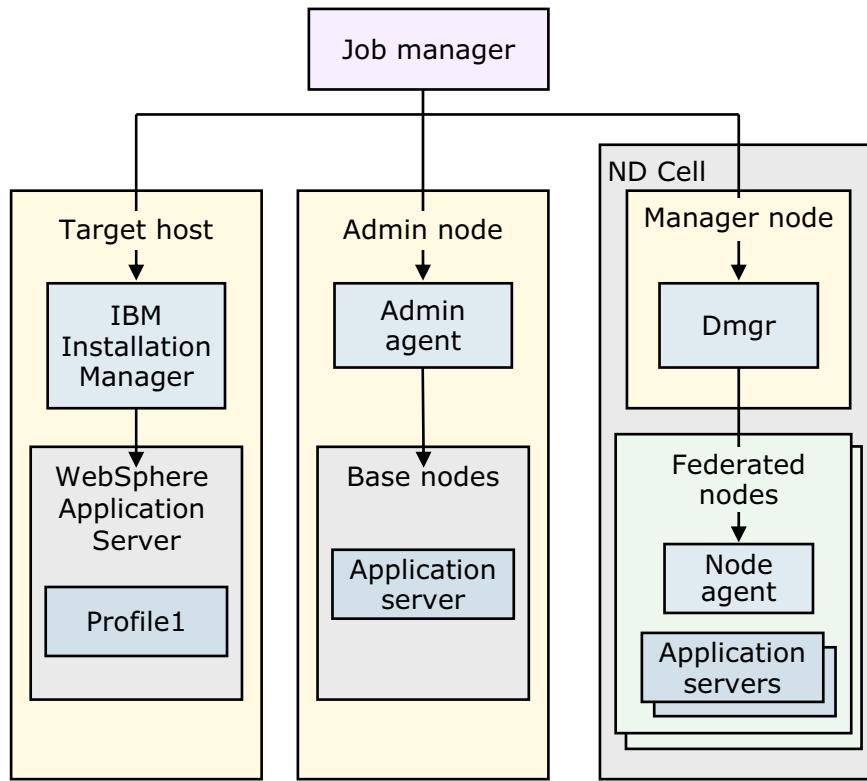
Notes:

The job manager is a new server type that was added to support flexible management. It is a new profile type, and the various tools that can create profiles are modified to support creation and maintenance of this profile. The job manager is central to flexible management.

To participate in flexible management, a base server first registers itself with the administrative agent. The base server must then register with the job manager. If a deployment manager wants to participate in an environment that a job manager controls, the deployment manager registers directly with the job manager; no administrative agent is involved in this case.

The main use of the job manager is to queue jobs to application servers in a flexible management environment. These queued jobs are pulled from the job manager by the administrative agent or deployment manager and distributed to the appropriate application server or servers.

Job manager topology



© Copyright IBM Corporation 2013

Figure 17-7. Job manager topology

WA680 / VA6801.0

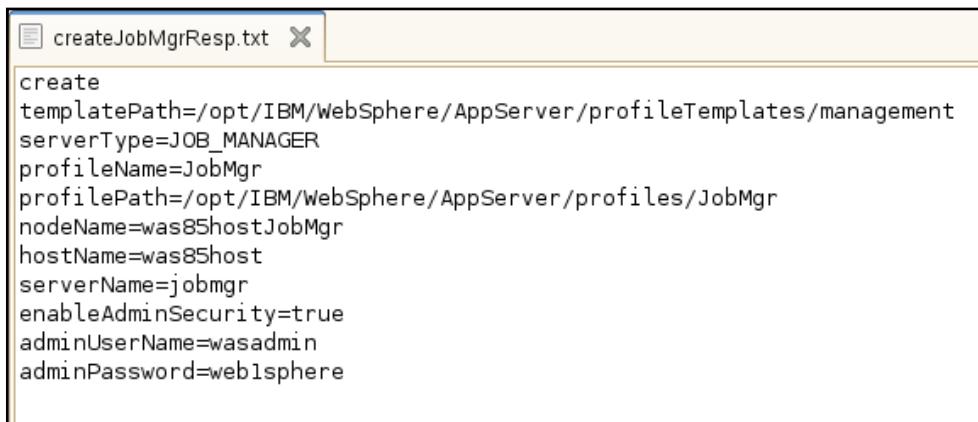
Notes:

The job manager is designed to complement the Network Deployment and Base edition topologies. Existing nodes do not need to be reconfigured. A single job manager can manage hundreds of nodes, and multiple job managers can manage a single node. This model is different from having ownership of a managed node by the deployment manager in a network deployment topology.

- The topologies that a job manager controls maintain their autonomy, including their security configuration, and thus they can be directly managed with existing administrative processes, such as scripts or the administrative console.
- This model allows coordinated management actions across multiple managed nodes that are defined as a group.
- The asynchronous job submission model facilitates the management of nodes that are geographically dispersed and reachable only through low-bandwidth, high-latency networks.

Creating the job manager profile

- Use the manageprofiles command to create the job manager profile
- First, create a response file
 - The profile template is **management**
 - The server type is **JOB_MANAGER**



```
createJobMgrResp.txt X
create
templatePath=/opt/IBM/WebSphere/AppServer/profileTemplates/management
serverType=JOB_MANAGER
profileName=JobMgr
profilePath=/opt/IBM/WebSphere/AppServer/profiles/JobMgr
nodeName=was85hostJobMgr
hostName=was85host
serverName=jobmgr
enableAdminSecurity=true
adminUserName=wasadmin
adminPassword=websphere
```

- Run the command:

```
./manageprofiles.sh -response <scripts>/createJobMgrResp.txt
```

© Copyright IBM Corporation 2013

Figure 17-8. Creating the job manager profile

WA680 / VA6801.0

Notes:

The manageprofiles command is run from the `<was_root>/bin` directory.

The management profile template is used with the `-serverType` parameter to indicate the type of management profile.

When the command completes successfully, you see a message as follows:

`"INSTCONFSUCCESS: Success: Profile JobMgr now exists. Please consult <profile_root>/JobMgr/logs/AboutThisProfile.txt for more information about this profile."`

Start the job manager by using the command

`<profile_root>/JobMgr/bin/startServer.sh jobmgr`

The `AboutThisProfile.txt` file shows the ports that are assigned to the `jobmgr` process. In particular, the following ports are important to take note of. The port values shown here are the defaults; the ports on your system might be different.

Administrative console port: 9960

Administrative console secure port: 9943

Management SOAP connector port: 8876

For example, to access the job manager's administrative console, use the web address
<http://localhost:9960.ibm/console>.

JobManagerNode command group for the AdminTask object

- Use the commands and parameters in the JobManagerNode group to register targets with the job manager.
- The **registerHost** command defines a remote host target to the job manager.
 - There are no software requirements for this host beyond its operating system.
- Use the **registerHost** command to:
 - Register a remote host target with the job manager.
 - Collect an inventory of the remote host. Information regarding managed resources and job types is available upon successful completion of the `registerHost` command.

© Copyright IBM Corporation 2013

Figure 17-9. JobManagerNode command group for the AdminTask object

WA680 / VA6801.0

Notes:

The `registerHost` command defines a remote host target to the job manager.

Use the `registerHost` command to:

Register a remote host target with the job manager. Unlike targets that are WebSphere Application Server profiles and are registered by using the `registerWithJobManager` command at the deployment manager or administrative agent, a remote host target is not required to have any WebSphere Application Server products installed. There are no software requirements for this host beyond its operating system.

Collect an inventory of the remote host. Information regarding managed resources and job types is available upon successful completion of the `registerHost` command.

Registering host computers with job managers

- Use the `AdminTask.registerHost()` Jython command to register a remote host to the job manager
- For example:

```
AdminTask.registerHost('[-host was85host -hostProps [ [osType linux] [username root]
[password websphere] [saveSecurity true]]]')
```

- The required properties are `-host` and `-hostProps`
- Host properties include:
 - `osType`
 - `username`
 - `password`
 - `privateKeyFile`
 - `passphrase`
 - `saveSecurity`

© Copyright IBM Corporation 2013

Figure 17-10. Registering host computers with job managers

WA680 / VA6801.0

Notes:

To run the Jython command that is shown on this slide, the job manager must be started, and wsadmin must be connected with the job manager.

saveSecurity: Specifies whether to store security properties (user name, password, `privateKeyFile`, `passphrase`) with the host and used as default values for job submissions. If this property is given a value of `true`, then the security properties are stored with the host and used for subsequent job submissions to this host.

