

# HP Operations Orchestration Software

Software Version: 7.60

## *Oracle WebLogic Server Integration Guide*

Document Release Date: January 2010

Software Release Date: January 2010



## Legal Notices

### Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

### Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

### Copyright Notices

© Copyright 2009-2010 Hewlett-Packard Development Company, L.P.

### Trademark Notices

For information on open-source and third-party software acknowledgements, see in the documentation set for this release, Open-Source and Third-Party Software Acknowledgements (3rdPartyOpenNotices.pdf).

# On the Web: Finding OO support and documentation

There are two Web sites where you can find support and documentation, including updates to OO Help systems, guides, and tutorials:

- The OO Support site
- BSA Essentials Network

## Support

Documentation enhancements are a continual project at Hewlett-Packard Software. You can obtain or update the HP OO documentation set and tutorials at any time from the HP Software Product Manuals Web site. You will need an HP Passport to log in to the Web site.

### To obtain HP OO documentation and tutorials

1. Go to the HP Software Product Manuals Web site (<http://support.openview.hp.com/selfsolve/manuals>).
2. Log in with your HP Passport user name and password.

OR

If you do not have an HP Passport, click **New users – please register** to create an HP Passport, then return to this page and log in.

If you need help getting an HP Passport, see your HP OO contact.

3. In the **Product** list box, scroll down to and select **Operations Orchestration**.
4. In the **Product Version** list, click the version of the manuals that you're interested in.
5. In the **Operating System** list, click the relevant operating system.
6. Click the **Search** button.
7. In the **Results** list, click the link for the file that you want.

## BSA Essentials Network

For support information, including patches, troubleshooting aids, support contract management, product manuals and more, visit the following site: <http://www.hp.com/go/bsaessentialsnetwork>

This is the **BSA Essentials Network** Web page. To sign in:

1. Click **Login Now**.
2. On the **HP Passport sign-in** page, enter your HP Passport user ID and password and then click **Sign-in**.
3. If you do not already have an HP Passport account, do the following:
  - a. On the **HP Passport sign-in** page, click **New user registration**.
  - b. On the **HP Passport new user registration** page, enter the required information and then click **Continue**.
  - c. On the confirmation page that opens, check your information and then click **Register**.
  - d. On the **Terms of Service** page, read the Terms of use and legal restrictions, select the **Agree** button, and then click **Submit**.

4. On the **BSA Essentials Network** page, click **Operations Orchestration Community**.  
**The Operations Orchestration Community** page contains links to announcements, discussions, downloads, documentation, help, and support.

**Note:** Contact your OO contact if you have any difficulties with this process.

## In OO: How to find Help, PDFs, and tutorials

The HP Operations Orchestration software (HP OO) documentation set is made up of the following:

- **Help for Central**  
Central Help provides information to the following:
  - Finding and running flows
  - For HP OO administrators, configuring the functioning of HP OO
  - Generating and viewing the information available from the outcomes of flow runsThe Central Help system is also available as a PDF document in the HP OO home directory, in the \Central\docs subdirectory.
- **Help for Studio**  
Studio Help instructs flow authors at varying levels of programming ability.  
The Studio Help system is also available as a PDF document in the HP OO home directory, in the \Studio\docs subdirectory.
- **Animated tutorials for Central and Studio**  
HP OO tutorials can each be completed in less than half an hour and provide basic instruction on the following:
  - In Central, finding, running, and viewing information from flows
  - In Studio, modifying flowsThe tutorials are available in the Central and Studio subdirectories of the HP OO home directory.
- **Self-documentation for operations and flows in the Accelerator Packs and ITIL folders**  
Self-documentation is available in the descriptions of the operations and steps that are included in the flows.

# Table of Contents

Warranty .....	ii
Restricted Rights Legend .....	ii
Trademark Notices .....	ii
On the Web: Finding OO support and documentation.....	iii
Support.....	iii
BSA Essentials Network.....	iii
In OO: How to find Help, PDFs, and tutorials.....	iv
Table of Contents.....	v
Overview of the Oracle WebLogic Server integration.....	1
Use cases and scenarios .....	1
Installation and configuration instructions.....	2
Versions and components .....	3
Architecture .....	3
WebLogic integration operation and flow infrastructure.....	3
Common inputs in the integration .....	5
Operation and flow specifics .....	5
Check Application State operation .....	5
Configure Application Deployment Attribute operation.....	5
Configure Server Attribute operation.....	6
Deploy Application operation.....	6
Get Application Names By Server flow.....	7

Is Application Up flow .....	7
Is WebLogic Up flow .....	7
Query Application Deployment Configuration operation .....	7
Query Application Deployment Configuration Target operation .....	7
Query Application Runtime operation.....	9
Query Configuration Manager flow.....	9
Query Domain Configuration flow.....	9
Query Domain Runtime flow .....	10
Query Server Configuration operation .....	10
Query Server Runtime operation .....	10
Restart Application flow .....	11
Restart Server flow .....	11
Restart Server flow .....	12
Resume Server operation .....	12
Start Application operation .....	12
Start Server operation .....	13
Stop Application operation.....	13
Suspend Server operation .....	13
Total Server Shutdown operation .....	13
Undeploy Application operation .....	14
Wait For State flow .....	14
<b>Troubleshooting.....</b>	<b>14</b>
General troubleshooting procedures and tools .....	14
Error messages .....	16
<b>Security .....</b>	<b>17</b>
<b>Tools.....</b>	<b>18</b>
<b>Appendices .....</b>	<b>19</b>
Query Application Deployment Configuration input list.....	19
Query Application Deployment Configuration Target input list.....	21
Query Server Configuration input list.....	21
Query Server Runtime input list .....	37
Query Domain Configuration result list.....	38



# Overview of the Oracle WebLogic Server integration

With the Oracle WebLogic Server (WebLogic) integration, administrators can build HP Operations Orchestration (OO) flows that are integrated with WebLogic.

This document explains how this integration has been implemented and how the integration OO operations communicate between OO and WebLogic.

## Use cases and scenarios

This section defines the major use cases for the WebLogic integration, and lists the operations and flows that you can use to implement them.

1. Query the WebLogic domain:
  - Query Configuration Manager
  - Query Domain Configuration
  - Query Domain Runtime
2. Manage and query application servers:
  - Configure Server Attribute
  - Query Server Configuration
  - Query Server Runtime
  - Restart Server
  - Resume Server
  - Start Server
  - Suspend Server
  - Total Server Shutdown
3. Manage and query applications:
  - Configure Application Deployment Attribute
  - Deploy Application
  - Get Application Names By Server
  - Query Application Deployment Configuration
  - Query Application Deployment Configuration Target
  - Query Application Runtime
  - Restart Application
  - Start Application
  - Stop Application
  - Undeploy Application
4. Check the health of WebLogic and applications:
  - Check Application State
  - Is Application Up
  - Is WebLogic Up
  - Wait For State



# Installation and configuration instructions

To connect to the WebLogic server, you must set up and configure the HP OO RJRAS service.

## To set up and configure the HP OO RSJRAS service to connect to the WebLogic server

1. Make sure that the server on which the OO RSJRAS service is running can access the WebLogic server.

The default WebLogic access URL is:

```
http://<weblogicServer>:7001
```

The default username is **weblogic** and the default password is **weblogic**.

2. Shut down the RSJRAS server.
3. Create a BEA folder in the %OO\_home%/RAS/Java/Default/repository/lib/ folder. Copy the WebLogic jar files shown below from the WebLogic server system to %OO\_home%/RAS/Java/Default/repository/lib/BEA/ folder.

For WebLogic 10.3 integration, copy the following .jar files:

- %WebLogic\_Install\_Directory%/wlserver\_10.3/server/lib/weblogic.jar
- %WebLogic\_Install\_Directory%/wlserver\_10.3/server/lib/wljmsclient.jar
- %WebLogic\_Install\_Directory%/wlserver\_10.3/server/lib/wljmxclient.jar
- %WebLogic\_Install\_Directory%/wlserver\_10.3/server/lib/wlclient.jar
- All of the com.bea.core.\*jar files in %WebLogic\_Install\_Directory%/wlserver\_10.3/server/lib/wseeclient.zip
- %WebLogic\_Install\_Directory%/modules/com.bea.core.utils.wrapper\_1.3.0.0.jar
- %WebLogic\_Install\_Directory%/modules/com.bea.core.weblogic.socket.api\_1.0.0.0.jar

For WebLogic 9.2 integration, copy the following .jar files:

- %WebLogic\_Install\_Directory%/weblogic92/server/lib/weblogic.jar
- %WebLogic\_Install\_Directory%/weblogic92/server/lib/wljmxclient.jar
- %WebLogic\_Install\_Directory%/weblogic92/server/lib/wlclient.jar

4. To modify the Wrapper.conf file, open %OO\_home%/RAS/Java/Default/webapp/conf/wrapper.conf file and add the following lines:  
#will needed to be added for the jar to tell the JRAS where to find the WebLogic license file

```
wrapper.java.additional.<number>=-Dbea.home=<mapped bea location>
```

```
#must have the following lineto turn on a backwards compatibility flag to allow the operations to work:
```

```
wrapper.java.additional.<number>=-Dsun.lang.ClassLoader.allowArraySyntax=true
```

**Note:** The <number> parameter should continue from the last

wrapper.java.additional.<number> additional property. For example, if

wrapper.java.additional.4 is the last additional property, use .5 and .6 for the WebLogic additional properties. For <mapped bea location>, map the driver to the WebLogic server.

Here are two examples:

```
wrapper.java.additional.5= Dbea.home=w:\bea
```

```
wrapper.java.additional.6=-Dsun.lang.ClassLoader.allowArraySyntax=true
```

5. For HTTP protocol support, make sure that the protocol for the HTTP configuration of the application has **Enable Tunneling** and **Send Server Header** checked.

6. Restart the RSJRAS server.

## Versions and components

Operations Orchestration Version	WebLogic Version
7.60	9.2, 10.3

## Architecture

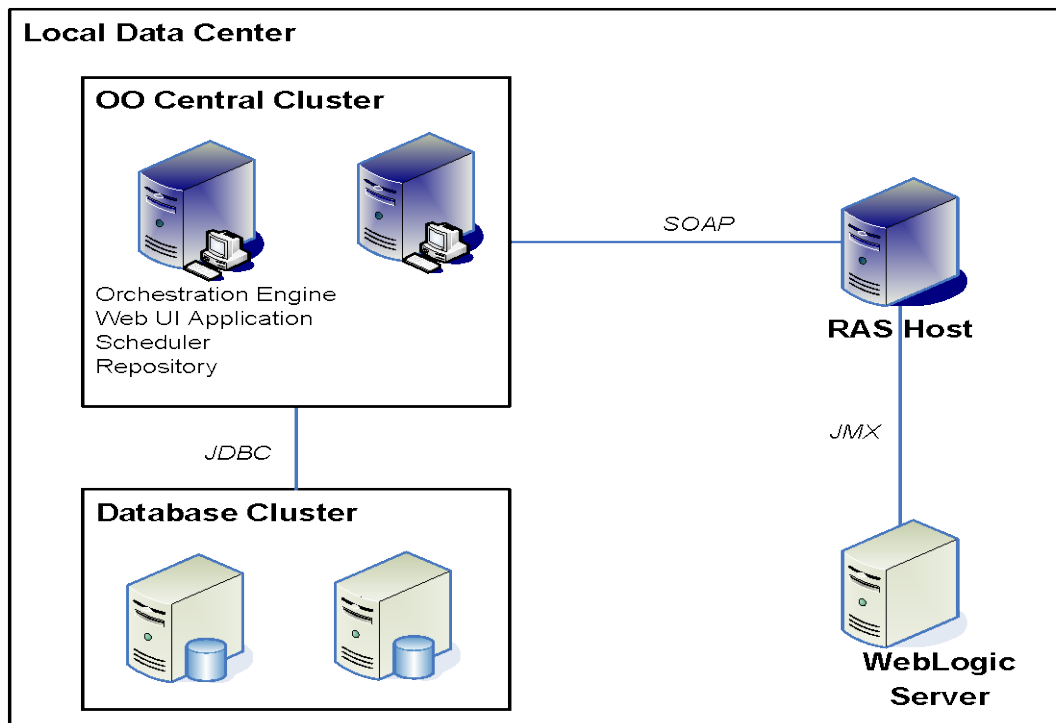


Figure 1 - WebLogic Server architecture

## WebLogic integration operation and flow infrastructure

The WebLogic integration includes the operations and flows in the following folders:

- Library/Operations/Application Servers/BEA WebLogic/
- Library/Accelerator Packs/Application Servers/BEA WebLogic/



**Figure 2 - WebLogic operations and flows in the Studio Library**

The WebLogic integration flows and operations support the following types of tasks:

- Retrieving configuration information about the configuration manager, domain, servers, and applications in the WebLogic.
- Configuring application servers or applications in the WebLogic.
- Verifying application states in the WebLogic.
- Performing start, stop, suspend, and resume operations on servers in the WebLogic.
- Performing start, stop, restart, deploy, and undeploy operations on applications in the WebLogic.

