

Deployit Weblogic Plugin Manual

Version 3.6.1

Table of Content

Preface	4
Overview	4
Features	4
Requirements	4
Usage in Deployment Packages	4
Using the deployables and deployed	5
Deployable vs. Container table	5
Deployed Actions Table	5
Deploying applications	6
Note about the version	7
Note about targeting to multiple containers	7
Creating resources	7
Note about managing JMS resources	7
Extension points	8
Extending the Plugin (A Tutorial)	9
Discovery	11
CI Reference	11
Configuration Item Overview	11
Deployable Configuration Items	11
Deployed Configuration Items	11
Topology Configuration Items	12
Virtual Deployable Configuration Items	12
Virtual Deployed Configuration Items	12
Virtual Topology Configuration Items	13
Configuration Item Details	13
wls.AbstractQueue	13
wls.AbstractUniformDistributedQueue	14
wls.Cluster	16
wls.ConnectionFactory	17
wls.ConnectionFactorySpec	18
wls.DataSource	19
wls.DataSourceSpec	21
wls.Domain	22
wls.Ear	23
wls.EarModule	24
wls.EjbJar	27
wls.EjbJarModule	27
wls.ExtensibleDeployedArtifact	30
wls.FilePersistentStore	32
wls.FilePersistentStoreSpec	33
wls.JmsDestination	33
wls.JmsResource	35
wls.JmsResourceSpec	36
wls.JmsServer	37
wls.JmsTarget	37
wls.MailSession	37
wls.MailSessionSpec	38
wls.PersistentStore	39
wls.PersistentStoreSpec	40
wls.Queue	40
wls.QueueSpec	42
wls.Resource	43
wls.ResourceSpec	44
wls.Server	44
wls.SharedLibraryWar	45
wls.SharedLibraryWarModule	46
wls.UniformDistributedQueue	49

wls.UniformDistributedQueueSpec	50
wls.War	51
wls.WarModule	52
wls.WlsContainer	55

Preface

This document describes the functionality provided by the WebLogic server (WLS) plugin.

See the **Deployit Reference Manual** for background information on Deployit and deployment concepts.

Overview

The WLS plugin is a Deployit plugin that adds capability for managing deployments and resources on WebLogic server. It works out of the box for deploying/ undeploying application artifacts, datasource and other JMS resources (see the *Features* section below) , and can easily be extended to support more deployment options or management of new artifacts/resources on WLS.

Features

- Deployment units
 - Enterprise application (EAR)
 - Web application (WAR)
 - Enterprise JavaBean (EJB)
 - J2EE Shared library
- Staging modes
 - Stage
 - Nostage
- Deployment strategies
 - Classical
 - Versioned
 - Side by side
- Resources
 - Datasource
 - JMS Queue
 - JMS uniform distributed Queue
 - JMS connection factory
 - Mail Session
 - Persistence Store (file)
- Discovery

Requirements

- **Deployit requirements**
 - **Deployit:** version 3.6+
 - **WLS versions:** WLS 9.x, WLS 10.3, WLS 11g (Unix and Windows)
 - **Other Deployit Plugins:** None
- **Infrastructural requirements**
 - **WebLogic Domain user credentials**
 - **User credentials** for accessing the Host managing the WebLogic Administration Server.
 - **User credentials** for accessing target Hosts of managed Servers (for NoStage mode)

Usage in Deployment Packages

The plugin works with the standard deployment package of DAR format. Please see the *Packaging Manual* for more details about the DAR format and the ways to compose one.

The following is a sample MANIFEST.MF file that can be used to create a WebLogic specific deployment package. It

contain declarations for an [Ear](#), a [datasource](#) and a couple of JMS resources.