Submitting administrative jobs by using the job manager

- Use the `submitJob` command to submit administrative jobs.
- Job submissions consist of the following parameters:
 - **jobType**: Specifies the type of job to run.
 - **targetList**: Specifies one or more managed targets where the job runs.
 - **jobParams**: Specifies extra configuration information. Parameters are specific to each job type.
- When the job is submitted, a job ID or job token is returned
 - The job ID can be used to query the status of the submitted job
- Many job types exist in the CIM and flexible management environment such as:
 - Product installation and maintenance
 - Application management
 - Configuration
 - Application server runtime control

© Copyright IBM Corporation 2013

Figure 17-11. Submitting administrative jobs by using the job manager

WA680 / VA6801.0

Notes:

In a flexible management environment, you can configure an administrative agent and job manager to submit jobs to multiple targets or target groups in your configuration. Then, you can submit administrative jobs to queue jobs across your managed environment. Each administrative job has a corresponding job type, which defines the required parameters to submit the job. You can use the commands in the `AdministrativeJobs` command group to submit administrative jobs to the job manager.

Testing connection to a remote target (1 of 2)

- Use the `testConnection` job type to test connection to a remote target host
 - When submitted, this job returns a job token such as 136362276093945262
 - The job token can be used to submit other queries such as `queryJobs` and `getOverallJobStatus`
- For example:

```
AdminTask.submitJob('[-jobType testConnection -targetList [was85host ]]')
```

```
AdminTask.queryJobs('[-query jobToken=136362276093945262 -maxReturn 1]')
```

```
AdminTask.getOverallJobStatus('[-jobTokenList [136362276093945262]]')
```

© Copyright IBM Corporation 2013

Figure 17-12. Testing connection to a remote target (1 of 2)

WA680 / VA6801.0

Notes:

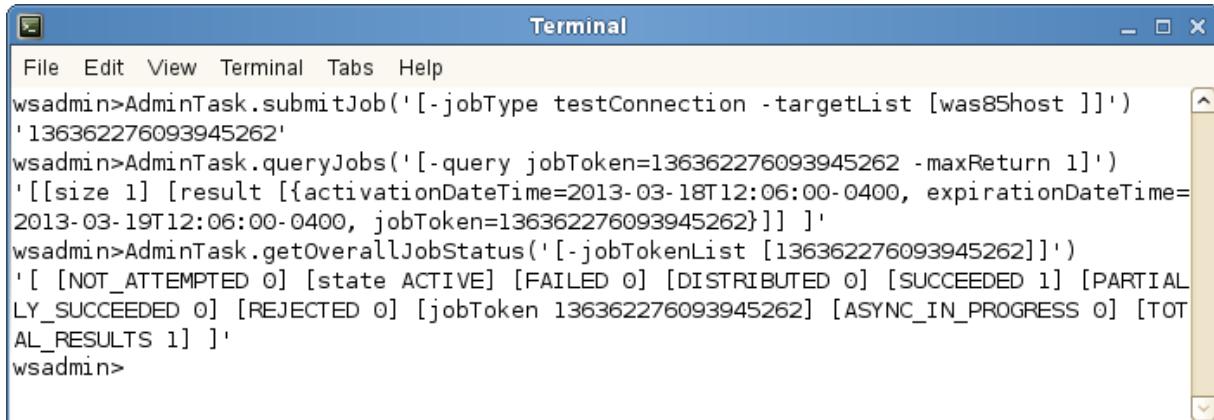
To run the Jython commands that are shown on this slide, the job manager must be started, and wsadmin must be connected with the job manager.

The job token is the job ID for the submitted job. You can use the job ID to query, suspend, resume, or delete the job.

If the target cannot be connected to, you might see a message such as, CWWSY0102E:
Target with name <host_name> was not found.

Testing connection to a remote target (1 of 2)

- Output from the testConnection, query, and status commands



```
Terminal
File Edit View Terminal Tabs Help
wsadmin>AdminTask.submitJob('[-jobType testConnection -targetList [was85host ]]')
'136362276093945262'
wsadmin>AdminTask.queryJobs('[-query jobToken=136362276093945262 -maxReturn 1]')
'[ [size 1] [result [{activationDateTime=2013-03-18T12:06:00-0400, expirationDateTime=
2013-03-19T12:06:00-0400, jobToken=136362276093945262}]] ]'
wsadmin>AdminTask.getOverallJobStatus('[-jobTokenList [136362276093945262]]')
'[ [NOT_ATTEMPTED 0] [state ACTIVE] [FAILED 0] [DISTRIBUTED 0] [SUCCEEDED 1] [PARTIAL-
LY_SUCCEEDED 0] [REJECTED 0] [jobToken 136362276093945262] [ASYNC_IN_PROGRESS 0] [TOT-
AL_RESULTS 1] ]'
wsadmin>
```

- **queryJobs** returns job scheduling data
- **getOverallJobStatus** returns data about the state of the job:
ACTIVE, SUCCEEDED, FAILED, and other job states

© Copyright IBM Corporation 2013

Figure 17-13. Testing connection to a remote target (1 of 2)

WA680 / VA6801.0

Notes:

The **queryJobs** command queries the job manager for each submitted job.

Use the **getOverallJobStatus** command to display the overall job status for a specific job or a list of jobs.

The command returns job status information for the job or jobs of interest. The system displays the following information in the overall job status:

- The STATE attribute specifies the current state of the job.
- The TOTAL_RESULTS attribute specifies the total number of jobs.
- The DISTRIBUTED attribute specifies the number of distributed jobs.
- The ASYNC_IN_PROGRESS attribute specifies the number of asynchronous jobs in progress.
- The SUCCEEDED attribute specifies the number of successful jobs.

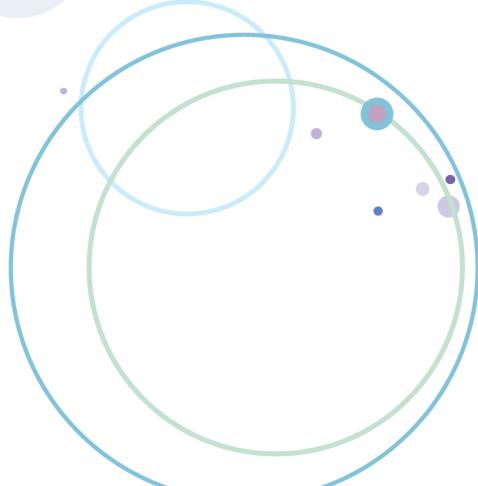
- The PARTIALLY_SUCCEEDED attribute specifies the number of partially successful jobs. Partial success can occur, for example, when a node represents multiple servers, and only some of the servers on the node complete successfully.
- The FAILED attribute specifies the number of failed jobs.
- The REJECTED attribute specifies the number of rejected jobs.
- The NOT_ATTEMPTED attribute specifies the number of jobs that the system has not attempted.

Job status results typically progress from DISTRIBUTED to ASYNC_IN_PROGRESS to SUCCEEDED. You might must run a getJobTargetStatus command more than once, until the result is FAILED, REJECTED, or SUCCEEDED.

By default, a job remains active for one day (24 hours).

17.2.Centralized Installation Manager (CIM) in the job manager

Centralized Installation Manager (CIM) in the job manager



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

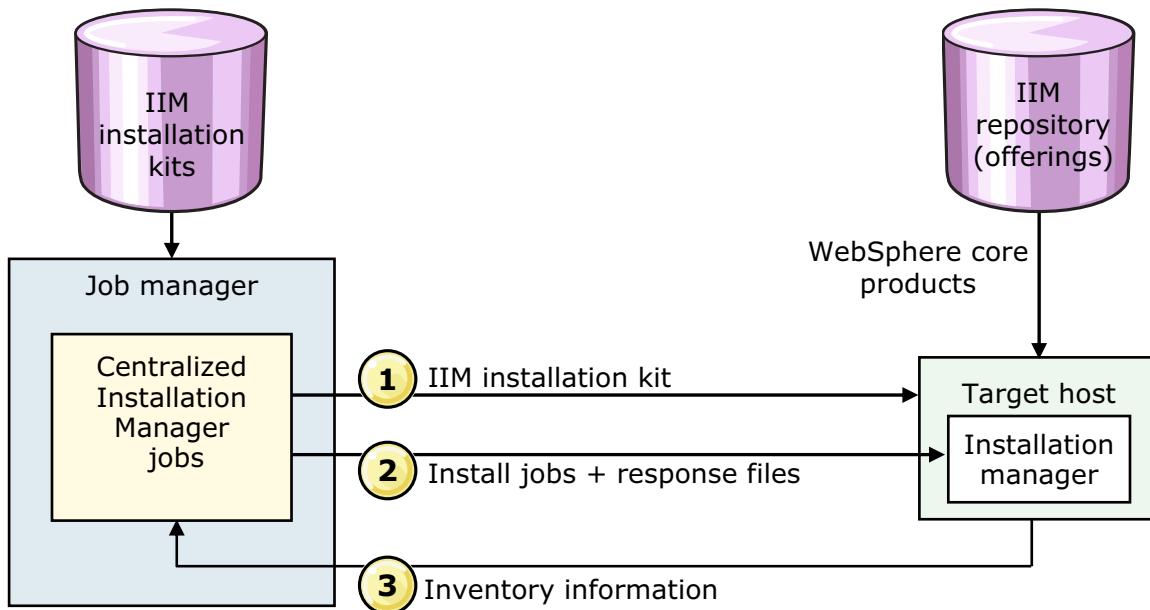
Figure 17-14. Centralized Installation Manager (CIM) in the job manager

WA680 / VA6801.0

Notes:

Centralized Installation Manager (CIM) in the job manager

- Architecture of job manager, target hosts, and IBM Installation Manager (IIM) repository



© Copyright IBM Corporation 2013

Figure 17-15. Centralized Installation Manager (CIM) in the job manager

WA680 / VA6801.0

Notes:

CIM functions are now a series of jobs available to a stand-alone job manager and the deployment manager's job manager. The job manager can store IIM installation kits for different operating systems in a repository on the local file system. A job can be submitted that "pushes" the required IBM Installation Manager kit to the target node and perform the remote installation with nothing more than target identification and the required authorizations. No agents are required on the target host. Installation jobs for WebSphere Application Server can be configured along with an appropriate response file for the remote target host. When installed on the target host, the Installation Manager can pull the WebSphere code from an IIM repository.