## Common inputs in the integration

OO flows and operations use inputs to specify how they obtain the data that they need and when the data is obtained. The following inputs are used consistently throughout the WebLogic integration's operations and flows.

### host

The DNS name or IP address host of the WebLogic server.

### port

The number of the port on which the WebLogic server is listening for connections. The default port is **7001**.

### protocol

The protocol to connect to the WebLogic server. The default protocol is **T3**. It also supports HTTP.

### username

The username to connect to the WebLogic server. The default username is **weblogic**.

### password

The password for the username to connect to the WebLogic server. The default password is **weblogic**.

## Operation and flow specifics

This section describes the WebLogic Server integration operations and flows, including any operation- or flow-specific inputs. Inputs that are common to all of the WebLogic Server integration operations and flows are described in [Common inputs in the integration](#).

### Check Application State operation

The **Check Application State** operation checks an application's state in the WebLogic. The possible application states are **STATE\_ACTIVE**, **STATE\_NEW**, **STATE\_PREPARED**, and **STATE\_ADMIN**.

#### target

The name of the server on which the application to check resides.

#### application

The name of the application to check.

#### module

The name of the module to check in the specified application.

#### subModule

The name of the submodule to check in the specified application.

### Configure Application Deployment Attribute operation

The **Configure Application Deployment Attribute** operation sets an attribute of an application deployment MBean to a new value in the WebLogic.

**name**

The name of the application.

**attribute**

The attribute name for the application deployment. You can find all of the attribute names by using the [Query Application Deployment Configuration](#) operation.

**value**

The new value to set for the attribute.

## Configure Server Attribute operation

The **Configure Server Attribute** operation configures an attribute for the specified application server on the WebLogic Server. You can find the server attributes using the [Query Server Configuration](#) operation.

**name**

The name of the application server whose attribute you want to configure.

**attribute**

The attribute whose value you want to set.

**value**

The new value of the attribute.

## Deploy Application operation

The **Deploy Application** operation deploys an application on a target or targets in the WebLogic. If an application with this name already exists, the flow deploys that application. Otherwise, this is a new deployment and an application MBean is created and fully configured based on the application descriptors found in the archive or directory specified in the **source** input.

**source**

An absolute path to the application to deploy on the WebLogic server. You must specify the **source** input if this operation is deploying a new application. The path has to be on the system where the WebLogic server is running. For example: `c:\temp\logging-helloworld.ear`

**name**

The name of the application to deploy. If an application with this name already exists, the flow deploys that application. Otherwise, the operation creates a new application.

**target1**

The application server or cluster to target. You can add more application servers to target by creating additional target inputs using sequential numbers (for example: **target2**, **target3**, **target4**). Once a null target is found, targets with higher numbers are not checked.

**id**

An ID for the deployment task. If you do not specify an ID, the operation automatically generates one.

**stagingMode**

An option that you can use to override the staging attribute of the target servers. The valid options are **stage**, **nostage**, and **external\_stage**.

## Get Application Names By Server flow

The **Get Application Names By Server** flow retrieves all of the application names in the WebLogic for the specified server. The flow returns a comma-delimited list of application names.

## Is Application Up flow

The **Is Application Up** flow allows the user to check the state of an application on a WebLogic server.

## Is WebLogic Up flow

The **Is WebLogic Up** flow checks whether the WebLogic is running.

### server

The name of the application server from which the application names are to be retrieved.

## Query Application Deployment Configuration operation

The **Query Application Deployment Configuration** operation returns the deployment configuration information for the specified application in the WebLogic. The following is an example of the returned attribute name/value pair result:

```
{ApplicationName=SamplesSearchWebApp;Type=AppDeployment;CompatibilityName=anything;InternalApp=false;StagingMode=;ModuleType=war;AbsolutePlanDir=;AbsolutePlanPath=;LocalSourcePath=C:\bea\wlserver_10.3\samples\server\docs;AutoDeployedApp=false;ApplicationIdentifier=SamplesSearchWebApp;VersionIdentifier=;Result=0;returnCode=0;BackgroundDeployment=false;LocalPlanDir=;LocalInstallDir=;PlanDir=;ValidateDDSecurityData=false;PlanPath=;InstallDir=;SourcePath=C:\bea\wlserver_10.3\samples\server\docs;Notes=test;Name=SamplesSearchWebApp;AbsolutePath=C:\bea\wlserver_10.3\samples\server\docs;LocalPlanPath=;sessionId=iconclude-8334314772115004177;SecurityDDModel=Advanced;AbsoluteInstallDir=;RootStagingDir=C:\bea\wlserver_10.3\samples\domains\wl_server\servers\examplesServer\stage\SamplesSearchWebApp;DeploymentOrder=100;}
```

There are many attributes, such as the following, that you can specify to limit the query for this operation.

### ApplicationName

The name of the application for which the deployment configuration information is to be retrieved. Note that the name of the current MBean is not the name of the application.

### Name

A unique identifier for this bean instance.

## Query Application Deployment Configuration Target operation

The **Query Application Deployment Configuration Target** operation returns the target server configuration information for the specified application in the WebLogic. The following is an example of the returned attribute name/value pair result:

```
{ConsensusProcessIdentifier=-1;NativeIOEnabled=true;CustomTrustKeyStorePassPhraseEncrypted=;SystemPasswordEncryp
```

```

ted=;JavaStandardTrustKeyStorePassPhrase=;AdminReconnectIntervalSeconds=55;CustomId
entityKeyStorePassPhrase=;ListenPort=7001;HealthCheckStartDelaySeconds=120;DefaultI
IOPasswordEncrypted=[B@e0128f;ServerLifecycleTimeoutVal=55;ConsoleInputEnabled=fal
se;TimedOutRefIsolationTime=0;CustomTrustKeyStorePassPhrase=;ListenDelaySecs=0;LowM
emoryTimeInterval=3600;Result=0;PreferredSecondaryGroup=;IgnoreSessionsDuringShutdo
wn=false;JavaCompiler=javac;RestartDelaySeconds=0;CompleteMessageTimeout=44;CustomI
dentityKeyStoreFileName=;DefaultSecureProtocol=t3s;LogRemoteExceptionsEnabled=false
;JavaStandardTrustKeyStorePassPhraseEncrypted=;JMSDefaultConnectionFactoriesEnabled
=true;HttpTraceSupportEnabled=false;AutoKillIfFailed=false;InstrumentStackTraceEnab
led=true;Use81StyleExecuteQueues=false;TransactionLogFileWritePolicy=Direct-
Write;PeriodLength=60000;IdleIIOPConnectionTimeout=-
1;StagingMode=nostage;MaxCOMMessageSize=-1;CompleteIIOPMessageTimeout=-
1;ExternalDNSName=;TunnelingClientPingSecs=45;UploadDirectoryName=.\\servers\\example
sServer\\upload;ReverseDNSAllowed=false;SelfTuningThreadPoolSizeMin=1;CompleteHTTPE
ssageTimeout=-
1;StagingDirectoryName=C:\\bea\\wlserver_10.3\\samples\\domains\\wl_server\\servers\\examp
lesServer\\stage;SelfTuningThreadPoolSizeMax=400;HostsMigratableServices=true;IdlePe
riodsUntilTimeout=4;MessagingBridgeThreadPoolSize=5;HttpdEnabled=true;TGIOPEntered=
true;CustomTrustKeyStoreFileName=;InterfaceAddress=;DGCIdlePeriodsUntilTimeout=5;Cu
stomIdentityKeyStorePassPhraseEncrypted=;SocketBufferSizeAsChunkSize=false;LoginTim
eoutMillis=5000;ClusterWeight=100;ManagedServerIndependenceEnabled=true;AutoMigrati
onEnabled=false;DefaultIIOPPassword=weblogic;ServerVersion=unknown;IdleConnectionTi
meout=65;HealthCheckIntervalSeconds=180;GracefulShutdownTimeout=0;LowMemorySampleSi
ze=10;DefaultProtocol=t3;ListenPortEnabled=true;ExpectedToRun=true;LowMemoryGranula
rityLevel=5;IIOPTxMechanism=ots;DefaultInternalServletsDisabled=false;TunnelingEnab
led=true;JavaCompilerPostClassPath=;StartupMode=RUNNING;RestartMax=2;MaxIIOPMessage
Size=-
1;WeblogicPluginEnabled=false;ListenAddress=;TransactionLogFilePrefix=./;CustomTrus
tKeyStoreType=;MSIFileReplicationEnabled=false;MaxHTTPMessageSize=-
1;Notes=test;MaxT3MessageSize=-
1;sessionId=iconclude4140800852459846361;CustomIdentityKeyStoreType=;JNDITransporta
bleObjectFactoryList=[Ljava.lang.String;@e28774;DefaultTGIOPPUser=guest;Administrati
onPortEnabled=false;AutoRestart=true;JMSThreadPoolSize=15;MaxOpenSockCount=-
1;ListenersBindEarly=false;MuxerClass=;LowMemoryGCThreshold=5;ClasspathServletDisab
led=false;Type=Server;JDBCCLRTableName=;DefaultIIOPUser=weblogic;ThreadPoolPercentS
ocketReaders=33;AdministrationPort=9002;StuckThreadTimerInterval=60;CompleteT3Messa
geTimeout=-1;SocketReaders=-
1;ReplicationGroup=;RestartIntervalSeconds=3600;ExtraRmicOptions=;HealthCheckTimeou
tSeconds=60;TunnelingClientTimeoutSecs=40;DefaultTGIOPPassword=;DefaultTGIOPPasswor
dEncrypted=;JDBCLoggingEnabled=false;StartupTimeout=0;VerboseEJBDeploymentEnabled=f
alse;AdministrationProtocol=t3s;StuckThreadMaxTime=600;returnCode=0;JavaCompilerPre
ClassPath=;CompleteCOMMessageTimeout=-
1;MessageIdPrefixEnabled=false;JDBCLoginTimeoutSeconds=0;AcceptBacklog=300;ListenTh
readStartDelaySecs=65;ExtraEjbcOptions=;Name=examplesServer;KeyStores=DemoIdentityA
ndDemoTrust;OutboundPrivateKeyEnabled=false;ClientCertProxyEnabled=false;MaxMessage
Size=10000000;OutboundEnabled=false;COMEnabled=false;IIOPEnabled=true;}

```

There are many attributes, such as the following, that you can specify to limit the query for this operation. A complete list of these attributes is contained in [Query Server Configuration input list](#) in the Appendices.

### deployment

The name of the application.

### Name

The name of the application server.

## Query Application Runtime operation

The **Query Application Runtime** operation retrieves detailed runtime information for WebLogic applications running on a specified application server. If you do not specify the **name** input or **applicationName** input, the runtime information for all of the applications is returned. The following is the example of the returned attribute name/value pair result:

```
{Result=0;returnCode=0;isEar=false;sessionId=iconclude4140800852459846363;name=SamplesSearchWebApp;applicationName=SamplesSearchWebApp;activeVersionState=2;type=ApplicationRuntime;version=;}
```

### **server**

The application server name.

### **applicationName**

The name of the application.

### **activeVersionState**

Specifies whether this application version is the currently active version. For example, 2 is for Active state, 1 is for Admin state.

### **isEar**

Checks if application is an EAR file. Returns **True** if it is.

### **name**

The name of the application.

### **type**

The type of the MBean. For example, **ApplicationRuntime**.

### **version**

The application version.

## Query Configuration Manager flow

The **Query Configuration Manager** flow retrieves detailed configuration information about the global configuration manager in the WebLogic.

The following is the example of the returned attribute name/value pairs:

```
result:{Result=0;returnCode=0;currentEditorExclusive=false;parent=;currentEditorExpirationTime=0;type=weblogic.management.mbeanservers.edit.ConfigurationManagerMBean;completedActivationTasksCount=10;editor=false;sessionId=iconclude3390855419080158433;currentEditor=;name=ConfigurationManager;path=/ConfigurationManager[ConfigurationManager];currentEditorStartTime=0;}
```

## Query Domain Configuration flow

The **Query Domain Configuration** flow retrieves configuration information about the state of a domain in the WebLogic.