Manifest-Version: 1.0
Deployit-Package-Format-Version: 1.3
CI-Application: SampleApp
CI-Version: 1.0
Name: SampleApp-1.0.ear
CI-Name: SampleApp
CI-Type: jee.Ear
Name: testDatasource
CI-Type: wls.DataSourceSpec
CI-jndiNames: jdbc/sampleDatasource
CI-url: jdbc:mysql://localhost/test
CI-driverName: com.mysql.jdbc.Driver
CI-username: {{DATABASE_USERNAME}}
CI-password: {{DATABASE_PASSWORD}}
Name: sampleQueue
CI-Type: wls.QueueSpec
CI-jndiNames: jms/testQueue
CI-jmsModuleName: {{JMS_MODULE_NAME}}
Name: sampleCf
CI-Type: wls.ConnectionFactorySpec
CI-jndiNames: jms/sampleCf
CI-jmsModuleName: {{JMS_MODULE_NAME}}

Using the deployables and deployed

The following table describes which deployable/container combinations are possible.

Deployable vs. Container table

Deployable	Container	Generated deployed
Application artifact: jee.Ear jee.War wls.EjbJar	wls.Cluster wls.Server	wls.EarModule wls.WarModule wls.EjbJarModule
wls.SharedLibraryWar	wls.Cluster wls.Server	wls.SharedLibraryWarModule
wls.DataSourceSpec	wls.Cluster wls.Server	wls.DataSource
wls.QueueSpec	wls.JmsServer	wls.Queue
wls.UniformDistributedQueueSpec	wls.Cluster wls.Server wls.JmsServer	wls.UniformDistributedQueue
wls.ConnectionFactorySpec	wls.Cluster wls.Server wls.JmsServer	wls.ConnectionFactory
wls.MailSessionSpec	wls.Cluster wls.Server	wls.MailSession

The following table describes the effect a deployed has on it's container

Deployed Actions Table

Deployed	Actions performed for operations
----------	----------------------------------

	Create	Destroy	Modify
wls.EarModule wls.WarModule wls.EjbJarModule	<ul style="list-style-type: none"> uploaded artifact to host deploy application start application 	<ul style="list-style-type: none"> stop application undeploy application delete artifact from host 	<ul style="list-style-type: none"> stop application undeploy application delete artifact from host uploaded new artifact to host deploy application start application
wls.SharedLibraryWarModule	<ul style="list-style-type: none"> upload library to host deploy library 	<ul style="list-style-type: none"> undeploy library delete library from host 	<ul style="list-style-type: none"> undeploy library delete library from host upload library to host deploy library
wls.DataSource	<ul style="list-style-type: none"> create datasource 	<ul style="list-style-type: none"> destroy datasource 	<ul style="list-style-type: none"> destroy datasource create new datasource
wls.Queue	<ul style="list-style-type: none"> create queue 	<ul style="list-style-type: none"> destroy queue 	<ul style="list-style-type: none"> modify queue (if modify-script specified in synthetic.xml) <p>OR</p> <ul style="list-style-type: none"> destroy queue create new queue
wls.UniformDistributedQueue	<ul style="list-style-type: none"> create UDD queue 	<ul style="list-style-type: none"> destroy UDD queue 	<ul style="list-style-type: none"> modify UDD queue (if modify-script specified in synthetic.xml) <p>OR</p> <ul style="list-style-type: none"> destroy UDD queue create new UDD queue
wls.ConnectionFactory	<ul style="list-style-type: none"> create connection factory 	<ul style="list-style-type: none"> destroy connection factory 	<ul style="list-style-type: none"> destroy connection factory create new connection factory
wls.MailSession	<ul style="list-style-type: none"> create mail session 	<ul style="list-style-type: none"> destroy mail session 	<ul style="list-style-type: none"> destroy mail session create mail session
wls.FilePersistentStore	<ul style="list-style-type: none"> create file persistence store 	<ul style="list-style-type: none"> destroy file persistence store 	<ul style="list-style-type: none"> destroy file persistence store create file persistence store

Deploying applications

The way an application is deployed to a container can be influenced by modifying properties of the corresponding deployed.