You can submit the **Inventory job** to refresh data on job types and on managed resources of the job manager. Resources include applications and servers of each target. If you installed a product that adds job types on a managed target, run the Inventory job to refresh data on job types and target resources. You can view the refreshed data in the job manager console.

Centralized installation manager (CIM) functions

- For Version 8.0 and later, centralized installation manager (CIM) functions are accessed through the job manager.
- Using the job manager, you can run the following functions:
 - Install, update, and uninstall WebSphere Application Server offerings on remote hosts
 - Install, update, and uninstall IBM Installation Manager on remote hosts.
 - Collect, distribute, and delete files on remote hosts
 - Run scripts on remote hosts
- CIM jobs
 - Test connection
 - Manage offering
 - Manage profile
 - Install SSH public key

© Copyright IBM Corporation 2013

Figure 17-16. Centralized installation manager (CIM) functions

WA680 / VA6801.0

Notes:

Install, update, and uninstall IBM Installation Manager on remote machines. This process is not supported on z/OS targets. For z/OS targets, you must install Installation Manager before working with CIM.

Important note about uninstalling IBM Installation Manager: Before you can uninstall IBM Installation Manager, you must uninstall all the products that it installed. Consequently, you cannot uninstall the Installation Manager from the host machine of the local deployment manager.

Manage offering: Install, update, uninstall WebSphere Application Server

Manage profile: Create, delete, augment, back up, list, and restore profiles on remote hosts

Install SSH public key

Distribute, collect, and delete file

Test connection: Verify access to remote hosts registered with the job manager

Installing the Installation Manager on a remote host

- The `installIM` administrative job installs a new instance of Installation Manager by sending an installation kit from the job manager to the hosts.
- For example:

```
AdminTask.submitJob('-jobType installIM -targetList [was85host1 was85host2 was85host3]
-jobParams [acceptLicense TRUE]
[kitPath /opt/IBM/WebSphere/AppServer/profiles/JobMgr/IMKits/agent.installer.linux153]
-username user1 -password pw')
```

- Job parameters
 - **acceptLicense**: Must be set to `true`
 - **kitPath**: Location of the Installation Manager kit
 - **skipPrereqCheck**: The default value is `false`. Prerequisite checking and disk space checking are done unless the `true` value is specified.

© Copyright IBM Corporation 2013

Figure 17-17. Installing the Installation Manager on a remote host

WA680 / VA6801.0

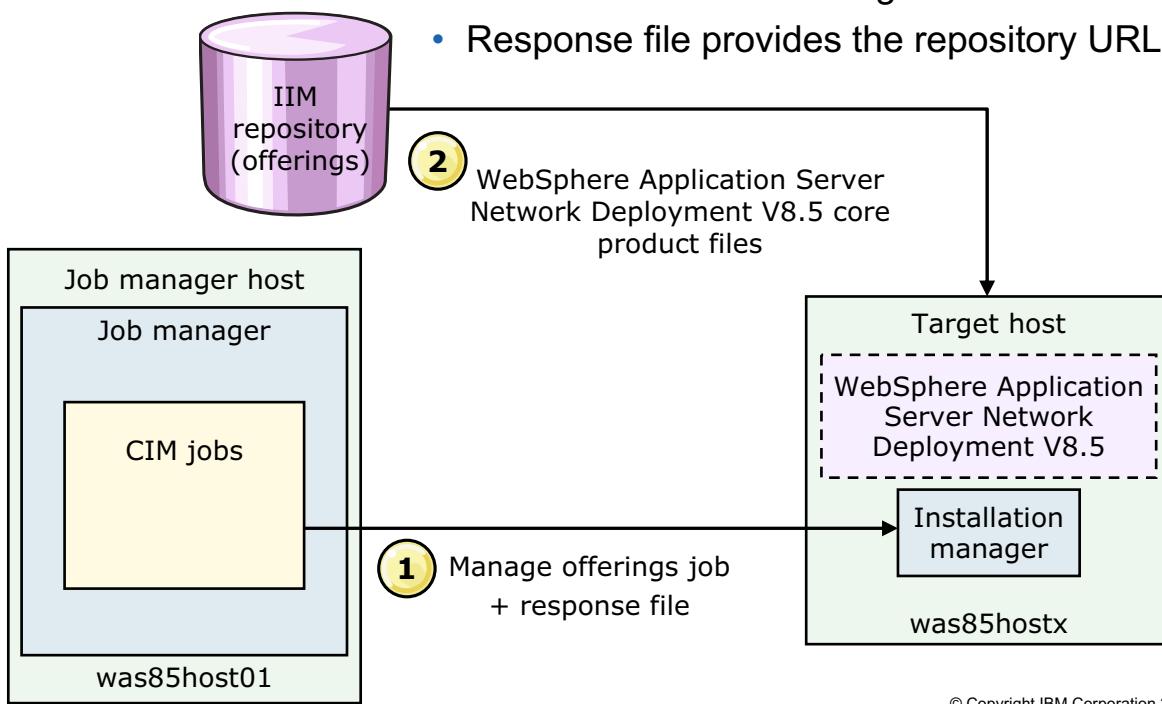
Notes:

If you have multiple Installation Manager offerings or must manage Installation Manager on multiple remote hosts, the job manager can automate this process. Job manager can also store your Installation Manager installation kits in a single repository. This capability means that you can manage your installation kits from a single location and send your installation kits to multiple machines.

You can also submit jobs to update and uninstall the Installation Manager. The job types are `updateIM` and `uninstallIM`.

Submitting a Manage offerings job

- was85hostx is a registered target
- IIM is installed on the target
- Response file provides the repository URL



© Copyright IBM Corporation 2013

Figure 17-18. Submitting a Manage offerings job

WA680 / VA6801.0

Notes:

When a target host is registered with a job manager and the Installation Manager is also installed on the target, a manage offerings job can be submitted to the job manager. A response file must be provided during the job configuration that specifies what product is going to be installed and how to access the binary files for that product from the IIM repository.

Install WebSphere on a remote host

- The `manageOfferings` administrative job can install, update, modify, rollback, or uninstall offerings in Installation Manager.
- For example:

```
AdminTask.submitJob('-jobType manageOfferings -targetList [was85host1 was85host2]
                     -jobParams [responseFile /usr/JobMgr/responsefiles/WAS85_response_file.xml]
                     -username user1 -password pw')
```

- Job parameters
 - **responseFile**: Specifies the location of the response file to install, update, modify, rollback, or uninstall Installation Manager offerings.
 - **IMPath**: Specifies the location of the Installation Manager installation.

© Copyright IBM Corporation 2013

Figure 17-19. Install WebSphere on a remote host

WA680 / VA6801.0

Notes:

responseFile: Specifies the location of the response file to install, update, modify, rollback, or uninstall Installation Manager offerings. The path can be for a local file on the job manager computer, or a URL that points to a remote file.

IMPath: Specifies the location of the Installation Manager installation. If this parameter is not specified, the job tries to discover and use the path of the default Installation Manager on the target host.

Creating a profile

- Use the **manageprofiles** job of the job manager to create, augment, or delete a WebSphere Application Server profile.
- This job runs the **manageprofiles** administrative command.
- For example:

```
AdminTask.submitJob('-jobType manageprofiles -targetList was85host1  
-jobParams [[wasHome /opt/WAS85] [responseFile /scripts/ASprofile.txt]]  
-username user1 -password pw')
```

- Job parameters
 - **wasHome**: Specifies the location of the WebSphere Application Server installation. This location is the *app_server_root* path on the remote host.
 - **responseFile**: Specifies the location of the **manageprofiles** response file that is stored on the job manager.