The following is the example of the returned attribute name/value pairs:

```
{DomainVersion=10.3.0.0;ClusterConstraintsEnabled=false;Type=Domain;ArchiveConfigurationCount=0;AdministrationPort=9002;AdministrationMBeanAuditingEnabled=false;LastModificationTime=0;ConfigBackupEnabled=false;ConsoleExtensionDirectory=console-ext;AdministrationProtocol=t3s;returnCode=0;Result=0;ConsoleContextPath=console;Notes=;Name=wl_server;ConfigurationVersion=10.3.0.0;AdminServerName=examplesServer;ConsoleEnabled=true;sessionId=iconclude3390855419080158434;ProductionModeEnabled=false}
```



```
;AdministrationPortEnabled=false;AutoDeployForSubmodulesEnabled=true;ConfigurationAuditType=none;RootDirectory=C:\bea\wlserver_10.3\samples\domains\wl_server;}
```

To see the complete list of return results for this operation, see the [Query Domain Configuration result list](#) appendix.

## Query Domain Runtime flow

The **Query Domain Runtime** flow retrieves detailed runtime information for a WebLogic domain.

The following is the example of the returned attribute name/value pairs:

```
{ActivationTime=Thu Jul 09 11:02:34 PDT
2009;Name=wl_server;Result=0;returnCode=0;sessionId=iconclude3390855419080158435;Type=DomainRuntime;}Troubleshooting
```

## Query Server Configuration operation

The **Query Server Configuration** operation retrieves detailed configuration information about an application server in the WebLogic domain.

The operation returns the target server configuration information for the application in the WebLogic. You can see sample returned values in the section [Query Application Deployment Configuration Target](#) flow.

There are many attribute inputs that you can use to limit the query for this operation. The following input is the most important one. To see the entire list of attribute inputs for the **Query Server Configuration** operation, see the [Query Server Configuration input list](#) appendix.

### Name

The name of the application server.

## Query Server Runtime operation

The **Query Server Runtime** operation retrieves runtime information for application server or servers in the WebLogic. If you specify a **name** input value, all of the server's runtime information is returned. The following is an example of the returned attribute name/value pair result:

```
{adminPortEnabled=false;defaultURL=t3://15.23.143.55:7001;adminServerPortSecure=false;adminServerPort=7001;listenAddress=ros-zhizhou.rose.hp.com/15.23.143.55;adminURL=t3://15.23.143.55:7001;state=RUNNING;totalOpenSockets=7;sslPort=7002;adminServer=true;type=ServerRuntime;version=WebLogic Server 10.3 Fri Jul 25 16:30:05 EDT 2008 1137967 ;startupTime=Wed Dec 31 16:00:36 PST 1969;restartRequired=false;adminServerHost=15.23.143.55;activationTime=Thu Jul 09 11:02:44 PDT
2009;name=examplesServer;adminPort=9002;Result=RUNNING;returnCode=0;currentDirectory=C:\bea\wlserver_10.3\samples\domains\wl_server\.;clusterMaster=false;sslAddress=ros-zhizhou.rose.hp.com/15.23.143.55;classpath=C:\bea\JROCKI~1\jre\bin\jrockit\jrockit1.6.0.jar;C:\bea\JROCKI~1\jre\bin\jrockit\jmapi.jar;C:\bea\JROCKI~1\jre\bin\jrockit\jmxmapi.jar;C:\bea\JROCKI~1\jre\bin\jrockit\rmp.jar;C:\bea\JROCKI~1\jre\bin\jrockit\latency.jar;C:\bea\JROCKI~1\jre\lib\resources.jar;C:\bea\JROCKI~1\jre\lib\rt.jar;C:\bea\JROCKI~1\jre\lib\sunrsasign.jar;C:\bea\JROCKI~1\jre\lib\jsse.jar;C:\bea\JROCKI~1\jre\lib\jce.jar;C:\bea\JROCKI~1\jre\lib\charsets.jar;C:\bea\JROCKI~1\jre\classes;C:\bea\wlserver_10.3\samples\server\examples\build\serverclasses;C:\bea\patch_wlw1030\profiles\default\sys_manifest_classpath\weblogic_patch.jar;C:\bea\patch_wls103
```

```
0\profiles\default\sys_manifest_classpath\weblogic_patch.jar;C:\bea\patch_cie660\pr
ofiles\default\sys_manifest_classpath\weblogic_patch.jar;C:\bea\JROCKI~1\lib\tools.
jar;C:\bea\WLSERV~1.3\server\lib\weblogic_sp.jar;C:\bea\WLSERV~1.3\server\lib\weblo
gic.jar;C:\bea\modules\features\weblogic.server.modules_10.3.0.0.jar;C:\bea\WLSERV~
1.3\server\lib\webservices.jar;C:\bea\modules\ORGAPA~1.5\lib\ant-
all.jar;C:\bea\modules\NETSFA~1.0_1\lib\ant-
contrib.jar;;C:\bea\WLSERV~1.3\common\eval\pointbase\lib\pbembedded57.jar;C:\bea\WL
SERV~1.3\common\eval\pointbase\lib\pbclient57.jar;C:\bea\WLSERV~1.3\server\lib\xqrl
.jar;;restartCount=0;sessionId=iconclude6763408203992038713;shuttingDown=false;cur
rentMachine=;currentOpenSockets=7;sslPortEnabled=true;}
```

There are many attributes that you can specify to limit the query for this operation. The following is the most common attribute used to narrow down the query. To see the complete list of the attributes, see the [Query Server Runtime input list](#) appendix.

**name**

The name of the application server.

## Restart Application flow

The **Restart Application** flow restarts an application on a WebLogic server. It gets the application name to start, stops the application, waits for it to stop, restarts the application, and then waits for it to restart. If you do not specify the **application** input, the flow finds all the application names for the specified application server and presents them in a selection list from which you select one of the available application names.

**server**

The application server name.

**application**

The application name. If you do not specify an application name, the flow finds all of the application names for the application server.

**delay**

The number of seconds to wait between retries.

**retry**

The number of times to check the application state before timing out.

## Restart Server flow

The **Restart Server** flow in the Library/Operations/Application Servers/BEA WebLogic/Servers/ folder restarts an application server and waits for to start completely in the WebLogic. This flow requires the use of a node manager. It cannot restart the application server that is the admin server of the WebLogic.

**Warning:** This flow completely disables your WebLogic server if you restart the application server that is the admin server for the WebLogic.

**name**

The application server name.

**delay**

The number of seconds to wait between checking the server. The default value is **10**.

**retry**

The number of times to check the server state before timing out. The default value is **30**.

**ignoreSessions**

Specifies whether or not the flow should ignore pending HTTP requests while finishing processing requests that have been started. The valid values are **True** and **False**.

**timeout**

The number of seconds to wait before the flow forces a restart of the server.

## Restart Server flow

The **Restart Server** flow in the Library/Accelerator Packs/Application Servers/BEA WebLogic/Utility/ folder finds the running application server, allows the user to select a server to restart, and waits for the server to completely start in the WebLogic. It requires the use of a node manager. The flow cannot restart an application server that is also the admin server of the WebLogic.

**delay**

The number of seconds to wait between retries. It defaults to 10 seconds.

**retry**

The number of times to check the server state before timing out. It defaults to 30 times.

**ignoreSessions**

Allows you to specify if a session needs to be ignored. The valid values are **true** and **false**. Whether or not to ignore pending http requests while finishing processing requests that have been started.

**timeout**

The number of seconds to wait before forcefully starting the application server.

## Resume Server operation

The **Resume Server** operation resumes an application server from the ADMIN or suspended state in the WebLogic. This operation requires that you still be able to connect to the domain in the WebLogic server.

**name**

The name of the application server to be resumed.

## Start Application operation

The **Start Application** operation starts a deployed application on the specified target WebLogic server or servers.

**name**

The name of the application to start.

**target1**

The application server to target. You can add more application servers to target by creating additional target inputs using sequential numbers (for example: **target2**, **target3**, **target4**). Once a null target is found, targets with higher numbers are not checked.

**id**

An ID for the deployment task. If you do not specify an ID, one is automatically generated.

## Start Server operation

The **Start Server** operation starts an application server and waits for it to start completely. If the server is the only server in the WebLogic, it requires the use of a node manager.

### **name**

The name of the application server to be started.

### **ignoreSessions**

Specifies whether or not the operation should ignore pending HTTP requests while finishing processing requests that have been started. The valid values are **True** and **False**.

### **timeout**

The number of seconds to wait before the operation forces the application server to start.

## Stop Application operation

The **Stop Application** operation stops a running application on the WebLogic server.

### **name**

The name of the application to stop.

### **target1**

The application server to target. You can add more application servers to target by creating additional target inputs using sequential numbers (for example: **target2**, **target3**, **target4**). Once a null target is found, targets with higher numbers are not checked.

### **id**

An ID for the deployment task. If you do not specify an ID, the operation automatically generates one.

## Suspend Server operation

The **Suspend Server** operation suspends an application server and waits for it to be completely suspended.

### **name**

The name of the application server to be suspended.

### **ignoreSessions**

Specifies whether or not the operation should ignore pending HTTP requests while finishing processing requests that have been started. The valid values are **True** and **False**.

### **timeout**

The number of seconds to wait before the operation forces the application server to suspend.

## Total Server Shutdown operation

The **Total Server Shutdown** operation shuts down an application server and waits for it to shut down completely. If the application server is also the ADMIN server for the WebLogic, it shuts down the WebLogic totally.

### **name**

The name of the application server to be shut down.

### **ignoreSessions**

Specifies whether or not the operation should ignore pending HTTP requests while finishing processing requests that have been started. The valid values are **True** and **False**.

**timeout**

The number of seconds to wait before the operation forces the shutdown of the application server.

## Undeploy Application operation

The **Undeploy Application** operation undeploys an application from the target application server or servers on the WebLogic server.

**name**

The name of the application.

**target1**

The application server to target. You can add more application servers to target by creating additional target inputs using sequential numbers (for example: **target2**, **target3**, **target4**). Once a null target is found, targets with higher numbers are not checked.

**id**

An ID for the deployment task. If you do not specify an ID, the operation automatically generates one.

## Wait For State flow

The **Wait For State** flow waits for an application to reach its intended state on the specified WebLogic server.

**application**

The name of the application.

**server**

The name of the application server on which to wait for the application to reach its intended state.

**id**


An ID for the deployment task. If you do not specify an ID, the operation automatically generates one.

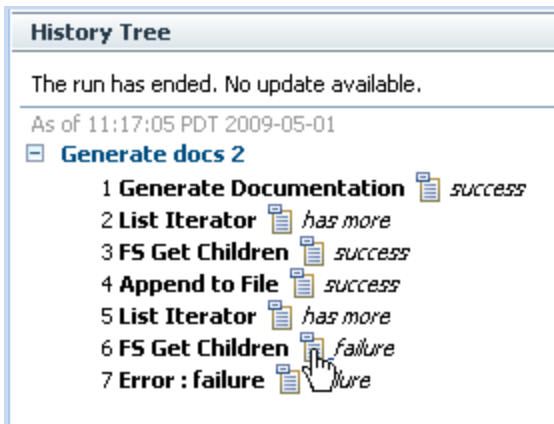
## Troubleshooting

This section provides troubleshooting procedures and tools you can use to solve problems you may encounter while using this integration. It also includes a list of the error messages you may receive while using the integration and offers descriptions and possible fixes for the errors.

### General troubleshooting procedures and tools

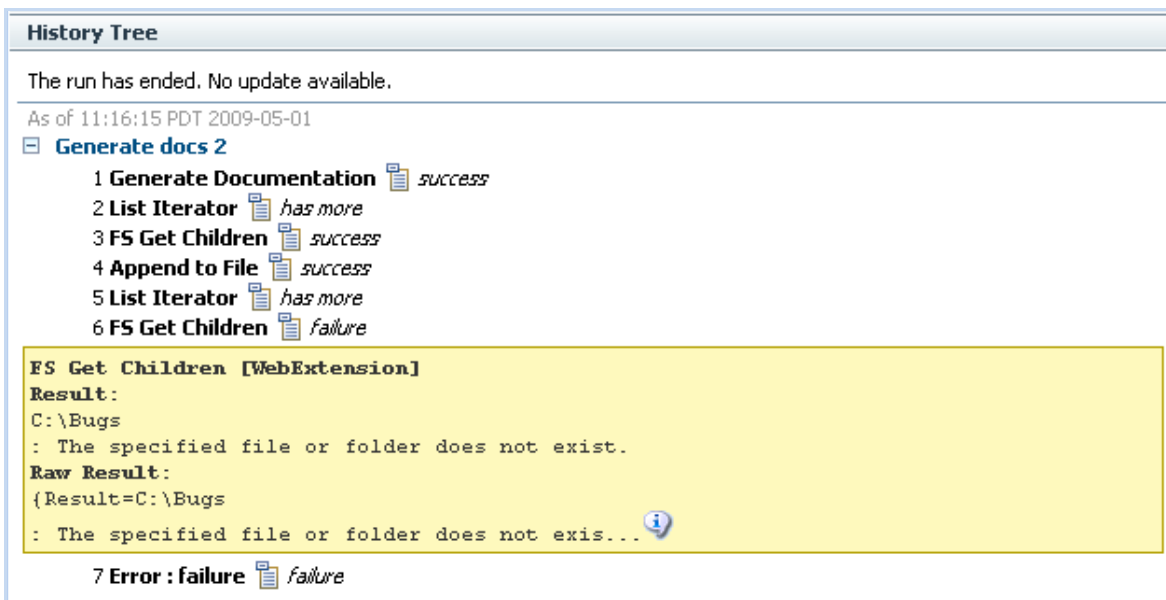
#### If a flow fails in OO Central

1. In Central, expand the **History Tree** of the flow execution and click the **expand details** () icon for a failed step.