For example, if an Ear is to be deployed as a *versioned* application, using *nostage* mode, specify these properties in the deployed [EarModule](#):

- `versioned` = `true` (or on UI, check the checkbox)
- `stage mode` = `NoStage` (or on UI, choose `NoStage` from the dropdown)
- `staging directory` = absolute path of the directory where Ear is to be uploaded

Similarly, if the deployed application needs to be upgraded using the *side by side* deployment strategy, modify these properties in the deployed [EarModule](#):

- `staging directory` = new path of the directory where the new version of Ear will be uploaded
- `deployment strategy` = `SIDE_BY_SIDE` (or on the UI, choose `SIDE_BY_SIDE` from the dropdown)
- `retire timeout` = Time interval (seconds) if a timeout period is needed for retiring the previous version of the application

Note about the version

The plugin allows to deploy non versioned artifacts (ear, war, ejbjar) as a versioned artifact. In this case, the plugin computes automatically the version using this pattern: Application-VersionOfThePackage. Of course, if you artifact is packaged with a version (for example Shared Library), the version will be read from the manifest file.

Note about targeting to multiple containers

Certain deployables can be targeted to multiple containers. For example, an Ear can be targeted to two clusters. Similarly a datasource can be targeted to two clusters.

Note that the way WLS plugin handles this multiple targeting is by generating steps for each targeting. So for example, if a datasource is targeted to two cluster (say Cluster-1, Cluster-2) , Deployit will create two datasource creation steps, wherein.

- The first step will create the datasource on Cluster-1, with all the properties of the deployed datasource.
- The second step will add Cluster-2 to the target list of the datasource created in first step. If there are difference in the datasource values of this deployed, it **overrides** the previous value.

Since the second targeting overrides the properties of the first targeting, take utmost care to keep the properties of the deployments (of the same deployable) uniform across each other.

Similar to creation, the following sequence of steps occurs if destroy operation takes place for such a multiple targeted datasource:

- The first step will remove Cluster-1 from datasource target's list
- The second step will remove Cluster-2 from datasource target's list, and since the datasource has no target set on it, it destroys the datasource.

Note that the actual datasource destruction takes place in the second step, and the first step simply removes the first container from datasource targets.

Creating resources

Deployit handles the creation of resources in the same way it handles deploying an application. Refer to *Reference Manual* for more details on deploying resources.

Note about managing JMS resources

The WLS plugin greatly simplifies the management of JMS resources. It does this by automatically managing the JMS modules and sub-deployments needed for JMS resources, letting the user to focus on the actual JMS resource he needs to manage. For example, the followings is the sequence of steps that happens behind the scene when a JMS resource like Queue is created:

- the JMS module name is specified by user in deployed resource (look at 'jmsModuleName' property in [Queue](#) for example)
- plugin automatically creates the *module* if it is not present, otherwise adds the deployed container to existing module targets
- plugin automatically creates a *subdeployment* if is not present, otherwise adds the deployed container to existing subdeployment targets
- plugin creates/updates the JMS resource and assign the subdeployment created in previous step as the resource subdeployment

Similarly, the destruction of a JMS resource is handled behind the scene in the following way:

- the resource container is removed from it's subdeployment targets.
- destroys the JMS resource only if it's subdeployment targets list is empty (if it's the last one)

- destroys the subdeployment automatically if it contains no targets
- destroy JMS module *if no other JMS resources are using it*.

The thing to note is that the WLS plugin manage modules intelligently unless you want to use your own.

Extension points

The WLS plugin is designed to be extended through Deployit's Plugin API type system and through the use of custom user defined WLST Python scripts. Refer to *Customization Manual* for an explanation of the type system.