© Copyright IBM Corporation 2013

Figure 17-20. Creating a profile

WA680 / VA6801.0

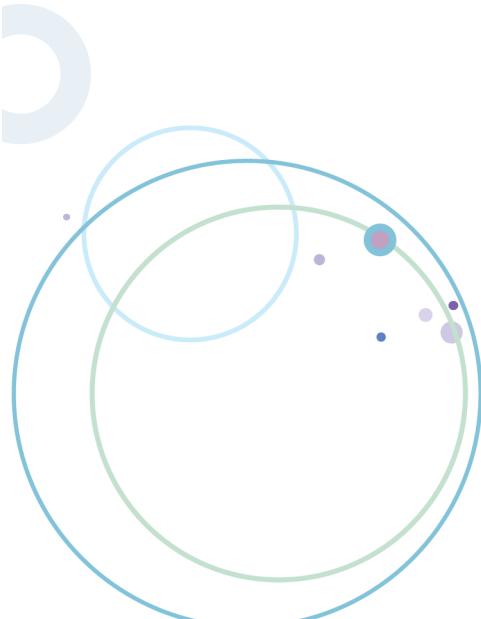
Notes:

username: Specifies the user name to use at the targets when the job is run. If the user name and password are not specified, and security is enabled, the user ID at the job manager is used.

password: Specifies the password for the user name to use at the targets when the job is run. If the user name and password are not specified, and security is enabled, the user ID at the job manager is used.

17.3.Creating a flexible management environment

Creating a flexible management environment



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 17-21. Creating a flexible management environment

WA680 / VA6801.0

Notes:

Registering deployment manager with job manager

- Use the wsadmin `registerWithJobManager` command to register deployment managers.
- The command is in the ManagedNodeAgent command group.
- Run this Jython command from the deployment manager's wsadmin
- For example:

```
AdminTask.registerWithJobManager('[-host jobmgr_host -port secure_console_port
-user wasadmin -password admin_pw -managedNodeName deployment_manager_node_name]')
```

- Where:
 - `jobmgr_host`: is the host name of the job manager
 - `Secure_console_port`: specifies the job manager administrative console secure port number
 - `user`: is the WebSphere administrative user for the job manager
 - `password`: is the WebSphere administrative user job manager
 - `deployment_manager_node_name`: is the node name of the deployment manager

© Copyright IBM Corporation 2013

Figure 17-22. Registering deployment manager with job manager

WA680 / VA6801.0

Notes:

To run the Jython command that is shown on this slide, both the job manager and the deployment manager must be started, and wsadmin must be connected with the deployment manager.

jobmgr_host is the host name of the job manager. The default value is localhost.

console_port specifies the deployment manager administrative console port number or the job manager administrative console port number. The value that you specify for *console_port* depends upon whether you want to run jobs on deployment manager nodes from the job manager function available in a deployment manager or from a separate job manager.

deployment_manager_node_name is the node name of the deployment manager. The node name typically is dmgrNode.

If the command is successful, wsadmin displays the unique ID (UUID) of the job manager. For example: 'JobMgr01-JOB_MANAGER-74cdda0c-68f6-4970-a959-6f6800b9f22d'

Flexible management jobs

- Manage servers
 - Create and Delete Application Server
 - Start and Stop Server
- Manage applications
 - Download application EAR files
 - Install and uninstall Application
 - Start and Stop Application
 - Update Application
- File distribution
 - Collect File
 - Distribute File
 - Remove File
- Run wsadmin script
- Configure properties
- Inventory
- Status

© Copyright IBM Corporation 2013

Figure 17-23. Flexible management jobs

WA680 / VA6801.0

Notes:

Examples of jobs in flexible management are installing an application, creating an application server, running a wsadmin script on a remote node, and many others.

Asynchronous nature of jobs

- When jobs are submitted, they are not necessarily run right away
- The administrative agents and deployment managers pull jobs from the job manager when they are online, based on configured polling intervals
- Because jobs can also be created with intervals of allowable execution, if the job is not retrieved during a validity period, it is not run
- The job status summary page presents four types of job completion status
 - Succeeded
 - Partially succeeded
 - Failed
 - Incomplete



© Copyright IBM Corporation 2013

Figure 17-24. Asynchronous nature of jobs

WA680 / VA6801.0

Notes:

In the administrative console, status is displayed in a chart that you can view to see which nodes are completed under each state. The chart indicates status by color and position in the summary bar.

When you submit a job, it starts out with incomplete status. When the job is retrieved, the detailed status shows it as distributed, then in progress. The normal progression of jobs at that point is for their status to go to successful, partially successful, or failed. It takes at least two polling cycles to retrieve a job and then return results, and it can take more cycles, according to how long the job actually takes to process on the node. The polling cycle time is configurable at the administrative agent or the deployment manager. You might find that the status summary shows incomplete nodes, but when you drill down, no nodes show in the list. When you refresh the status summary, you find that the node either completed or failed in that window of time and is no longer incomplete.

Managing the server run time

- Use the application server runtime control jobs to start and stop application servers.
 - startServer
 - stopServer
 - startCluster: Can use ripplestart option
 - stopCluster
- Examples:

```
AdminTask.submitJob('-jobType startServer -targetList [was85host]  
-jobParams [nodeName profile1 serverName server1]')
```

```
AdminTask.submitJob('[-jobType startCluster -targetList [dmgrNode]  
-jobParams [ [clusterName PlantsCluster] [rippleStart true] ]]')
```

- To start or stop a cluster in a cell, you must register the deployment manager of the cell with a job manager.
- To start cluster members, the node agents must be running on each node

© Copyright IBM Corporation 2013

Figure 17-25. Managing the server run time

WA680 / VA6801.0

Notes:

To run the Jython commands that are shown on this slide, the job manager must be started, and wsadmin must be connected with the job manager.

To start a cluster in a cell, you must register the deployment manager of the cell with a job manager. The deployment manager can be registered with its own job manager or with a stand-alone job manager.

Notice that the target list value is the name of the deployment manager's node (dmgrNode in this example) when you are submitting a job on a cluster.

Scheduling future and recurring jobs (1 of 2)

- Use the following job parameters to schedule future and recurring administrative jobs in a flexible management environment
 - **activationDateTime**: Date and time to activate the job
 - **expirationDateTime**: Expiration date for the job (default is 24 hours)
 - **expireAfter**: amount of time, in minutes, to wait before the job expires
 - **executionWindow**: Specifies the recurring interval for the job.
 - **executionWindowUnit**: Recurring interval unit for value that the executionWindow parameter sets (DAILY, WEEKLY, MONTHLY)
 - **email**: Address to which the system sends job notifications

© Copyright IBM Corporation 2013

Figure 17-26. Scheduling future and recurring jobs (1 of 2)

WA680 / VA6801.0

Notes:

activationDateTime: Specifies the date and time to activate the job in the format "2006-05-03T10:30:45-0000". The -0000 section of the activationDateTime parameter value represents RFC 822 format. You can specify Z as a shortcut for Greenwich Mean time (GMT), such as "2006-05-03T10:30:45Z". If you do not specify the time zone, the system uses the time zone of the server.

expirationDateTime: Specifies the expiration date for the job, in the format "2006-05-03T10:30:45-0000". The -0000 section of the activationDateTime parameter value represents RFC 822 format. You can specify Z as a shortcut for Greenwich Mean time (GMT), such as "2006-05-03T10:30:45Z". If you do not specify the time zone, the system uses the time zone of the server.

expireAfter: Specifies the amount of time, in minutes, to wait before the job expires.

executionWindow: Specifies the recurring interval for the job.

executionWindowUnit: Specifies the recurring interval unit of measure for the value that is set by the executionWindow parameter. Specify DAILY to run the job daily, WEEKLY to run the job weekly, MONTHLY to run the job monthly, YEARLY to run the job annually. Specify CONNECTION to run the job each time the node connects and polls for jobs.

email: Specifies the email address to which the system sends job notifications.

Scheduling future and recurring jobs (2 of 2)

- **Scenario:** An administrator must restart a cluster once a day during off-peak hours for two weeks. The administrator might submit the following jobs.
- Schedule the stopping of the cluster:

```
AdminTask.submitJob('[-jobType stopCluster -targetList [dmgrNode ]
-jobParams [ [clusterName PlantsCluster] ]
-activationDateTime 2013-03-19T22:45:00 -expirationDateTime 2013-04-03T01:00:00
-executionWindowUnit DAILY -executionWindow 22:45:00-22:55:00]
-email Joe_admin@company.com')
```

- Schedule the ripple-starting of the cluster:

```
AdminTask.submitJob('[-jobType startCluster -targetList [dmgrNode ]
-jobParams [ [rippleStart TRUE] [clusterName PlantsCluster] ]
-activationDateTime 2013-03-19T23:00:00 -expirationDateTime 2013-04-03T01:00:00
-executionWindowUnit DAILY -executionWindow 23:00:00-23:20:00]
-email Joe_admin@company.com')
```

© Copyright IBM Corporation 2013

Figure 17-27. Scheduling future and recurring jobs (2 of 2)

WA680 / VA6801.0

Notes:

The commands that are shown on this slide are for illustrative purposes. Before using similar commands, administrators can run tests in their environment, and adjust the time intervals according to the stopping and starting behavior of their cluster.