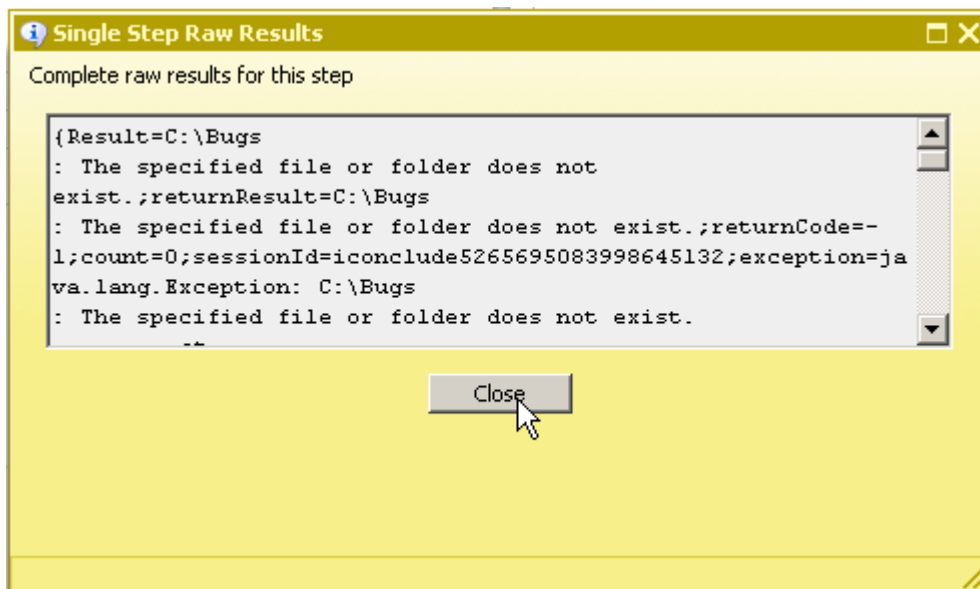
**Figure 3 - Central History Tree for failed flow**

A yellow text box opens to show the execution result of the failed step.



**Figure 4 - Yellow box showing the result of the failed step**

2. Click the **more...** (i) icon to open the **Single Step Raw Results** window.



**Figure 5 - Single Step Raw Results window**

If you run a flow in OO Studio using **Debug Flow in Central**, you can use the Step Result Inspector to check the errors and exceptions in detail.

Similar information can also be found in the OO Central log %OO\_home%/Central/logs/Central\_wrapper.log. For information about the RAS service, check the log %OO\_home%/RAS/Java/Default/webapp/logs/wrapper.log.

## Error messages

### **java.net.ConnectException: Connection refused: connect**

The RSJRAS might not be started or it may have started with errors. Please make sure that the JRAS service starts without any problems.

### **exception=java.io.IOException ...Caused by: javax.naming.CommunicationException [Root exception is java.net.ConnectException: t3://ros-zhizhou:7001: Destination unreachable; nested exception is: java.net.ConnectException: Connection refused: connect; No available router to destination]**

The target WebLogic might not be started or the host for the flow doesn't have WebLogic running. Please make sure that WebLogic starts without any problems and make sure that the host input is correct.

### **exception=java.lang.Exception: No elements were found that match the specified filters.**

The attribute value you entered to limit the query might not be valid. For example, if you use the **Query Server Configuration** flow and enter **examplesServer5** for the **Name** input when **exampleServer5** is not in the target WebLogic, you get this error. Or, if you use the **Query Server Runtime** flow and enter **examplesServer4** for the **name** input when **examplesServer4** is in SHUTDOWN state, you get this error as well.

### **exception=java.lang.SecurityException: User: weblogic, failed to be authenticated.**

Either the username or password used to connect to the Weblogic is not correct. Please make sure to enter correct username and password.

### **exception=java.lang.Exception: no such MBean: badServerName**

You are trying to query or perform operations on a nonexistent application server.

**exception=java.lang.Exception: Attribute: 'badAttr' does not appear to exist**

You entered an incorrect attribute name when you ran the **Configure Server Attribute** operation or the **Configure Application Deployment Attribute** operation. Make sure that the attribute you enter exists. For attribute information regarding application server, please check the [Query Server Configuration input list](#) appendix. For attribute information regarding application deployment, see the [Query Application Deployment Configuration input list](#) appendix.

**exception=javax.management.MBeanException: MBean invoke failed:  
weblogic.management.ManagementException: [Deployer:149001]No application named 'logging-helloworld' exists for operation undeploy**

You attempted to undeploy an application that is not on the application server. Make sure that the application and application server names are correct.

**exception=javax.management.MBeanException: MBean invoke failed:  
weblogic.management.ManagementException: [Deployer:149003]Unable to access application source information in 'c:\temp\noneixting.ear' for application 'logging-helloworld'. The specific error is: No application files exist.**

The input source file entered is not on the system where WebLogic is running. Make sure the file is at the location where the WebLogic Server can access it.

**exception=java.lang.NoClassDefFoundError:weblogic/management/deploy/DeploymentData**

The proper WebLogic .jar files, including weblogic.jar, may not be in the %OO\_home%/RAS/Java/Default/repository/lib/BEA/ folder. When a NoClassDefFoundError occurs, make sure that the proper WebLogic libraries are in above-mentioned OO folder. See [Installation and configuration instructions](#) for information on how to set up this OO integration with WebLogic server.

**exception=java.lang.NoClassDefFoundError: Could not initialize class  
weblogic.rjvm.ServerURL**

The proper WebLogic .jar files, including com.bea.core.weblogic.socket.api\_1.0.0.0.jar for integrating with WebLogic 10.3, might not be in the %OO\_home%/RAS/Java/Default/repository/lib/BEA/folder. When a NoClassDefFoundError occurs, make sure that the proper WebLogic libraries are in above-mentioned OO folder. See [Installation and configuration instructions](#) for information on how to set up this OO integration with WebLogic server.

**exception =FailureMessage=AxisFault...faultCode: {http://xml.apache.org/axis/}HTTP  
...faultString: (404)Not Found...{:}return code: 404**

The error is most likely when the user has already had integration with WebLogic 9.2 . After copying all the necessary libraries to <OOInstallDir>/RAS/Java/Default/repository/lib/BEA, Users might have weblogic.jar, wlclient.jar, wljmxclient.jar copied to the <OOInstallDir>/RAS/Java/Default/webapp/WEB-INF/lib folder as well. To correct, remove any WebLogic jars from <OOInstallDir>/RAS/Java/Default/webapp/WEB-INF/lib.

**exception=java.net.MalformedURLException: Unsupported protocol: t3**

The error indicates that the libraries of WebLogic are not available. To correct, check the "Installation and configuration" section for how to configure the integration with WebLogic.

## Security

This section describes how security is handled by the WebLogic integration.



The WebLogic server is accessed using JMX API over HTTP or T3. The WebLogic administrator provides logon credentials for connecting with JMX API. The client of JMX API needs the username and password of an admin user who can access the WebLogic domain, server information, and application information, and who can perform configuration and operations on them.

## Tools

Following are OO tools that you can use with the HP UCMDB integration:

- **RSFlowInvoke.exe** and **JRSFlowInvoke.jar**

RSFlowInvoke (RSFlowInvoke.exe or the Java version, JRSFlowInvoke.jar) is a command-line utility that allows you to start a flow without using Central (although the Central service must be running). RSFlowInvoke is useful when you want to start a flow from an external system, such as a monitoring application that can use a command line to start a flow.

- **Web Services Wizard (wswizard.exe)**

When you run the Web Services Wizard, you provide it with the WSDL for a given Web service. The WSDL string you provide as a pointer can be a file's location and name or a URL. The Web Services Wizard displays a list of the methods in the API of the Web service that you specify. When you run the wizard, pick the methods you want to use, and with one click for each method you have selected, the wizard creates an HP OO operation that can execute the method. This allows you to use the Web Services Wizard to create operations from your monitoring tool's API.

These tools are located in the %OO\_home%/Studio/tools/ folder.

# Appendices

## Query Application Deployment Configuration input list

Following are additional inputs for the *Query Application Deployment Configuration* operation.

### **AbsoluteInstallDir**

This is the fully resolved location of the application's installation root directory on the admin server.

### **AbsolutePlanDir**

This is the fully resolved location of the application's deployment plan directory on the admin server.

### **AbsolutePlanPath**

This is the fully resolved location of the application's deployment plan on the admin server.

### **AbsoluteSourcePath**

This is the fully resolved location of the application's source files on the admin server.

### **ApplicationIdentifier**

This is the Application Identifier of the application version which uniquely identifies the application version across all versions of all applications. If the application is not versioned, the Application Identifier is the same as the application name.

### **ApplicationName**

This is the name of the application. Note that the name of the current MBean is not the name of the application.

### **AutoDeployedApp**

This input specifies that the application was autodeployed (whether the application was autodeployed in this session or not).

### **BackgroundDeployment**

This input specifies that the application should be deployed in the background. This is only allowed for internal applications and should be used with caution.

### **CompatibilityName**

This is set for beans that are created as a result of a conversion from an 8.1 application configured using ApplicationMBean and ComponentMBean. Standalone modules in 8.1 have both an ApplicationMBean name and ComponentMBean name. This input stores the name of the ComponentMBean name to be used when the server creates the transient ComponentMBean for backward compatibility.

### **DeploymentOrder**

This is a numerical value that indicates when this unit is deployed during startup relative to other DeployableUnits on a server. Units with lower Load Order values are deployed before those with higher values.

### **InstallDir**

This is the path to application's install-root directory, relative to the domain\config\deployments\ directory. When the Install Directory is specified, **SourcePath**, **PlanDir**, and **PlanPath** are derived from this path and need not be specified. The default value is the name of the deployment.

### **InternalApp**

This input indicates whether the application is an internal application. Such applications are not displayed in the console or persisted in the config.xml.

### **LocalInstallDir**

This is the location of the application's installation root directory on the current server. This method throws an unchecked Illegal State Exception if not invoked from within the context of a server.

### **LocalPlanDir**

This is the location of the application's deployment plan directory on the current server. This method throws an unchecked Illegal State Exception if not invoked from within the context of a server.

### **LocalPlanPath**

This is the location of the application's deployment plan on the current server. This method throws an unchecked Illegal State Exception if not invoked from within the context of a server.

### **LocalSourcePath**

This is the location of the application's source files on the current server. This method throws an unchecked Illegal State Exception if not invoked from within the context of a server.

### **ModuleType**

These are the values that match those defined by jsr88. This input may move to another MBean.

### **Name**

This is the unique identifier for this bean instance. For example, **SamplesSearchWebApp**.

### **Notes**

This is optional information that you can include to describe this configuration. The WebLogic server saves this note in the domain's configuration file (config.xml) as XML PCDATA. All left angle brackets (&lt;) are converted to the XML entity &amp;lt;. Carriage returns and line feeds are preserved. **Note:** If you create or edit a note from the Administration Console, the Administration Console does not preserve carriage returns or line feeds.

### **PlanDir**

This is the location of the application's configuration area. This directory can contain external descriptor files as specified within the deployment plan document. If **InstallDir** is null, the plan directory should be an absolute path. If the plan directory is a relative path, it is resolved relative to **InstallDir**. Use **AbsolutePlanDir** to get a fully resolved value.

### **PlanPath**

This is the path to the deployment plan document on the Administration server. If **PlanDir** is null, the plan path must be an absolute path. If the plan path is a relative path, it is resolved relative to **PlanDir**. Use **AbsolutePlanPath** to get a fully resolved value. If there is no plan, this returns "no plan specified".