The WLS plugin associates Create, Modify and Destroy operations received from Deployit with WLST Python scripts that need to be executed for the operation. The operation specific script is given a Python object representation of the Deployed that triggered the operation. The script is then executed using WLST on the target Domain. Below, for example is the definition of `wls.DataSource` in `synthetic.xml`:

```
<type type="wls.DataSource" extends="wls.Resource" deployable-type="wls.DataSourceSpec" description="An object bound to the
    JNDI tree that provides database connectivity through a pool of JDBC connections">
  <generate-deployable type="wls.DataSourceSpec" extends="wls.ResourceSpec" description="Specification for a datasource"/>
  <property name="additionalPropertiesNotToExpose" hidden="true" default="jndiNames, url, driverName, username,
    password, properties"/>

  <property name="createScript" default="wls/ds/create-datasource.py" hidden="true" />

  <property name="destroyScript" default="wls/ds/destroy-datasource.py" hidden="true" />

  <property name="jndiNames" description="JNDI path to where this data source is bound" />
  <property name="url" description="URL of the database to connect to." />
  <property name="driverName" description="Full package name of JDBC driver class used to create the physical
    database connections in the connection pool" />
  <property name="username" description="Username attribute passed to the JDBC driver when creating
    physical database connections" />
  <property name="password" password="true" description="Password attribute passed to the
    JDBC driver when creating physical database connections" />
</type>
```

The script has all the information from the Deployed at its disposal to translate into the WLST API calls needed to configure WebLogic. The following sample Python snippet is using deployed to create a datasource:

```
cmo.createJDBCSystemResource(deployed.name)
datasourcePath = '/JDBCSystemResources/%s/JDBCResource/%s' % (deployed.name, deployed.name)
cd(datasourcePath)
cd('%s/JBCDriverParams/%s' % (datasourcePath, deployed.name))
set("Url", deployed.url)
set("DriverName", deployed.driverName)
set('Password', deployed.password)
# use jmsModuleName, jmsServer and jndiName to create the queue
```

The WLS plugin also offers the ability to influence the order in which scripts are executed in relation to other Deployed operations. The order allows for the chaining of scripts to create a logical sequence of events. For example, the following `synthetic.xml` snippet says that creation of the queue (order = 60) will happen before deployment of the Ear (order = 70), and the destruction of the queue (order = 40) will take place the after undeployment of the Ear (order = 30)

```
<type type="wls.EarModule" extends="wls.ExtensibleDeployedArtifact" deployable-type="jee.Ear" description="Ear
    with values configured for a deployment">
  <generate-deployable type="wls.Ear" extends="jee.Ear" description="A JEE EAR archive"/>
  <property name="createScript" default="wls/application/deploy-application.py" hidden="true"/>
  <property name="createVerb" default="Deploy" hidden="true" />
  <property name="createOrder" kind="integer" default="70" hidden="true" />

  <property name="destroyScript" default="wls/application/undeploy-application.py" hidden="true"/>
  <property name="destroyVerb" default="Undeploy" hidden="true" />
  <property name="destroyOrder" kind="integer" default="30" hidden="true" />
```



```

<property name="startScript" default="wls/application/start-application.py" hidden="true"/>
<property name="startOrder" kind="integer" default="90" hidden="true" />

<property name="stopScript" default="wls/application/stop-application.py" hidden="true"/>
<property name="stopOrder" kind="integer" default="10" hidden="true" />
</type>

<type type="wls.Queue" extends="wls.AbstractQueue" deployable-type="wls.QueueSpec"
  description="A point-to-point destination type">
  <generate-deployable type="wls.QueueSpec" extends="wls.JmsResourceSpec" description="Specification for a JMS Queue"/>
  <property name="createScript" default="wls/jms/create-queue.py" hidden="true"/>
  <property name="createVerb" default="Create" hidden="true" />
  <property name="createOrder" kind="integer" default="60" hidden="true" />

  <property name="destroyScript" default="wls/jms/destroy-queue.py" hidden="true"/>
  <property name="destroyVerb" default="Destroy" hidden="true" />
  <property name="destroyOrder" kind="integer" default="40" hidden="true" />

  <property name="setErrorDestinationScript" default="wls/jms/set-error-queue.py" hidden="true"/>
  <property name="unsetErrorDestinationScript" default="wls/jms/unset-error-queue.py" hidden="true"/>
</type>