Administrative agent

- Profile type to support flexible management
- Register base application server profiles with the administrative agent
- Register base application server profiles with the job manager through the administrative agent
- Administrative agent polls job manager for jobs on behalf of application servers

© Copyright IBM Corporation 2013

Figure 17-28. Administrative agent

WA680 / VA6801.0

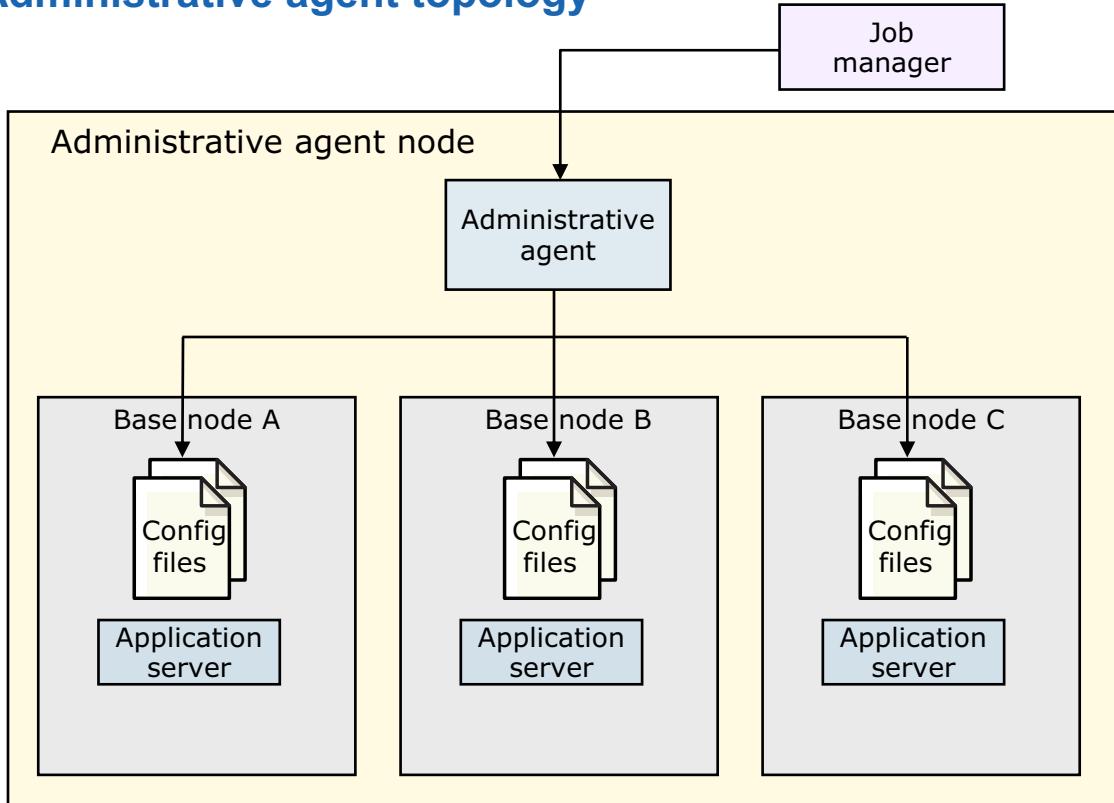
Notes:

An administrative agent provides a single interface to administer multiple unfederated application server nodes in environments such as development, unit test, or that portion of a server farm that is on a single machine.

The administrative agent and application servers must be on the same machine, but you can connect to the machine from a browser or the wsadmin tool on another machine.

You can register an application server node with the administrative agent or federate the node with a deployment manager, but not both.

Administrative agent topology



© Copyright IBM Corporation 2013

Figure 17-29. Administrative agent topology

WA680 / VA6801.0

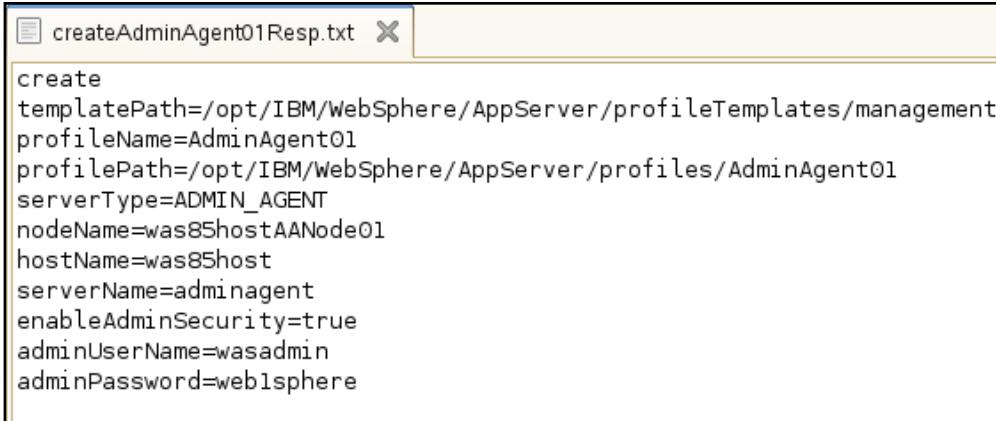
Notes:

The administrative agent is designed as an option to complement the WebSphere Application Server base topology, in which the stand-alone application server continues to serve the application requests. Only administrative services from the server are consolidated into the administrative agent. For every WebSphere Application Server base profile registered with the administrative agent, an administrative subsystem is created within the administrative agent to represent the new administrative entry point for that profile.

The administrative agent polls job manager for jobs on behalf of application servers that run on its node.

Creating the administrative agent profile

- Use manageprofiles command to create the administrative agent profile
- First, create a response file
 - The profile template is **management**
 - The server type is **ADMIN_AGENT**



```
create
templatePath=/opt/IBM/WebSphere/AppServer/profileTemplates/management
profileName=AdminAgent01
profilePath=/opt/IBM/WebSphere/AppServer/profiles/AdminAgent01
serverType=ADMIN_AGENT
nodeName=was85hostAANode01
hostName=was85host
serverName=adminagent
enableAdminSecurity=true
adminUserName=wasadmin
adminPassword=websphere
```

- Run the command:
`./manageprofiles.sh -response /createAdminAgent01Resp.txt`

© Copyright IBM Corporation 2013

Figure 17-30. Creating the administrative agent profile

WA680 / VA6801.0

Notes:

The manageprofiles command is run from the <was_root>/bin directory.

The management profile template is used with the -serverType parameter to indicate the type of management profile.

When the command completes successfully, you see a message as follows:

“INSTCONFSUCCESS: Success: Profile AdminAgent01 now exists. Please consult <profile_root>/AdminAgent01/logs/AboutThisProfile.txt for more information about this profile.”

Start the job manager by using the command:

<profile_root>/AdminAgent01/bin/startServer.sh adminagent

The AboutThisProfile.txt file shows the the ports that are assigned to the jobmgr process. In particular, the following ports are important to take note of. The port values shown here are the defaults; the ports on your system might be different.

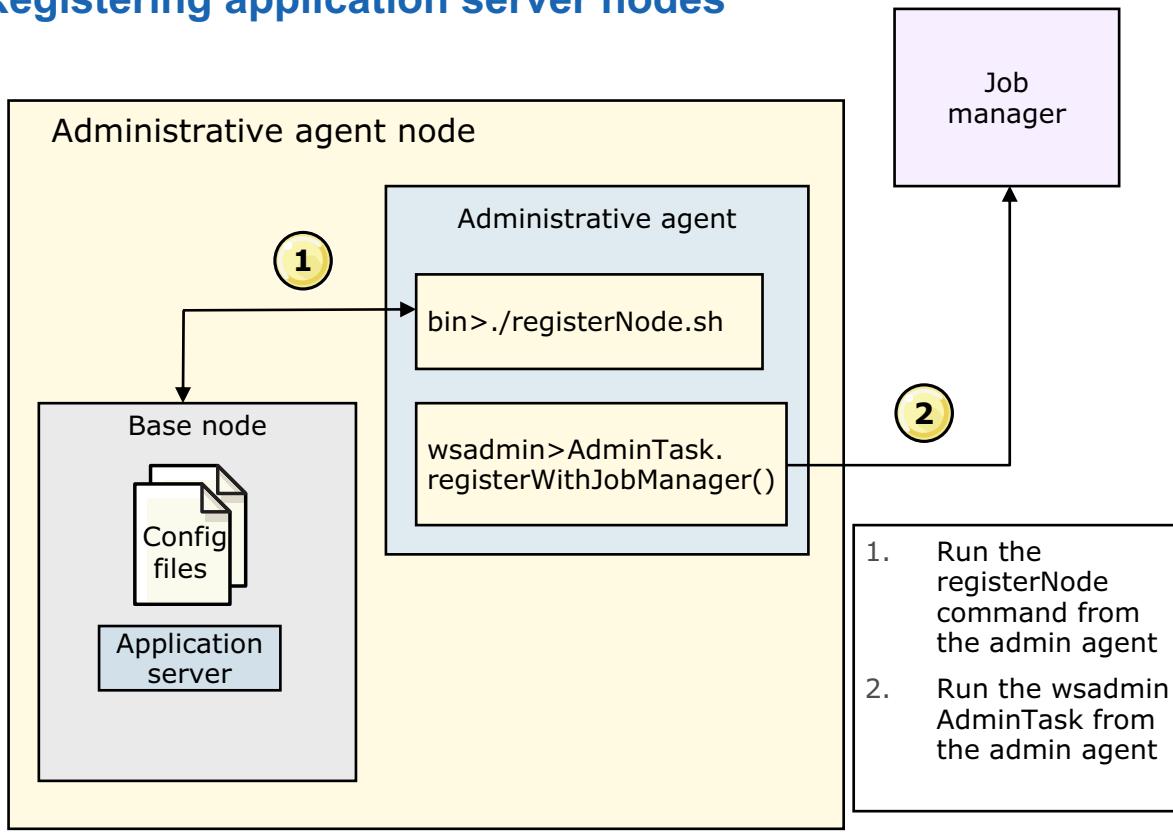
Administrative console port: 9964

Administrative console secure port: 9047

Management SOAP connector port: 8877

For example, to access the administrative agent's administrative console, use the web address, <http://localhost:9964/ibm/console>