### **RootStagingDir**

This is the root directory under which the application is staged. This method throws an unchecked Illegal State Exception if not invoked from within the context of a server.

### **SecurityDDModel**

This is the security model which is used to secure a deployed module. To set this value, you can use the **WebLogic.Deployer** command-line tool, the **Deployment Assistant** in the Administration Console, the **WebLogic Scripting Tool (WLST)**, or another JMX client. If you deploy a module using one of the previously mentioned tools and you do not specify a security model value, the module is secured with the security realm's default model.

If you deploy a module by modifying the domain's config.xml file and restarting the server, but you do not specify a security model value for the module, the module is secured with the DDOnly model, which is the default value of this **AppDeploymentMBean** input.

In summary, the order of precedence for the value of this input is as follows:

- If you deploy a module using a runtime deployment utility, the order of precedence is the value set by the deployment utility. This is the value set as the security realm's default security model.
- If you deploy a module by modifying config.xml and restarting the server, the order of precedence is the value that you specify for the module in config.xml. This is the default value of this **AppDeploymentMBean SecurityDDModel** input.

#### **SourcePath**

This is the path to the source of the deployable unit on the Administration server. If **InstallDir** is null, the source path must be an absolute path. If the source path is relative, it is resolved relative to InstallDir/app/. Use **AbsoluteSourcePath** to get a fully resolved value.

#### **StagingMode**

This is the mode that specifies whether a deployment's files are copied from a source on the Administration server to the Managed server's staging area during application preparation. You can only set the staging mode for an application the first time you deploy it. After you set the staging mode for an application, you cannot change it while the application is configured in the domain. The only way to change the staging mode is to undeploy, then redeploy the application. This input overrides the server's staging mode.

#### **Type**

This input returns the type of the MBean.

#### **ValidateDDSecurityData**

This input is not used in the current BEA release.

#### **VersionIdentifier**

This input uniquely identifies the application version across all versions of the same application. If the application is not versioned, this input returns null.

## Query Application Deployment Configuration Target input list

Following are additional inputs for the [Query Application Deployment Configuration Target operation](#).

Please see the inputs in [Query Server Configuration input List](#).

## Query Server Configuration input list

Following are additional inputs for the [Query Server Configuration operation](#).

#### **AcceptBacklog**

This is the number of backlogged new TCP connection requests allowed for this server's regular and SSL ports. Setting the backlog to 0 may prevent this server from accepting any incoming connections on some operating systems.

#### **AdminReconnectIntervalSeconds**

This is the number of seconds between reconnection attempts to the administration server. When the administration server fails, the managed server periodically tries to reconnect to it.

### **AdministrationPort**

This is the secure administration port for the server. This port requires that you enable the domain's administration port and that SSL is configured and enabled. By default, the server uses the administration port that is specified at the domain level. To override the domain-level administration port for the current server instance, set the server's administration port.

### **AdministrationPortEnabled**

This input indicates whether or not the administration port is enabled for the server. This field is derived from the DomainMBean and has no additional settings. All servers (7.0 and later) in a single domain either have or do not have an administration. The administration port uses Single Sign-On (SSL), so SSL must be properly configured and enabled for the port to be active.

### **AdministrationProtocol**

This input returns the protocol that is used for administrative connections when none is specified.

### **AutoKillIfFailed**

This input specifies whether the Node Manager should automatically kill the specified server if its health state is **failed**.

### **AutoMigrationEnabled**

This input specifies whether the Node Manager automatically restarts the specified server and its services on another machine if the server fails.

### **AutoRestart**

This input specifies whether the Node Manager automatically restarts the specified server if it crashes or otherwise or goes down unexpectedly.

### **COMEnabled**

This input specifies whether COM support is enabled on the regular (non-SSL) port. COM is not supported on the SSL port.

### **ClasspathServletDisabled**

ClasspathServlet serves any class file in the classpath and is registered by default in every Web application (including management). ClasspathServletDisabled is not needed for many applications, and represents a security hole if you set it to be true. By default it is false.

### **ClientCertProxyEnabled**

This input specifies whether the HttpClusterServlet proxies the client certificate in a special header. By default (or if you specify **false**), the WebLogic.xml deployment descriptor for each Web application deployed on the specified server determines whether the Web application trusts certificates sent from the proxy server plugin. Also by default (or if you specify **false**), you cannot log on to the Web application from a proxy server plugin. A value of **true** causes proxy-server plugins to pass identity certifications from clients to all Web applications that are deployed on this server instance.

A proxy-server plugin encodes each identify certification in the WL-Proxy-Client-Cert header and passes the header to WebLogic server instances. A WebLogic server instance takes the certificate information from the header, trusting that it comes from a secure source, and uses that information to authenticate the user. If you specify **true**, use a WebLogic.security.net.ConnectionFilter to ensure that this WebLogic server instance only accepts connections from the machine on which the proxy-server plugin is running. Specifying **true** without using a connection filter creates a security vulnerability as the WL-Proxy-Client-Cert header can create a security hole. A cluster can also specify whether the HttpClusterServlet proxies the client certificate in a special header. The cluster-level setting overrides the setting in individual servers that are part of the cluster.

### **ClusterWeight**

This input specifies the proportion of the load that the server can bear relative to other servers in a cluster. If all servers have the default weight or the same weight, each bears an equal proportion of the load. If one server has weight 50 and all other servers have weight 100, the 50-weight server bears half as much load as any other server.

### **CompleteCOMMessageTimeout**

This input specifies the maximum number of seconds the server waits for a complete COM message to be received. This setting does not apply to network channels that you have configured for the server. This timeout helps guard against a denial of service attack in which callers indicate that they will be sending messages of a certain size which they never finish sending.

### **CompleteHTTPMessageTimeout**

This input specifies the maximum number of seconds the server waits for a complete HTTP message to be received. If you configure network channels for this server, each channel can override this HTTP message timeout. This timeout helps guard against a denial of service attack in which callers indicate that they will be sending messages of a certain size which they never finish sending. A value of **-1** indicates that this value should be obtained from network channels configured for this server.

### **CompleteIIOpMessageTimeout**

This input specifies the maximum number of seconds the server waits for a complete IIOp message to be received. This timeout helps guard against denial of service attacks in which callers indicate that they will be sending messages of a certain size which they never finish sending.

### **CompleteMessageTimeout**

This input specifies the maximum number of seconds the server waits for a complete message to be received. If you configure network channels for this server, each channel can override this message timeout. This timeout helps guard against a denial of service attack in which callers indicate that they will be sending messages of a certain size which they never finish sending.

### **CompleteT3MessageTimeout**

This input specifies the maximum number of seconds the server waits for a complete T3 message to be received. If you configure network channels for this server, each channel can override this T3 message timeout. This timeout helps guard against a denial of service attack in which callers indicate that they will be sending messages of a certain size which they never finish sending.

### **ConsensusProcessIdentifier**

This input specifies the identifier for consensus-based algorithms. Each server should have a unique identifier indexed from 0.

### **ConsoleInputEnabled**

This input is **true** if commands can be typed at the console.

### **CustomIdentityKeyStoreFileName**

This input specifies the path and file name of the identity keystore. The path name can be absolute or relative to where the server was booted. The custom identity keystore file name is only used if KeyStores is **CUSTOM\_IDENTITY\_AND\_JAVA\_STANDARD\_TRUST**, **CUSTOM\_IDENTITY\_AND\_CUSTOM\_TRUST**, or **CUSTOM\_IDENTITY\_AND\_COMMAND\_LINE\_TRUST**.

### **CustomIdentityKeyStorePassPhrase**

This input specifies the encrypted custom identity keystore's passphrase. If the value is empty or null, the keystore is opened without a passphrase. This input is only used if KeyStores is **CUSTOM\_IDENTITY\_AND\_JAVA\_STANDARD\_TRUST**,

**CUSTOM\_IDENTITY\_AND\_CUSTOM\_TRUST**, or  
**CUSTOM\_IDENTITY\_AND\_COMMAND\_LINE\_TRUST**.

When you retrieve the value of this attribute, the WebLogic server does the following:

- Retrieves the value of the **CustomIdentityKeyStorePassPhraseEncrypted** input.
- Decrypts the value and returns the unencrypted password as a string.

When you set the value of this input, the WebLogic server does the following:

- Encrypts the value.
- Sets the value of the **CustomIdentityKeyStorePassPhraseEncrypted** input to the encrypted value.

The **CustomIdentityKeyStorePassPhrase** input is a potential security risk as the **String** object (which contains the unencrypted password) remains in the JVM's memory until garbage collection removes it and the memory is reallocated. Depending on how memory is allocated in the JVM, a significant amount of time can pass before this unencrypted data is removed from memory. Instead of using this input, we recommend using the **CustomIdentityKeyStorePassPhraseEncrypted** input.

### **CustomIdentityKeyStorePassPhraseEncrypted**

This input returns the encrypted passphrase you defined when you created the keystore.

### **CustomIdentityKeyStoreType**

This input specifies the type of keystore. Usually this is **JKS**. If the input value is empty or null, the JDK's default keystore type (specified in `java.security`) is used. The custom identity keystore type is only used if KeyStores is

**CUSTOM\_IDENTITY\_AND\_JAVA\_STANDARD\_TRUST**,  
**CUSTOM\_IDENTITY\_AND\_CUSTOM\_TRUST**, or  
**CUSTOM\_IDENTITY\_AND\_COMMAND\_LINE\_TRUST**.

### **CustomTrustKeyStorePassPhraseEncrypted**

This input specifies the custom trust keystore's encrypted passphrase. If the input value is empty or null, the keystore can be opened without a passphrase. This input is only used if KeyStores is **CUSTOM\_IDENTITY\_AND\_CUSTOM\_TRUST**. To set this input, use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the value. Then set this input to the output of the `encrypt()` method. To compare a password that a user enters with the encrypted value of this input, use the same WebLogic server instance you used to set and encrypt this input and use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the user-supplied password. Then compare the encrypted values.

### **CustomTrustKeyStoreType**

This input specifies the type of the keystore. Usually this is **JKS**. If the input value is empty or null, the JDK's default keystore type (specified in `java.security`) is used. This keystore type is only used if KeyStores is **CUSTOM\_IDENTITY\_AND\_CUSTOM\_TRUST**.

### **DefaultIIOPPassword**

This input specifies the password for the default IIOP user. (This requires you to enable IIOP.) As of WebLogic 8.1 sp4, when you get the value of this input, the WebLogic server does the following:

- Retrieves the value of the **DefaultIIOPPasswordEncrypted** input.
- Decrypts the value and returns the unencrypted password as a string.

The **DefaultIIOPPassword** attribute is a potential security risk as the **String** object (which contains the unencrypted password) remains in the JVM's memory until garbage collection removes it and the memory is reallocated. Depending on how memory is allocated in the JVM, a significant amount of time can pass before this unencrypted data is removed from memory.

Instead of using this attribute, we recommend using the **DefaultIIOPPasswordEncrypted** input.

### **DefaultIIOPPasswordEncrypted**

This input specifies the encrypted password for the default IIOP user. To set this attribute, use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the value. Then set this attribute to the output of the `encrypt()` method. To compare a password that a user enters with the encrypted value of this attribute, go to the same WebLogic server instance you used to set and encrypt this attribute and use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the user-supplied password. Then compare the encrypted values.

### **DefaultIIOPUser**

This input specifies the user name of the default IIOP user. (This requires you to enable IIOP.)

### **DefaultInternalServletsDisabled**

This input specifies whether the default servlets in the servlet engine are disabled. The default servlets include:

- `weblogic.servlet.ClasspathServlet`
- `weblogic.servlet.utils.iiop.GetIORServlet`
- `weblogic.rjvm.http.TunnelSendServlet`
- `weblogic.rjvm.http.TunnelRecvServlet`
- `weblogic.rjvm.http.TunnelLoginServlet`
- `weblogic.rjvm.http.TunnelCloseServlet`

If the input value is set to **true**, this property overrides the `ClasspathServletDisabled` property.

### **DefaultProtocol**

This input specifies the protocol to use for connections when none is specified.

### **DefaultSecureProtocol**

This input specifies the protocol to use for secure connections when none is specified.

### **DefaultTGIOPPassword**

This input specifies the password for the default user associated with the Tuxedo GIOP (TGIOP) protocol. (This requires you to configure the WebLogic Tuxedo Connector (WTC) for this server.)

As of WebLogic 8.1 sp4, when you get the value of this attribute, the WebLogic server does the following:

- Retrieves the value of the `DefaultTGIOPPasswordEncrypted` attribute.
- Decrypts the value and returns the unencrypted password as a string.