```

Next section describes the extensibility by examples:

Extending the Plugin (A Tutorial)

Hiding an existing property from a deployed/deployable

The following synthetic.xml snippet shows how JDBCConnectionPoolParams_CapacityIncrement property in wls.Datasource can be made hidden giving it a default value of 2.

```

<type-modification type="wls.DataSource">
  <!-- makes the property hidden from the UI -->
  <property name="JDBCConnectionPoolParams_CapacityIncrement" category="Connection Pool" label="Capacity Increment"
    kind="integer" hidden="true" default="2"/>
</type-modification>

```

Adding a new property to a deployed/deployable

The following synthetic.xml snippet shows how a new property inactiveConnectionTimeoutSeconds can be added to wls.Datasource

```

<type-modification type="wls.DataSource">
  <!-- adding new property -->
  <property name="JDBCConnectionPoolParams_InactiveConnectionTimeoutSeconds" category="Connection Pool"
    label="Inactive Connection Timeout (sec)" kind="integer" description="inactive Connection Timeout in Seconds" />
</type-modification>

```

Note that while adding a new property in WLS plugin, the *property name should correspond to the relative path of the property (file) from the configuration item in WLST (minus the type name)*. For example, since the relative path of property InactiveConnectionTimeoutSeconds in WLST is {datasource-name}/JDBCConnectionPoolParams/{datasource-name}/InactiveConnectionTimeoutSeconds, the property name to use while adding a new property is JDBCConnectionPoolParams_InactiveConnectionTimeoutSeconds.

Adding a new type

The following synthetic.xml snippet shows the definition of a new CI type wls.WorkManager. Since it's a resource and it can be targeted to a Cluster or a Server, it has been made to extend wls.Resource

```

<type type="wls.WorkManager" extends="wls.Resource" deployable-type="wls.WorkManagerSpec">
  <generate-deployable type="wls.WorkManagerSpec" extends="wls.ResourceSpec"/>

```

```
<property name="createScript" default="wls/env/create-work-manager.py" hidden="true" />
<property name="destroyScript" default="wls/env/destroy-work-manager.py" hidden="true" />
</type>
```

The name property is automatically added to all CIs so it has not been defined explicitly as a property. Additional properties can be added in the definition as per the need.

Next step involves adding the Python scripts for the steps. For the `wls.WorkManager` example, two Python scripts needs to be created: create-work-manager.py and destroy-work-manager.py

wls/env/create-work-manager.py

```
workManagerPath='/SelfTuning/%s/WorkManagers/%s' %(deployed.container.domain.name, deployed.name)
connectAndEdit()

if exists(workManagerPath):
    print 'Modifying work manager %s for target %s' %(deployed.name, deployed.container.name)
    setOrOverride = overrideWithWarning
else:
    print 'Creating work manager %s for target %s' %(deployed.name, deployed.container.name)
    cd('/SelfTuning/' + deployed.container.domain.name + '/WorkManagers')
    cmo.createWorkManager(deployed.name)
    setOrOverride = set

cd(workManagerPath)
newTargets = []
for t in get('Targets'):
    newTargets.append(t)

newTargets.append(ObjectName(deployed.container.objectName))
set('Targets', jarray.array(newTargets, ObjectName))

saveAndExit()
```