Registering application server nodes



© Copyright IBM Corporation 2013

Figure 17-31. Registering application server nodes

WA680 / VA6801.0

Notes:

You must configure a flexible management environment, consisting of a job manager, and an administrative agent for the application server node to register. Start the job manager and the administrative agent processes before registering nodes with the job manager.

Register application server profiles with the administrative agent

- Use the `registerNode` command utility to register the application server profile with the administrative agent
- Run this command from the `bin` directory of the administrative agent server to register a profile (node) with the administrative agent
- For example:

```
./registerNode.sh -profilePath <profile_root>/profile1 -host was85host
-conntype SOAP -port 8877
```

- The administrative agent and the node that is being registered must be on the same system.
 - Verify the SOAP port of the administrative agent by using the `AboutThisProfile.txt` in the `<profile_root>/admin_agent_name/logs` directory
- You can run this command only on an unfederated node.

© Copyright IBM Corporation 2013

Figure 17-32. Register application server profiles with the administrative agent

WA680 / VA6801.0

Notes:

Run the `registerNode` command from the `bin` directory of the administrative agent server to register a node with the administrative agent. Both the administrative agent server and the profile's application server must be running. If security is enabled, you are prompted to authenticate.

When you run the command, the stand-alone node is converted into a node that the administrative agent manages. The administrative agent and the node being registered must be on the same system. You can run the command only on an unfederated node. If the command is run on a federated node, the command exits with an error.

Register application server profiles with the job manager through the administrative agent

- Use the `registerWithJobManager` command to register a profile as a node on the job manager.
- Start wsadmin from the `bin` directory of the administrative agent
- The following example command registers the `was85hostNode01` application server profile with the job manager

```
AdminTask.registerWithJobManager(['-host localhost -port secure_console_port
-user wasadmin -password admin_pw -managedNodeName was85hostNode01'])
```

- Parameters
 - `host`: Specifies the host name of the job manager.
 - `port`: secured administrative console port of the job manager
 - `user`: is the WebSphere administrative user for the job manager
 - `password`: is the WebSphere administrative user job manager
 - `managedNodeName`: Specifies the name of the node that is registered with the administrative agent.

© Copyright IBM Corporation 2013

Figure 17-33. Register application server profiles with the job manager through the administrative agent

WA680 / VA6801.0

Notes:

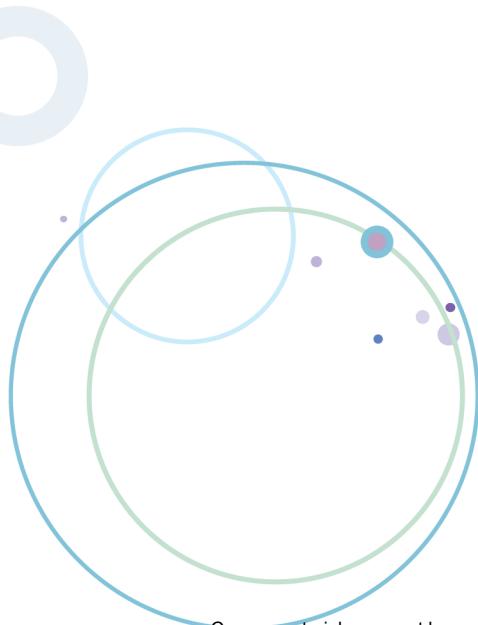
Some optional parameters include:

-startPolling: Optionally specifies whether to start polling after registering the node. Specify false to disable polling. The default value is true.

-autoAcceptSigner: Optionally specifies whether to automatically accept the signer that is provided by the server. Specify false to disable this option. The default value is true.

17.4. Automating the installation of WebSphere Application Server

Automating the installation of WebSphere Application Server



© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 17-34. Automating the installation of WebSphere Application Server

WA680 / VA6801.0

Notes:

Silent installation of IBM Installation Manager (IIM)

- IBM Installation Manager Kit
 - Provides a directory that contains an instance of IBM Installation Manager that can install a package and the IBM Installation Manager.
 - The file that starts Installation Manager is named `install`.
 - You can configure the kit to install both Installation Manager and one or more other packages.
- IBM Installation Manager (IIM)
 - This item is the installed version of IBM Installation Manager.
 - IIM is installed by default into the `/opt/IBM/Installation Manager` directory
 - The name of the file that starts Installation Manager is `IBMMIM`

© Copyright IBM Corporation 2013

Figure 17-35. Silent installation of IBM Installation Manager (IIM)

WA680 / VA6801.0

Notes:

IBM Installation Manager is the Eclipse-based tool to manage the installation, update, modification, rollback, and uninstallation of product packages.

It includes a number of wizards that make it easy to maintain packages throughout their lifecycles:

- Installation wizard: walks you through the installation
- Update wizard: searches for updates to packages installed
- Modify wizard: modify certain elements of a package
- Roll Back wizard: allows you to revert to a previous version of a package
- Uninstall wizard: removes a package
- Manage Licenses wizard: allows you to manage and configure various licenses

The Installation manager can be installed interactively or silently.

Silent installation of IBM Installation Manager (IIM)

- You must use the IBM Installation Manager Kit to install IBM Installation Manager on a client computer.
- Download the IBM Installation Manager Kit, agent.installer.platform.zip, for a specific operating system.
 - For example, a recent version for Linux is named agent.installer.linux.gtk.x86_1.5.3000.20120531.zip.
- Extract the compressed file to a directory.
- Install by using the Installation Manager silent installation command
 - For example: ./installc -acceptLicense -log log_file_path_and_name
- Add a repository and download product packages to it.
 - Refer to the WebSphere Application Server V8.5 information center for details about setting up a repository

© Copyright IBM Corporation 2013

Figure 17-36. Silent installation of IBM Installation Manager (IIM)

WA680 / VA6801.0

Notes:

A **repository** is where the installable packages are found. The repository includes metadata that describes the software version and how it is installed. It has a list of files that are organized in a tree structure. The repository can be local or on a remote server. A **package** is a software product that Installation Manager installs. It is a separately installable unit that can operate independently from other packages of that software.

Create a response file for installing WebSphere core product

- Create a response file that is based on samples in the WebSphere Application Server Information Center
- Or use the IBM Installation Manager in record mode to create a response file
 - Navigate to the directory /opt/IBM/InstallationManager/eclipse
- Enter the following command:


```
./IBMIM -skipInstall /usr/IBM-repositories/WAS85 -record
/var/temp/WAS85_response_file.xml
```
- The IBM Installation Manager starts and guides you through installation steps
- These steps record your selections into an XML response file

© Copyright IBM Corporation 2013

Figure 17-37. Create a response file for installing WebSphere core product

WA680 / VA6801.0

Notes:

Several sample response files are available in the WebSphere Application Server Information Center. See the topic: **Using the sample response files**.

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=%2Fcom.ibm.websphere.installation.nd.doc%2Fae%2Ftins_WASv85_sample_response.html

In particular look at the topic: **Sample response file: Installing IBM WebSphere Application Server Network Deployment**

Example: Response file for WebSphere core product (1 of 2)

- Top section of the response file must
 - Point to location of a local or remote repository
 - Specify profile ID
 - Specify installation directory

```
<?xml version="1.0" encoding="UTF-8"?>
<!--The "acceptLicense" attribute has been deprecated.-->
<agent-input acceptLicense='true'>
<server>
<repository location='/usr/IBM-repositories/WAS85' />
</server>
<profile id='IBM WebSphere Application Server V8.5_2'
  installLocation='/opt/IBM/WebSphere/AppServer_2'>
<data key='eclipseLocation' value='/opt/IBM/WebSphere/AppServer_2' />
<data key='user.import.profile' value='false' />
<data key='cic.selector.os' value='linux' />
<data key='cic.selector.ws' value='gtk' />
<data key='cic.selector.arch' value='x86' />
<data key='cic.selector.nl' value='en' />
</profile>
```

© Copyright IBM Corporation 2013

Figure 17-38. Example: Response file for WebSphere core product (1 of 2)

WA680 / VA6801.0

Notes:

The acceptLicense attribute is deprecated. Use the -acceptLicense command-line option to accept license agreements.