When you set the value of this attribute, the WebLogic server does the following:

- Encrypts the value.
- Sets the value of the `DefaultTGIOPPasswordEncrypted` attribute to the encrypted value.

The **DefaultTGIOPPassword** attribute is a potential security risk in as the **String** object (which contains the unencrypted password) remains the JVM's memory until garbage collection removes it and the memory is reallocated. Depending on how memory is allocated in the JVM, a significant amount of time can pass before this unencrypted data is removed from memory.

Instead of using this attribute, we recommend using the **DefaultTGIOPPasswordEncrypted** input.

### **DefaultTGIOPPasswordEncrypted**

This input specifies the encrypted password for the default TGIOP user. To set this attribute, use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the value. Then set this attribute to the output of the `encrypt()` method. To compare a password that a user enters with



the encrypted value of this attribute, go to the same WebLogic server instance you used to set and encrypt this attribute and use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the user-supplied password. Then compare the encrypted values.

#### **DefaultTGIOPUser**

This input specifies the default user associated with the Tuxedo GIOP (TGIOP) protocol. (This requires you to configure the WebLogic Tuxedo Connector (WTC) for this server.)

#### **DGCIIdlePeriodsUntilTimeout**

This input specifies the number of idle periods allowed before object is collected.

#### **ExpectedToRun**

This input specifies whether this server is expected to run if the domain is started.

#### **ExtraEjbcOptions**

This input specifies the options passed to the EJB compiler during server-side generation. Each EJB component can override the compiler options that you specify here. The following options are valid:

- `-forcegeneration` forces generation of wrapper classes. Without this flag, the classes may not be regenerated if it is determined to be unnecessary.
- `-disableHotCodeGen` generates `ejb` stub and `skel` as part of `ejbc`. For better performance, avoid `HotCodeGen`.
- `-keepgenerated` keeps the generated `.java` files.
- `-compiler javac` specifies the Java compiler to execute. If you do not specify the `-compiler javac` option, the `-compilerclass` option is used.
- `-compilerclass com.sun.tools.javac.Main` specifies the compiler class to invoke.
- `-g` compiles debugging information into a class file.
- `-normi` passes through to Symantec's `sj`.
- `-classpath path` specifies which classpath to use.
- `-source source` specifies the source version.
- `-Joption` specifies the flags passed through to the java runtime.

#### **ExtraRmicOptions**

This input specifies the options passed to the RMIC compiler during server-side generation. Each EJB component can override the compiler options that you specify here.

#### **ExternalDNSName**

This input specifies the external IP address or DNS name for this server. This address is sent with HTTP session cookies and with dynamic server lists to HTTP proxies. It is also used by external application clients to enable the propagation of RMI traffic through network address translating (NAT) firewalls. Unless clients are accessing the WebLogic server using T3 and the default channel, you must specify an external DNS name for configurations in which a firewall is performing network address translation. For example, define the external DNS name for configurations in which a firewall is performing network address translation, and clients are accessing the WebLogic server using HTTP via a proxy plug-in.

#### **GracefulShutdownTimeout**

This input specifies the number of seconds a graceful shutdown operation waits before forcing a shut down. A graceful shutdown gives WebLogic server subsystems time to complete certain application processing. If subsystems are unable to complete this processing within the number of seconds that you specify in this input, the server forces an automatic shutdown. Use an input value of **0** to specify that the server will wait indefinitely for graceful shutdown to complete. The graceful shutdown timeout applies only to graceful shutdown operations.

**HealthCheckIntervalSeconds**

This input specifies the number of seconds that define the frequency that the server monitors the health of its subsystems and changes the server's overall state, if required.

**HealthCheckStartDelaySeconds**

This input specifies the number of seconds the Node Manager waits before starting to monitor the server.

**HealthCheckTimeoutSeconds**

This input specifies the number of seconds the Node Manager waits before timing out its health query to the server. If the timeout is reached, the Node Manager assumes that the Managed server has failed.

**HostsMigratableServices**

This input retrieves the **hostsMigratableServices** attribute of the ServerMBean object.

**HttpdEnabled**

This input specifies whether or not HTTP support is enabled on the regular port or SSL port.

**HttpTraceSupportEnabled**

This input returns the **HttpTraceSupportEnabled** value.

**IdleIIOPConnectionTimeout**

This input specifies the maximum number of seconds that an IIOP connection is allowed to be idle before it is closed by the server. This timeout helps guard against server deadlock because there are too many open connections.

**IdleConnectionTimeout**

This input specifies the maximum number of seconds that a connection is allowed to be idle before it is closed by the server. The T3 and T3S protocols ignore this input. If you configure network channels for the server, each channel can override this idle connection message timeout. This timeout helps guard against server deadlock because there are too many open connections.

**IdlePeriodsUntilTimeout**

This input specifies the number of idle periods that can occur until a peer is considered unreachable.

**IIOPEnabled**

This input specifies whether the server has IIOP support enabled for both the regular (non-SSL) and SSL ports.

**IIOPTxMechanism**

This input configures IIOP propagate transactions using the WebLogic-specific JTA or the OMG-specified OTS. You cannot use both as this affects the way transactions are negotiated.

**IgnoreSessionsDuringShutdown**

This input indicates whether or not a graceful shutdown operation drops all HTTP sessions immediately. If this is set to **false**, a graceful shutdown operation waits for HTTP sessions to complete or timeout.

**InstrumentStackTraceEnabled**

This input specifies whether the server returns stack traces for RMI calls that generate exceptions. With RMI stack tracking enabled, if a client issues an RMI call to a server subsystem or a module running within the server, and if the subsystem or module generates an exception that includes a stack trace, the server returns the exception as well as the stack trace. With RMI stack tracking disabled, the server returns the exception without the stack trace details.

**InterfaceAddress**

This input specifies the IP address of the NIC that the server uses for multicast traffic.

## JavaCompiler

This input specifies the Java compiler to use for all applications hosted on the server that need to compile Java code.

## JavaCompilerPreClassPath

This input specifies the options to prepend to the Java compiler classpath when compiling Java code.

## JavaCompilerPostClassPath

This input specifies the options to append to the Java compiler classpath when compiling Java code.

## JavaStandardTrustKeyStorePassPhrase

This input specifies the password for the Java Standard Trust keystore. This password is defined when the keystore is created. If the input value is empty or null, the keystore is opened without a passphrase. This input is only used if KeyStores is

**CUSTOM\_IDENTITY\_AND\_JAVA\_STANDARD\_TRUST** or **DEMO\_IDENTITY\_AND\_DEMO\_TRUST**.

When you get the value of this input, the WebLogic server does the following:

- Retrieves the value of the **JavaStandardTrustKeyStorePassPhraseEncrypted** attribute.
- Decrypts the value and returns the unencrypted password as a string.

When you set the value of this input, WebLogic Server does the following:

- Encrypts the value.
- Sets the value of the **JavaStandardTrustKeyStorePassPhraseEncrypted** attribute to the encrypted value.

Using the **JavaStandardTrustKeyStorePassPhrase** input poses a potential security risk as the **String** object (which contains the unencrypted password) remains in the JVM's memory until garbage collection removes it and the memory is reallocated. Depending on how memory is allocated in the JVM, a significant amount of time can pass before this unencrypted data is removed from memory. Instead of using this input, we recommend that you use the **JavaStandardTrustKeyStorePassPhraseEncrypted** input.

## JavaStandardTrustKeyStorePassPhraseEncrypted

This input specifies the encrypted password for the Java Standard Trust keystore. This password is defined when the keystore is created. To set this input, use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the value. Then set this input to the output of the `encrypt()` method. To compare a password that a user enters with the encrypted value of this input, go to the same WebLogic server instance that you used to set and encrypt this input and use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the user-supplied password. Then compare the encrypted values.

## JDBCLLRTTableName

This input specifies the table name for the server's Logging Last Resource (LLR) database tables. The WebLogic server creates the tables and then uses them during transaction processing for the LLR transaction optimization. This setting must be unique for each server. The default table name is `WL_LLRT_SERVERNAME`. This setting only applies if the server hosts one or more LLR-enabled JDBC data sources.

The format for the tables that the WebLogic server creates is `[[[catalog.]schema.]name`. Each `\."` in the table name is significant, and the schema generally corresponds to a username in many databases.

**IMPORTANT:** If this value is changed, but the LLR table already exists in the database, you must preserve the existing table's data. Consequently, when changing the table name, the existing database table must be renamed by a database administrator to match the new

configured table name. Otherwise, transaction records may be lost, resulting in heuristic failures that aren't logged.

**IMPORTANT:** Each server's table name must be unique. Multiple LLR-enabled data sources within the same server may share the same table, but multiple servers must not share the same table. If multiple same-named servers share a table, the behavior is undefined and it is likely that transactions will not recover properly after a crash, creating heuristic hazards.

### **JDBCLoggingEnabled**

This input specifies whether the server maintains a JDBC log file.

### **JDBCLoginTimeoutSeconds**

This input retrieves the JDBC Login Timeout value. The specified value is passed into `java.sql.DriverManager.setLoginTimeout()`. Note that this `DriverManager` setting impacts *\*all\** JDBC drivers loaded into this JVM. This feature is disabled by default.

### **JMSDefaultConnectionFactoryEnabled**

This input specifies whether the server uses JMS default connection factories. The WebLogic server provides the following JMS default connection factories:

- `weblogic.jms.ConnectionFactory`
- `weblogic.jms.XAConnectionFactory`

An XA factory is required for JMS applications to use JTA user-transactions, but is not required for transacted sessions. All other preconfigured attributes for the default connection factories are set to the same default values as a user-defined connection factory. If the preconfigured settings of the default factories are appropriate for your application, you do not need to configure any additional factories for your application. **Note:** When using the default connection factories, you have no control over targeting the WebLogic server instances where the connection factory may be deployed. However, you can disable the default connection factories on a per-server basis. To deploy a connection factory on independent servers, on specific servers within a cluster, or on an entire cluster, configure a connection factory and specify the appropriate server targets.

### **JMSThreadPoolSize**

This input specifies the size of the JMS execute thread pool. **Note:** Incoming RMI calls execute in the JMS execute queue/thread pool, if one exists; otherwise, they execute in the default execute queue. Additional executes (work that cannot be completed in the initial RMI thread) are executed in the default execute queue. The difference in setting up a JMS-specific thread pool is that JMS is not be starved by other execute threads.

### **JNDITransportableObjectFactoryList**

This input specifies the list of factories that create transportable objects.

### **KeyStores**

This input specifies which configuration rules are used for finding the server's identity and trust keystores.

### **ListenAddress**

This input specifies the IP address or DNS name that the server uses to listen for incoming connections. Servers can be reached through the following URL: `protocol://listen-address:listen-port`. Any network channel that you configure for the server can override this listen address. If a server's listen address is undefined, clients can reach the server through one of the following:

- The IP address of the computer that hosts the server
- A DNS name that resolves to the host
- The localhost string

The localhost string can only be used for requests from clients running on the same computer as the server. If you want to limit the valid addresses for a server instance, specify one of the following:

- An IP address. Clients can specify either the IP address or a DNS name that maps to the IP address. Clients that specify an IP address and attempt to connect through an SSL port must disable hostname verification.

If you specify an IP address for **ListenAddress** and then a client request specifies a DNS name, the WebLogic server attempts to resolve the DNS name, but if it cannot access DNS name mapping, the request fails.

- A DNS name. Clients can specify either the DNS name or the corresponding IP address. Do not leave the listen address undefined on a Windows computer that uses multiple IP addresses (a multihomed computer) or the server binds to all available IP addresses.

**Note:** To resolve a DNS name to an IP address, the WebLogic server must be able to contact an appropriate DNS server or obtain the IP address mapping locally. So, if you specify a DNS name for the listen address, you must either leave a port open long enough for the WebLogic server instance to connect to a DNS server and cache its mapping, or you must specify the IP address mapping in a local file.

### **ListenDelaySecs**

This input is perpetuated for compatibility with WebLogic 6.1 only.

### **ListenersBindEarly**

This input determines whether the server binds server sockets early. Early binding detects port conflicts quickly and also provides user feedback on the default listen port as to the server state.

### **ListenPort**

This input specifies the default TCP port that the server uses to listen for regular (non-SSL) incoming connections. If this port is disabled, the SSL port must be enabled. Additional ports can be configured using network channels. The cluster (multicast) port is configured separately.

### **ListenPortEnabled**

This input specifies whether the server can be reached through the default plain-text (non-SSL) listen port. If you disable the listen port, you must enable the default SSL listen port. You can define additional listen ports for the server by configuring network channels.