wls/env/destroy-work-manager.py

```
workManagerPath='/SelfTuning/%s/WorkManagers/%s' %(deployed.container.domain.name, deployed.name)
connectAndEdit()

if not exists(workManagerPath):
    print "Work manager with name %s does not exist." %(deployed.name)
    sys.exit(1)

cd(workManagerPath)
currentTargets = get('Targets')
print 'oldTargets: %s' %(currentTargets)
containerTarget = ObjectName(deployed.container.objectName)
newTargets = []
for t in currentTargets:
    if t != containerTarget:
        newTargets.append(t)

print 'new targets: %s' %(newTargets)
if len(newTargets) > 0:
    print 'Modifying work manager %s' %(deployed.name)
    set('Targets', jarray.array(newTargets, ObjectName))
else:
    print 'Deleting workmanager %s' %(deployed.name)
    cd('../')
    delete(deployed.name, 'WorkManagers')

saveAndExit()
```

Note 1: In the above example Python files, functions 'connectAndEdit()', 'saveAndExit()' are utility functions defined in the base.py file in WLS plugin. Have a look at the base.py file to see other utility functions.

Discovery

Once the admin server's Host and Domain are specified, the following containers can be discovered by the WLS plugin:

- [Cluster](#)
- [Server](#)
- [JMSServer](#)

Here is an example CLI script which discovers a sample WLS domain:

```
adminServerHost = repository.create(factory.configurationItem('Infrastructure/adminServerHost', 'overthere.SshHost',
    {'os':'UNIX', 'connectionType':'SFTP', 'address':'wls-103', 'username':'demo-user', 'password':'demo-password'}))

wlsDomain = factory.configurationItem('Infrastructure/demoWlsDomain', 'wls.Domain',
    {'wlHome':'/opt/bea-10.3/wlserver_10.3', 'domainHome':'/opt/bea-10.3/user_projects/domains/demoWlsDomain',
    'port':'7001', 'username':'weblogic', 'password':'weblogic', 'adminServerName':'adminServer',
    'startMode':'NodeManager', 'host':'Infrastructure/adminServerHost'})

discoveredItems = deployit.discover(wlsDomain)
print discoveredItems

#discovery just discovers the topology and keeps the configuration items in memory. Save them in Deployit repository
repository.create(discoveredItems)
```

Limitations

- The WLS topology discovery doesn't discover/associate the Host associated with the managed [Servers](#). So if a [Cluster](#) is spanned on multiple Hosts, the creation of the managed server's Host and it's association with the [Server](#) is a manual process. This can be done using the CLI or more easily, using the UI. This may be needed for certain deployment scenarios where knowledge of the [Server's](#) Host is needed (like NoStage deployments) .

CI Reference

Configuration Item Overview

Deployable Configuration Items

CI	Description
wls.ConnectionFactorySpec	Specification for a JMS connection factory
wls.DataSourceSpec	Specification for a datasource
wls.Ear	A JEE EAR archive
wls.EjbJar	A JEE EJB archive
wls.FilePersistentStoreSpec	Description unavailable (deployable)
wls.MailSessionSpec	Specification for a mail session
wls.PersistentStoreSpec	Description unavailable (deployable)
wls.QueueSpec	Specification for a JMS Queue
wls.SharedLibraryWar	A JEE library archive
wls.UniformDistributedQueueSpec	Specification for a JMS uniform distributed queue
wls.War	A JEE WAR archive

Deployed Configuration Items

CI	Description
wls.ConnectionFactory	A connection factory defines a set of connection configuration parameters that are used to create connections for JMS clients
wls.DataSource	An object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections
wls.EarModule	Ear with values configured for a deployment
wls.EjbJarModule	EJB with values configured for a deployment
wls.FilePersistentStore	Description unavailable
wls.MailSession	Mail sessions facilitate the process of using the JavaMail APIs, which provide applications and other J2EE modules with access to Internet Message Access Protocol (IMAP)- and Simple Mail Transfer Protocol (SMTP)-capable mail servers on your network or the Internet
wls.Queue	A JMS Queue Defines a point-to-point destination type, which are used for asynchronous peer communications
wls.SharedLibraryWarModule	The Java EE library feature provides an easy way to share one or more types of Java EE modules among multiple Enterprise Applications
wls.UniformDistributedQueue	A distributed Queue defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS clients
wls.WarModule	War with values configured for a deployment