Repositories are locations that Installation Manager queries for installable packages. Repositories can be local (on the machine with Installation Manager) or remote (on a corporate intranet or hosted elsewhere on the internet). If the machine that is using this response file has access to the Internet, then include the IBM WebSphere Live Update Repositories in the list of repository locations. If the machine that is using this response file cannot access the internet, then specify the URL or UNC path to custom intranet repositories and directory paths to local repositories to use.

The profile attribute is required and typically is unique to the offering. If modifying or updating an existing installation, the profile attribute must match the profile ID of the targeted installation of WebSphere Application Server.

The installLocation specifies where the offering is installed. If the response file is used to modify or update an existing installation, then ensure the installLocation points to the location where the offering was installed previously.

Example: Response file for WebSphere core product (2 of 2)

- Installation features
 - The `features` line specifies product components to install
 - This line can be modified to update an existing installation

```
<install modify='false'>
<offering id='com.ibm.websphere.ND.v85' version='8.5.0.20120501_1108'
profile='IBM WebSphere Application Server V8.5_2'
features='core.feature,ejbdeploy,thinclient,embeddablecontainer,
com.ibm.sdk.6_32bit' installFixes='none' />
</install>
```

- Some of the features in this example include:
 - `core.feature` Indicates the full WebSphere Application Server profile
 - `thinclient` Indicates the stand-alone thin clients and resource adapters

© Copyright IBM Corporation 2013

Figure 17-39. Example: Response file for WebSphere core product (2 of 2)

WA680 / VA6801.0

Notes:

The `features` attribute is optional. Offerings always have at least one feature; a required core feature that is installed regardless of whether it is explicitly specified. If other feature names are provided, then only those features are installed. Features must be comma delimited without spaces.

You must install **core.feature** (full WebSphere Application Server profile), **liberty** (Liberty profile), or both.

The `features` list contains the names of features that are installed by default.

- **core.feature** indicates the full WebSphere Application Server profile
- **ejbdeploy** indicates the EJBDeploy tool for pre-EJB 3.0 modules
- **thinclient** indicates the standalone thin clients and resource adapters
- **embeddablecontainer** indicates the embeddable EJB container
- **com.ibm.sdk.6_32bit** allows you to choose a 32-bit Software Development Kit if you are installing on a 64-bit system

- **samples** indicates the sample applications feature (This feature is not listed in the response file because it is not installed by default.)

Silent installation of WebSphere core product files

- Run Installation Manager command line (imcl) installation command.
- Navigate to /opt/IBM/InstallationManager/eclipse/tools
- For example, enter the following command all on one line:


```
./imcl -acceptLicense -input
<response_file_location>/WAS85_response_file.xml -log
<log_location>/WAS_install_log.xml
```

```
File Edit View Terminal Tabs Help
was85host:/opt/IBM/InstallationManager/eclipse/tools # ./imcl -acceptLicense -in
put /usr/Software/Scripts/Exercisell/WAS85_response_file.xml -log /usr/Software/
Scripts/Exercisell/WAS_install_log.xml
Installed com.ibm.websphere.ND.v85_8.5.0.20120501_1108 to the /opt/IBM/WebSphere
/AppServer_2 directory.
```

© Copyright IBM Corporation 2013

Figure 17-40. Silent installation of WebSphere core product files

WA680 / VA6801.0

Notes:

Run the following command from the /eclipse/tools subdirectory in the directory where you installed Installation Manager:

Windows: imcl.exe -acceptLicense -showProgress -input
 <response_file_path_and_name> -log <log_file_path_and_name>

Linux, UNIX, IBM i and z/OS: ./imcl -acceptLicense -showProgress -input
 <response_file_path_and_name> -log <log_file_path_and_name>

Create a profile with response file and the manageprofiles command

- The response file for creating an application server profile with administrative security requires at least the following arguments and values:
 - **create**
 - **templatePath** <was_root>/profileTemplates/managed
 - **profileName** profile_name
 - **profilePath** <profile_root>/profile_name
 - **enableAdminSecurity** true
 - **adminUserName** admin_name
 - **adminPassword** admin_password
- Run the manageprofiles command
 - Navigate to <was_root>/bin
 - Enter the following command:
 ./manageprofiles.sh -response
 <response_file_location>/profile_response_file.txt

© Copyright IBM Corporation 2013

Figure 17-41. Create a profile with response file and the manageprofiles command

WA680 / VA6801.0

Notes:

-create

create a profile

-templatePath *template_path*

Specifies the directory path to the template files in the installation root directory. Within the profileTemplates directory are various directories that correspond to different profile types and that vary with the type of product installed. The profile directories are the paths that you indicate while using the -templatePath option.

-profileName *profile_name*

Specifies the name of the profile. Use a unique value when creating a profile. Each profile that shares the same set of product binaries must have a unique name.

-profilePath *profile_root*

Specifies the fully qualified path to the profile, which is referred to as the *profile_root*.

-enableAdminSecurity true | false

Enables administrative security. Valid values include true or false. The default value is false.

When enableAdminSecurity is set to true, you must also specify the parameters -adminUserName and -adminPassword along with the values for these parameters.

Unit summary

Having completed this unit, you should be able to:

- Describe the flexible management topology
- Describe the function of the job manager and the administrative agent
- Use scripts to create job manager and administrative agent profiles
- Use scripts to configure target hosts for the job manager
- Use scripts to configure CIM jobs for the job manager
- Use scripts to submit a job to install the Installation Manager on a remote host
- Use scripts to install the WebSphere Application Server and create an application server profile
- Use scripts to monitor the status of a job manager job
- Automate the installation of WebSphere Application Server

© Copyright IBM Corporation 2013

Figure 17-42. Unit summary

WA680 / VA6801.0

Notes:

Checkpoint questions

1. (True or False) You can use the manageprofiles command to create the job manager profile.
2. (True or False) The installIM administrative job installs a new instance of Installation Manager.
3. (True or False) The deployment manager can be registered with a job manager by using only the administrative console.

© Copyright IBM Corporation 2013

Figure 17-43. Checkpoint questions

WA680 / VA6801.0

Notes:

Write your answers here:

1.

2.

3.



Checkpoint answers

1. True
2. True
3. False: You can also use the `registerWithJobManager` command to register deployment managers.

© Copyright IBM Corporation 2013

Figure 17-44. Checkpoint answers

WA680 / VA6801.0

Notes:

Exercise 11



Automating the installation of
WebSphere Application Server

© Copyright IBM Corporation 2013

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

7.0

Figure 17-45. Exercise 11

WA680 / VA6801.0

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Describe the silent installation of IBM Installation Manager
- Use the IBM Installation Manager to record a response file
- Use a recorded response file to silently install the WebSphere Application Server Network Deployment product files
- Edit the response file to modify the installation
- Customize a response file to create a WebSphere profile by using the manageprofiles command

© Copyright IBM Corporation 2013

Figure 17-46. Exercise objectives

WA680 / VA6801.0

Notes:

Unit 18. Course summary

What this unit is about

This unit provides a brief summary of the course.

What you should be able to do

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study



Unit objectives

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

Figure 18-1. Unit objectives

WA680 / VA6801.0

Notes:

Course learning objectives

Having completed this course. You are able to:

- Describe the WebSphere Application Server support for scripting and automation
- Use Jython and the IBM Assembly and Deploy Tools (IADT) to develop automated scripts
- Identify the administrative objects and programming APIs needed for administrative scripting
- Use the wsadmin tool to prototype and run scripts
- Write scripts to automate common WebSphere Application Server administration tasks
- Describe the use of Ant and ws_ant to automate tasks
- Use scripting to submit job manager jobs
- Explain the new scripting and automation capabilities in WebSphere Application Server V8.5

© Copyright IBM Corporation 2013

Figure 18-2. Course learning objectives

WA680 / VA6801.0

Notes:



Class evaluation

- Your comments about this class are useful to WebSphere Education
- Feedback on the site, curriculum, and instructor tells WebSphere Education what was good about the class and what can be improved
- Please take the time to complete the course evaluation on the IBM Training website:
<http://www.ibm.com/training/osart>

© Copyright IBM Corporation 2013

Figure 18-3. Class evaluation

WA680 / VA6801.0

Notes:



To learn more on the subject

- WebSphere Education website:
www.ibm.com/websphere/education
- Training paths:
www.ibm.com/software/websphere/education/paths/
 - Identify the next courses in this sequence
- Resource Guide
 - Contains information about many useful sources of information
 - Many of these sources are free
 - See handout in your class materials, or download a copy
 - www.ibm.com/developerworks/wikis/display/WEinstructors/WebSphere+Resource+Guide
- Course catalog:
<http://bit.ly/wecatalog>
- Email address for more information:
websphere_skills@us.ibm.com