### **ListenThreadStartDelaySecs**

This input returns the maximum amount of time that the server waits for server sockets to bind before starting a listen thread.

### **LogRemoteExceptionsEnabled**

This input specifies whether the server message log includes exceptions that are raised in remote systems.

### **LoginTimeoutMillis**

This input specifies the login timeout for the server's default regular (non-SSL) listen port. This is the maximum amount of time allowed for a new connection to establish. A value of **0** indicates that there is no maximum.

### **LowMemoryGCThreshold**

This input specifies the threshold level (in percent) that the server uses for logging low memory conditions and changing the server health state to **Warning**. For example, if you specify a value of **5**, the server logs a low memory warning in the log file and changes the server health state to **Warning** after the average free memory reaches 5% of the initial free memory measured at the server's boot time.

### **LowMemoryGranularityLevel**

This input specifies the granularity level (in percent) that the server uses for logging low memory conditions and changing the server health state to **Warning**. For example, if you specify value of **5** and the average free memory drops by 5% or more over two measured intervals, the server logs a low memory warning in the log file and changes the server health state to **Warning**.

### **LowMemorySampleSize**

This input specifies the number of times that the server samples free memory during the time period specified by the **LowMemoryTimeInterval** input. Increasing the sample size can improve the accuracy of the reading.

### **LowMemoryTimeInterval**

This input specifies the amount of time (in seconds) that defines the interval over which the server determines average free memory values. By default, the server obtains an average free memory value every 3600 seconds. This interval is not used if the JRockit VM is used, as the memory samples are collected immediately after a VM-scheduled garbage collection. Taking memory samples after a garbage collection gives a more accurate average value of the free memory.

### **ManagedServerIndependenceEnabled**

This input specifies whether this Managed server can be started when the Administration server is unavailable. In such a case, the Managed server retrieves its configuration by reading a configuration file and other files directly.

### **MaxCOMMessageSize**

This input specifies the maximum number of bytes allowed in messages that are received over the COM protocol. If you configure custom network channels for the server, each channel can override this maximum message size. The maximum message size helps guard against a denial of service attack in which a caller attempts to force the server to allocate more memory than is available and keep the server from responding quickly to other requests. A value of **-1** causes the COM protocol to use the maximums that are specified by channels in order of precedence.

### **MaxHTTPMessageSize**

This input specifies the maximum number of bytes allowed in messages that are received over the HTTP protocol. If you configure custom network channels for the server, each channel can override this maximum message size. The maximum message size helps guard against a denial of service attack in which a caller attempts to force the server to allocate more memory than is available and keep the server from responding quickly to other requests. A value of **-1** causes the HTTP protocol to use the maximums that are specified by channels in order of precedence.

### **MaxIIOPMessageSize**

This input specifies the maximum number of bytes allowed in messages that are received over the IIOP protocol. If you configure custom network channels for the server, each channel can override this maximum message size. The maximum message size helps guard against a denial of service attack in which a caller attempts to force the server to allocate more memory than is available and keep the server from responding quickly to other requests. A value of **-1** causes the IIOP protocol to use the maximums that are specified by channels in order of precedence.

### **MaxMessageSize**

This input specifies the maximum number of bytes allowed in messages that are received over all supported protocols, unless overridden by a protocol-specific setting or a custom channel setting. The order of precedence for setting message size maximums is as follows:

- A channel-wide maximum in a custom network channel
- A protocol-specific setting in the default network channel. See [MaxCOMMessageSize](#), [MaxHTTPMessageSize](#), [MaxIIOPMessageSize](#), and [MaxT3MessageSize](#).

The maximum message size helps guard against a denial of service attack in which a caller attempts to force the server to allocate more memory than is available and keep the server from responding quickly to other requests.

### **MaxOpenSockCount**

This input specifies the maximum number of open sockets allowed in the server at a given point in time. When the maximum threshold is reached, the server stops accepting new requests until the number of sockets drops below the threshold. A value less than **0** indicates an unlimited size.

### **MaxT3MessageSize**

This input specifies the maximum number of bytes allowed in messages that are received over the T3 protocol. If you configure custom network channels for this server, each channel can override this maximum message size. The maximum message size helps guard against a denial of service attack in which a caller attempts to force the server to allocate more memory than is available and keep the server from responding quickly to other requests. A value of **-1** causes the T3 protocol to use the maximums that are specified by channels in the order of precedence.

### **MessagingBridgeThreadPoolSize**

This input returns the size of the messaging bridge execute thread pool.

### **MessageIdPrefixEnabled**

This input indicates whether message IDs in logged messages should include a prefix. Message IDs are 6 digit numeric strings that can be optionally presented in a log entry with a prefix. The prefix used by server messages is `\\"BEA-\\`.

### **MSIFileReplicationEnabled**

This input specifies the whether the Administration server replicates its configuration files to this Managed server. With file replication enabled, the Administration server copies its configuration file and SerializedSystemIni.dat into the Managed server's root directory every 5 minutes. This option does not replicate a boot identity file. No matter what configuration file name you used to start the Administration server, the replicated configuration file is always named msi-config.xml. For example, if you specified `-Dweblogic.ConfigFile=MyConfig.xml` when you started the Administration server and you enabled file replication, the Administration server copies MyConfig.xml and names the copy msi-config.xml. Depending on your backup schemes and the frequency with which you update your domain's configuration, this option might not be worth the performance cost of copying potentially large files across a network.

### **MuxerClass**

This input specifies the muxer class name.

### **Name**

This input specifies an alphanumeric name for the server instance (spaces are not valid.) The name must be unique for all configuration objects in the domain. Within a domain, each server, machine, cluster, JDBC connection pool, virtual host, and any other resource type must be named uniquely and must not use the same name as the domain. The server name is not used as part of the URL for applications that are deployed on the server. It is for identification purposes only. The server name displays in the Administration Console and, if you use WebLogic server command-line utilities or APIs, use this name to identify the server. After you create a server, you cannot change its name. Instead, clone the server and provide a new name for the clone.

### **NativeIOEnabled**

This input specifies whether native I/O is enabled for the server.

### **Notes**

This input specifies the optional information you can include to describe this configuration. The WebLogic server saves this note in the domain's configuration file (config.xml) as XML PCDATA. All left angle brackets (`&lt;`) are converted to the XML entity `&lt;`. Carriage returns and line

feeds are preserved. **Note:** If you create or edit a note from the Administration Console, the Administration Console does not preserve carriage returns and line feeds.

### **OutboundEnabled**

This input specifies whether new server-to-server connections can consider the default server channel when initiating a connection. This is only relevant if the connection needs to be bound to the default listen address, and only works for binary protocols that support both outbound and inbound traffic. When this feature is not enabled, connections are initiated using a local address selected by the underlying hardware. For the default channel, this is usually what is wanted for IP-routing to be effective. Note that since the default is **false**, other outbound channels are considered in preference to the default channel. Default administration channels, created when the domain-wide administration port is turned on, are always considered and bound when initiating an administrative connection. To allow IP-routing for administration traffic, create custom admin with {@link NetworkAccessPointMBean#isOutboundEnabled isOutboundEnabled} set to false instead of enabling the domain-wide ADMIN port.

### **OutboundPrivateKeyEnabled**

This input specifies the whether the SSL identity specified by {@link SSLMBean#ServerPrivateKeyAlias SSLMBean#ServerPrivateKeyAlias} for this server should be used for outbound SSL connections on the default server channel. Under normal circumstances, the outbound identity is determined by the caller's environment.

### **PeriodLength**

This input specifies the time interval in milliseconds of the heartbeat period. A value of **0** indicates that heartbeats are turned off.

### **PreferredSecondaryGroup**

This input defines secondary clustered instances considered for hosting replicas of the primary HTTP session states created on the server.

### **ReplicationGroup**

This input defines preferred clustered instances considered for hosting replicas of the primary HTTP session states created on the server.

### **RestartDelaySeconds**

This input specifies the number of seconds the Node Manager should wait before restarting the server. After killing a server process, the system may need several seconds to release the TCP port(s) the server was using. If the Node Manager attempts to restart the Managed server while its ports are still active, the startup attempt fails. If AutoMigration is enabled and **RestartDelaySeconds** is set to **0**, **RestartDelaySeconds** is automatically set to the lease time. This prevents the server from failing to restart after migration when the previous lease is still valid.

### **RestartIntervalSeconds**

This input specifies the number of seconds during which the server can be restarted, up to the number of times specified in **RestartMax**.

### **RestartMax**

This input specifies the number of times the Node Manager can restart the server within the interval specified in **RestartInterval**.

### **ReverseDNSAllowed**

This input specifies whether the kernel is allowed to perform reverse DNS lookups.

### **SelfTuningThreadPoolSizeMax**

This input sets the maximum thread pool size of the self-tuning thread pool. The self-tuning thread pool starts with the default size of 1. It grows and shrinks automatically as required. Setting this attribute changes the default maximum pool size. The active thread count never increases beyond this value. This value defines the maximum number of threads permitted in



the server. Note that the server adds threads only if it improves throughput. Measurements are taken every two seconds and the decision to increase or decrease the thread count is based on the current throughput measurement versus past values. This attribute is used only when {[@link #setUse81StyleExecuteQueues](#)} is turned off (this is the default).

### **SelfTuningThreadPoolSizeMin**

This input gets the minimum thread pool size of the self-tuning thread pool. The self-tuning thread pool starts with the default size of 1. It grows and shrinks automatically as required. Setting this attribute changes the default minimum pool size. The thread count never shrinks below this value. The self-tuning thread pool can add threads to improve throughput but never decreases below the set minimum. This attribute is used only when {[@link #setUse81StyleExecuteQueues](#)} is turned off (this is the default).

### **ServerLifeCycleTimeoutVal**

This input specifies the number of seconds a force shutdown operation waits before timing out and killing itself. If the operation does not complete within the configured timeout seconds, the server shuts down automatically if the state of the server at that time was SHUTTING\_DOWN. A value of 0 means that the server will wait indefinitely for the life cycle operation to complete.

### **SocketReaders**

This input specifies the number of socket reader threads.

### **ServerVersion**

This input specifies the release identifier for the server. Since this is a configured attribute, it is only as accurate as the configuration. The form of the version is major.minor.servicepack.rollingpatch. Not all parts of the version are required, for instance `"7"` is acceptable.

### **StagingDirectoryName**

This input specifies the directory path on the Managed server where all staged (prepared) applications are placed. If you do not specify an absolute directory name, the path is relative to `rootdirectory/`. Once you configure it, you cannot change the staging directory name. Remove all applications from the server prior to changing this attribute. The default staging directory is `"stage"` relative to the server root.

### **StagingMode**

This input specifies the mode which indicates whether an application's files are copied from a source on the Administration server to the Managed server's staging area during application preparation. During application preparation, the application's files are copied from the source on the Administration server to the Managed server's staging area. If you specify **nostage** or **external\_stage**, the copy does not occur. This is useful when the staging area is a shared directory already containing the application files, or if it is a single server domain. The administrator must ensure that the Managed server's staging directory is set appropriately. Deployment errors result if the application is not available during the preparation or activation of the application. Each application can override the staging mode specified by this input.

### **StartupMode**

This input specifies the state in which the server should be started. If you specify **STANDBY**, you must also enable the domain-wide administration port. In the **RUNNING** state, a server offers its services to clients and can operate as a full member of a cluster. In the **ADMIN** state, the server is up and running, but available only for administration operations allowing you to perform server and application-level administration tasks without risk to running applications. In the **STANDBY** state, a server instance does not process any request; its regular Listen Port is closed but the Administration Port is open. It only accepts life cycle commands that transition the server instance to either the **RUNNING** or **SHUTDOWN** state. Other Administration requests are not accepted. A **STANDBY** server's only purpose is to resume the **RUNNING** state quickly, saving server startup time.

### **StartupTimeout**

This input specifies the timeout value for server start and resume operations. If the server fails to start in the timeout period, it forces a shutdown. A value of **0** means that the server waits indefinitely for the operation to complete.

### **SocketBufferSizeAsChunkSize**

This input specifies whether the server's buffer size for sending or receiving data through a raw socket should be set to 4KB. Otherwise, the server does not impose a limit to the buffer size and defers to the operating system. This option is only useful in some operating systems for improving performance. It should be disabled in most environments.

### **SystemPasswordEncrypted**

This input specifies the password required to access administrative functions on the server. To set this attribute, use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the value. Then set this attribute to the output of the `encrypt()` method. To compare a password that a user enters with the encrypted value of this attribute, go to the same WebLogic server instance used to set and encrypt this attribute and use `weblogic.management.EncryptionHelper.encrypt()` to encrypt the user-supplied password. Then compare the encrypted values.