© Copyright IBM Corporation 2013

Figure 18-4. To learn more on the subject

WA680 / VA6801.0

Notes:

References

Books

- *WebSphere Application Server Administration Using Jython*, Robert A. Gibson, Arthur Kevin McGrath, Noel J. Bergman, IBM Press, 2010
- *WebSphere Application Server V8.5 Administration and Configuration Guide*, IBM Redbooks, SG24-8056-00

Web resources

- IBM developerWorks paper: *Command assistance simplifies administrative scripting in WebSphere Application Server*
http://www.ibm.com/developerworks/websphere/library/techarticles/0812_rhodes/0812_rhodes.html
- IBM developerWorks paper: *Properties-based configuration*
http://www.ibm.com/developerworks/websphere/techjournal/0904_changing/0904_chang.html
- Sample Scripts for WebSphere Application Server:
<http://www.ibm.com/developerworks/websphere/library/samples/SampleScripts.html>

© Copyright IBM Corporation 2013

Figure 18-5. References

WA680 / VA6801.0

Notes:

Unit summary

Having completed this unit, you should be able to:

- Explain how the course met its learning objectives
- Submit an evaluation of the class
- Identify other WebSphere Education courses that are related to this topic
- Access the WebSphere Education website
- Locate appropriate resources for further study

© Copyright IBM Corporation 2013

Figure 18-6. Unit summary

WA680 / VA6801.0

Notes:

Appendix 19. List of abbreviations and acronyms

AB	ApacheBench
AFS	Andrew File System
AIX	Advanced IBM UNIX
AMI	asynchronous message interface
Ant	Another Neat Tool
APAR	Authorized Program Analysis Report
API	application programming interface
ARM	Application Response Measurement
ASCII	American Standard Code for Information Interchange
AST	Application Server Toolkit
BSF	Bean Scripting Framework
CA	certificate authority
CIM	Centralized Installation Management
CIP	custom installation package
CMP	container-managed persistence
CMS	Certificate Management System
CMT	Configuration Migration Tool
CP	caching proxy
CPU	central processing unit
CSlv2	Common Secure Interoperability Protocol Version 2
DB	database
DC	domain controller
DCS	Distribution and Consistency Services
DD	deployment descriptor
DHCP	Dynamic Host Configuration Protocol
DLL	Dynamic Link Library
DMgr or dmgr	deployment manager
DMZ	demilitarized zone
DN	distinguished name
DNS	Domain Name System

DRS	data replication service
DTD	document type definition
EAR	enterprise archive
EE	Enterprise Edition
EIS	enterprise information system
EJB	Enterprise JavaBean
EJS	Enterprise Java Services
EL	Expression Language
ENC	Enterprise Naming Context
ESB	Enterprise service bus
ESI	Edge Side Include
FIPS	Federal Information Processing Standard
FQDN	fully qualified domain name
FTP	File transfer protocol
GA	generally available
GC	garbage collection
GCD	greatest common divisor
GCMV	Garbage Collection and Memory Visualizer
GIF	Graphics Interchange Format
GMT	Greenwich mean time
GTK	GNU GUI Tool Kit
GSS	Generic Security Services
GUI	graphical user interface
HA	high availability or highly available
HACMP	High Availability Cluster Multi-Processing
HAM	high availability manager
HFS	Hierarchical File System
HPEL	High Performance Extensible Logging
HPUX	Hewlett Packard UNIX
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPD	HTTP Daemon

HTTPS	HTTP over SSL
IADT	IBM Assembly and Deploy Tools
IIM	IBM Installation Manager
IOP	Internet Inter-ORB Protocol
IMS	Information Management System
I/O	input/output
IP	Internet Protocol
ISC	Integrated Solutions Console
IPSEC	IP Security
ISMP	InstallShield MultiPlatform
IT	information technology
ITCAM	IBM Tivoli Composite Application Manager
ITDS	IBM Tivoli Directory Server
IVT	Installation Verification Tool
J2C	Java 2 Connector
J2EE	Java 2 Enterprise Edition
J2SE	Java 2 Platform Standard Edition
JAAS	Java Authentication and Authorization Service
JACC	Java Authorization Contract for Containers
JAF	Java Activation Framework
JAR	Java archive
JASPIC	Java Authentication Service Provider Interface for Containers
Java EE	Java Platform, Enterprise Edition
JAXB	Java Architecture for XML Binding
JAXP	Java API for XML Processing
JAXR	Java API for XML Registries
JAX-RPC	Java API for XML Remote Procedure Calls
JAX-RS	Java API for XML-based Remote Procedure Calls
JAX-WS	Java API for XML Web Services
JCA	Java EE Connector Architecture
JCE	Java Cryptology Extension
JDBC	Java Database Connectivity

JDE	JD Edwards
JDK	Java Development Kit
JIT	just-in-time
JMS	Java Message Service
JMX	Java Management Extensions
JNDI	Java Naming and Directory Interface
JPA	Java Persistence API
JPG	Graphics file type or extension (lossy compressed 24-bit color image storage format developed by the Joint Photographic Experts Group)
JRE	Java Runtime Environment
JSF	JavaServer Faces
JSP	JavaServer Pages
JSR	Java Specification Request
JSTL	JavaScript Tag Library
JTA	Java Transaction API
JVM	Java virtual machine
LAN	Local area network
LB	Load Balancer
LDAP	Lightweight Directory Access Protocol
LSD	Location service daemon
LTPA	Lightweight Third Party Authentication
MAC	message authentication code or media access control
MDB	message-driven bean
ME	messaging engine
MQ	Message Queue
MQI	Message Queue Interface
MVS	Multiple Virtual System
NAS	network attached storage
NA	network address translation
NIC	network interface card
NIM	Network Installment Management

NTP	Network Time Protocol
OLTP	online transaction processing
ORB	Object Request Broker
OS	operating system
OVF	Open Virtualization Format
PAR	Parchive Index File
PCT	Plug-ins Configuration Tool
PD	problem determination
PFBC	properties file based configuration
PGP	Pretty Good Privacy
PHP	personal home page
PID	process identifier
PKI	public key infrastructure
PME	programming model extensions
PM	Performance Monitoring Infrastructure
PMR	problem management record
PMT	Program Management Tool
POJO	plain old Java object
PWB	PlantsByWebSphere
QoS	quality of service
RA	registration authority
RAM	random access memory
RAR	resource archive
RACF	Resource Access Control Facility
RDBMS	relational database management system
RDN	relative distinguished name
REST	Representational State Transfer
RM	request metrics
RMI	Remote Method Invocation
RMI/IOP	Remote Method Invocation over Internet InterORB Protocol
RUP	Rational Unified Process
RXA	Remote Execution and Access

SAAJ	SOAP with Attachments API for Java
SAM	Security Access Manager
SAR	SIP application resource
SAS	Secure Association Service
SBDT	Smart Business Development and Test
SCA	Service Component Architecture
SCADA	supervisory control and data acquisition
SCM	source code management
SDO	Service Data Objects
SDK	software development kit
SDLC	systems development lifecycle
SFSB	stateful session bean
SIB or SIBus	service integration bus
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SOA	service-oriented architecture
SOAP	A lightweight, XML-based protocol for exchanging information in a decentralized, distributed environment. Usage note: SOAP is not an acronym; it is a word in itself (formerly an acronym for Simple Object Access Protocol)
SPI	service provider interface
SPNEGO	Simple and Protected GSS-API Negotiation Mechanism
SQL	Structured Query Language
SSB	stateful session bean
SSL	Secure Sockets Layer
SSO	single sign-on
StAX	Streaming API for XML
SUSE	Software und System Entwicklung (German: Software and Systems Development)
SVC	supervisor call
SVG	Scalable Vector Graphics
SWAM	Simple WebSphere Authentication Mechanism
TAM	Tivoli Access Manager

TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TKCARS1	Toolkit for Custom and Reusable Solution Information
TLS	Transport Layer Security
TP	Trade Processor (application in lab exercises)
UDDI	Universal Description, Discovery, and Integration
UNC	Universal Naming Convention
UNIX	Uniplexed Information and Computing System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Universal Test Client
VMM	virtual member manager
VPN	virtual private network
WAR	web archive
WCT	WebSphere Customization Toolbox
WLM	workload management
WPM	WebSphere Platform Messaging
WPS	WebSphere Process Server
WS	web services
WS-AT	web services atomic transaction
WS-BA	web services business activity
WS-COOR	web services coordination
WSDL	Web Services Description Language
WS-I	Web Services Interoperability
WSIF	Web Services Invocation Framework
XA	Extended Architecture
XML	Extensible Markup Language
XTP	extreme transaction processing
z/OS	zSeries operating system
zMMT	z/OS Migration Management Tool
zPMT	z/OS Profile Management Tool

IBM
®