### **StuckThreadMaxTime**

This input specifies the number of seconds that a thread must be working continuously before the server considers the thread stuck. For example, if you set this to **600** seconds, the WebLogic server considers a thread to be \"stuck\" after 600 seconds of continuous use. In WebLogic Server 9.x and later, we recommend that you use the `ServerFailureTriggerMBean` in the `OverloadProtectionMBean`. The `ServerFailureTriggerMBean` transitions the server to a **FAILED** state after the specified number of stuck threads are detected. The `OverloadProtectionMBean` has options to suspend or shutdown a failed server.

### **StuckThreadTimerInterval**

This input specifies the number of seconds after which the WebLogic server periodically scans threads to see if they have been working continuously for the configured maximum length of time.

### **TGIOPEnabled**

This input specifies whether the server supports Tuxedo GIOP (TGIOp) requests. This requires you to configure WebLogic Tuxedo Connector (WTC) for the server.

### **ThreadPoolPercentSocketReaders**

This input specifies the percentage of execute threads from the default queue that can be used as socket readers.

### **TimedOutRefIsolationTime**

This input specifies the amount of time in milli seconds that a reference should not be used after a request has timed out. The clusterable ref avoids using this remote ref for the period specified.

### **TransactionLogFilePrefix**

This input specifies the path prefix for the server's JTA transaction log files. If the pathname is not absolute, the path is assumed to be relative to the server's root directory. For a clustered server, if you plan to migrate the Transaction Recovery Service from this server if it fails to another server (backup server) in the same cluster, you must store transaction log files on persistent storage, such as a Storage Area Network (SAN) device or a dual-ported disk, available to both servers. Do not use an NFS file system to store transaction log files. Because of the NFS caching scheme, transaction log files on disk may not always be current. Using transaction log files stored on an NFS device for recovery may cause data corruption.

## TransactionLogFileWritePolicy

This input specifies the policy that determines how transaction log file entries are written to disk; this policy can affect transaction performance. (**Note:** To be transactionally safe, the Direct-Write policy may require additional OS or environment changes on some Windows systems.) The WebLogic server supports the following policies:

- Cache-Flush. Flushes the operating system and on-disk caches after each write.
- Direct-Write. Tells the operating system to write directly to disk with each write. Direct-Write performs better than Cache-Flush and is available on Windows, HP-UX, and Solaris.

If Direct-Write is not supported on the host platform, the policy becomes Cache-Flush and a log message is printed. **Note:** On Windows, unlike Solaris and HP, the "\"Direct-Write\"" policy may leave transaction data in the on-disk cache without writing it to disk immediately. This is not transactionally safe as a power failure can cause loss of on-disk cache data. For transactionally safe writes in Windows using "\"Direct-Write\"", disable all write caching for the disk (enabled by default), or use a disk with a battery-backed cache.

### To disable the on-disk cache for a hard drive in Windows

1. Open the Control-Panel and double-click **System**.
2. In the **System Properties** dialog box, click the **Hardware** tab and then click **Device Manager**.
3. Click the plus sign next to **Disk drives**, and then double-click the name of the hard drive.
4. In the **Properties** dialog box, click the **Policies** tab and then select **Enable write caching on the disk**.

Some file systems, such as a RAID system that has a reliable cache, do not allow this value to be changed.

## TunnelingClientPingSecs

This input specifies the interval (in seconds) at which the server pings a tunneled client to see if it is still alive. If you create network channels for this server, each channel can override this setting.

## TunnelingClientTimeoutSecs

This input specifies the amount of time (in seconds) after which a missing tunneled client is considered dead. If you create network channels for this server, each channel can override this setting.

## TunnelingEnabled

This input specifies whether tunneling for the T3, T3S, HTTP, HTTPS, IIOP, and IIOPS protocols should be enabled for this server. If you create network channels for this server, each channel can override this setting.

## Type

This input returns the type of the MBean.

## UploadDirectoryName

This input specifies the directory path on the Administration server where all uploaded applications are placed. If you do not specify an absolute directory name, the path is relative to rootdirectory/. The default staging directory is "\"stage\"", relative to the server root. On the Managed server this returns null, and is not configurable.

## Use81StyleExecuteQueues

This input specifies the backward compatibility mode to switch to in WebLogic 8.1 to execute queues instead of WorkManagers. Each of the WorkManagers is converted to an individual execute queue. Setting this attribute requires a server restart.

## VerboseEJBDeploymentEnabled

This input specifies whether or not verbose deployment of EJBs is enabled.

## **WeblogicPluginEnabled**

This input specifies whether the server uses the proprietary WL-Proxy-Client-IP header. This is recommended if the server instance receives requests from a proxy plug-in. If the server instance is a member of a cluster that receives proxied requests, enable the WebLogic plugin at the cluster level. For servers that are members of a cluster, the setting at the cluster level overrides the server's setting. When the WebLogic plugin is enabled, a call to `getRemoteAddr` returns the address of the browser client from the proprietary WL-Proxy-Client-IP header instead of the Web server.

## **Query Server Runtime input list**

Following are additional inputs for the *Query Server Runtime* operation.

### **activationTime**

This input specifies how long the server has been running.

### **adminPort**

This input specifies the port on which the Administration Console is listening.

### **adminPortEnabled**

This input specifies the whether the Administration port is enabled.

### **adminServer**

This input specifies the address of the Administration server.

### **adminServerHost**

This input specifies the host that is running the Administration server.

### **adminServerPort**

This input specifies the port on which the adminServer is listening.

### **adminServerPortSecure**

This input specifies whether the Administration server's listen port uses a secure protocol.

### **adminURL**

This input specifies the URL of the Administration Console.

### **classpath**

This input specifies the Java classpath on the server.

### **clusterMaster**

This input specifies whether or not this host is the cluster master.

### **currentDirectory**

This input specifies the path on the host from which the server was started.

### **currentMachine**

This input specifies the machine that is running the server.

### **currentOpenSockets**

This input specifies the current number of open sockets.

### **defaultURL**

This input specifies the URL of the default network listener.

### **listenAddress**

This input specifies the address of the default network listener.

### **restartCount**

This input specifies the total number of restarts since the server was last started.

**restartRequired**

This input specifies whether the server has had configuration changes made to it that require a restart.

**shuttingDown**

This input specifies if the server currently shutting down.

**sslAddress**

This input specifies the address from which SSL is listening.

**sslPort**

This input specifies the port on which SSL is listening.

**sslPortEnabled**

This input specifies that the sslPort is enabled.

**startupTime**

This input specifies the time the server started.

**state**

This input specifies the state of the server.

**totalOpenSockets**

This input specifies the total number of open sockets.

**Type**

This input specifies the type of the server.

**Version**

This input specifies the version of WebLogic running on the server.

## Query Domain Configuration result list

Following are results for the *Query Domain Configuration* operation.

**AdminServerName**

This return specifies The name of the Administration server.

**AdministrationMBeanAuditingEnabled**

This return specifies whether the Administration server generates a log message when the WebLogic server domain's configuration is modified. Any change to a server, module, or other item in the domain (either through the Administration Console, command-line utilities, or the APIs) causes the Administration server to generate this informational message. This attribute has been deprecated in favor of **ConfigurationAuditType**. If values for both attributes are specified, the resultant behavior is the logical OR condition of the two settings.

**AdministrationPort**

This return specifies the common secure administration port for the WebLogic server domain. (This requires you to enable the Administration port.)

**AdministrationPortEnabled**

This return specifies whether the domain-wide Administration port should be enabled for the WebLogic server domain. Because the Administration port uses SSL, enabling it requires that SSL be configured for all servers in the domain. The domain-wide Administration port enables you to start a WebLogic server instance in **STANDBY** state. It also allows you to separate administration traffic from application traffic in your domain. Because all servers in the domain must enable or disable the Administration port at the same time, you configure the default Administration port settings at the domain level.

If you enable the Administration port:

- The Administration port accepts only connections that specify Administrator credentials.
- Connections that specify Administrator credentials can only use the Administration port.
- The command that starts Managed servers must specify a secure protocol and the administration port:

```
Dweblogic.management.server=https://admin_server:administration_port
```

### **AdministrationProtocol**

This return specifies the default protocol for communicating through the Administration port or Administration channels. (This requires you to enable the Administration port or to create an Administration channel.) If requests through the Administration port or an Administration channel do not specify a protocol, the WebLogic server uses the protocol specified here.

### **ArchiveConfigurationCount**

This return specifies the number of archival versions of config.xml that are saved by the Administration server each time the domain configuration is modified.

### **AutoDeployForSubmodulesEnabled**

This return indicates whether autodeployed applications should include JMS modules. If **true**, then any submodules defined in the application's JMS modules are deployed with default targets. The submodules define the different destinations in the JMS module, such as topics and queues. If you do not provide them with explicit targets, they may not be properly deployed.

### **ClusterConstraintsEnabled**

This return specifies that deployments targeted to a cluster succeed only if all servers in the cluster are running. By default, cluster constraints are disabled and deployment is attempted only on the servers that are reachable at the time of deployment from the Administration server. Any servers that have been shut down or are temporarily partitioned from the Administration server retrieve the deployment during server startup or shortly after the network partition is resolved.

### **ConfigBackupEnabled**

If **true**, this return specifies that backups of the configuration are made during server boot.

### **ConfigurationAuditType**

This returns the criteria used for auditing configuration events (configuration changes and other operations). The options are:

- **CONFIG\_CHANGE\_NONE** causes configuration events not to be written to the server log or directed to the Security Audit Framework.
- **CONFIG\_CHANGE\_LOG** causes configuration events to be written to the server log.
- **CONFIG\_CHANGE\_AUDIT** causes configuration events to be directed to the Security Audit Framework.
- **CONFIG\_CHANGE\_LOG\_AND\_AUDIT** causes configuration events to be written to the server log and directed to the Security Audit Framework.

### **ConfigurationVersion**

This return specifies the release identifier for the configuration. This identifier is used to indicate the version of the configuration. All server-generated configurations are established with the release identifier of the running server. The form of the version is major.minor.servicepack.rollingpatch. Not all parts of the version are required, for instance "7" is acceptable.

### **ConsoleContextPath**

This return specifies the context path to use in URLs that specify the Administration Console. (This requires you to enable the Administration Console for the current domain.) To access the Administration Console, use the following URL: <http://listen-address:listen-port/context-path>. For example, if you set the context path to `myconsole`, use the following URL to access

the Administration Console: <http://localhost:7001/myconsole>. To specify the listen address and listen port that you use to access the Administration Console, configure the listen address and listen port of the Administration server.

### **ConsoleEnabled**

This return specifies whether the Administration server automatically deploys the Administration Console in the current domain. If the Administration Console is not deployed, you can still use the WebLogic Scripting Tool or the management APIs to configure and monitor the domain.

### **ConsoleExtensionDirectory**

This returns the directory path from which console extensions are loaded.

### **DomainVersion**

This return defines the common version of all servers in a domain. In a domain containing servers that are not all at the same release version, this attribute is used to determine the feature level that servers assume. The value must be less than or equal to the version of any Managed server in the domain. If this value is not equal to the release version of the Administration server, the server cannot make modifications to the configuration.

### **LastModificationTime**

This returns the last time that the domain was updated. This is unique for a given transactional modification.

### **Name**

This return specifies the user-specified name of this MBean instance. This name is included as one of the key properties in the MBean's `javax.management.ObjectName`: `Name=user-specified-name`.

### **Notes**

This return specifies optional information that you can include to describe this configuration. The WebLogic server saves this note in the domain's configuration file (`config.xml`) as XML PCDATA. All left angle brackets (`&lt;`) are converted to the XML entity `&lt;`. Carriage returns and line feeds are preserved. **Note:** If you create or edit a note from the Administration Console, the carriage returns and line feeds are not preserved.

### **ProductionModeEnabled**

This return specifies whether all servers in this domain run in production mode. You can configure servers in your domain to start in one of two modes—development or production. Use development mode while you are developing your applications. Development mode uses a relaxed security configuration and enables you to auto-deploy applications. Use production mode when your application is running in its final form. A production domain uses full security and may use clusters or other advanced features. The runtime mode is a domain-wide setting. As each Managed server starts, it refers to the mode of the Administration server to determine its runtime mode. If you configure the domain to run in production mode, the Administration server saves this setting to the domain's configuration document.

### **RootDirectory**

This returns the root directory for the domain by a server process [`ServerMBean.getRootDirectory`] or [`ServerMBean.getDomainDirectory`].

### **Type**

This returns the type of the MBean.