Topology Configuration Items

CI	Description
wls.Cluster	WebLogic Cluster which defines groups of WebLogic servers that work together to increase scalability and reliability
wls.Domain	WebLogic Domain which is a collection of WebLogic Server instances that is managed by a single Administration Server
wls.JmsServer	WebLogic JMS server, that act as management containers for the queues and topics in JMS modules that are targeted to them
wls.Server	WebLogic Server that runs in its own Java Virtual Machine (JVM) and has its own configuration

Virtual Deployable Configuration Items

CI	Description
wls.JmsResourceSpec	Base deployable type for all JMS related resources
wls.ResourceSpec	Base deployable of all Resources

Virtual Deployed Configuration Items

CI	Description
wls.AbstractQueue	Base class for all JMS destinations, which can have a error destination property defined on them
wls.AbstractUniformDistributedQueue	Base class for all JMS destinations, which can have a error destination property defined on them
wls.ExtensibleDeployedArtifact	Base class for all deployeds meant to contain Applications
wls.JmsDestination	Base class for all JMS destinations, which can have a error destination property defined on them
wls.JmsResource	Base deployed type for all JMS related resources
wls.PersistentStore	Description unavailable
wls.Resource	Base deployed of all Resources

Virtual Topology Configuration Items

CI	Description
wls.JmsTarget	
wls.WlsContainer	


Configuration Item Details

[wls.AbstractQueue](#)

Hierarchy [wls.JmsDestination](#) >> [wls.JmsResource](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

Base class for all JMS destinations, which can have a error destination property defined on them

Public Properties	
 container : CI<udm.Container>	
The container on which this deployed runs.	
jmsModuleName : STRING	
Existing or new Jms system module which will be used to hold this resource	
jndiName : STRING	
Global JNDI name used to look up the destination within the JNDI namespace	
deployable : CI<udm.Deployable>	
The deployable that this deployed is derived from.	
errorDestination : CI<wls.JmsDestination>	
Target error destination for messages that have expired or reached their redelivery limit	

Hidden Properties

createOrder : INTEGER = 60

Create Order

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

setErrorDestinationOrder : INTEGER = 61

Set Error Destination Order

setErrorDestinationVerb : STRING = *Set error queue for*

Set Error Destination Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName, errorDestination, setErrorDestinationScript, setErrorDestinationVerb, setErrorDestinationOrder, unsetErrorDestinationVerb, unsetErrorDestinationOrder, unsetErrorDestinationScript*

Standard Properties Not To Expose

unsetErrorDestinationOrder : INTEGER = 37

Unset Error Destination Order

unsetErrorDestinationVerb : STRING = *Unset error queue from =*

Unset Error Destination Verb

createScript : STRING

Python script invoked to create this resource

destroyScript : STRING

Python script invoked to destroy this resource

modifyScript : STRING

Python script invoked to upgrade this resource

setErrorDestinationScript : STRING

Python script invoked to set error destination on this jms resource

unsetErrorDestinationScript : STRING

Python script invoked to unset error destination from this jms resource

wls.AbstractUniformDistributedQueue

Hierarchy [wls.JmsDestination](#) >> [wls.JmsResource](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

Interfaces [udm.Deployed](#), [udm.ConfigurationItem](#)

Base class for all JMS destinations, which can have a error destination property defined on them

Public Properties



container : [CI<udm.Container>](#)

The container on which this deployed runs.

jmsModuleName : [STRING](#)

Existing or new Jms system module which will be used to hold this resource

jndiName : [STRING](#)

Global JNDI name used to look up the destination within the JNDI namespace

deployable : [CI<udm.Deployable>](#)

The deployable that this deployed is derived from.

errorDestination : [CI<wls.JmsDestination>](#)

Target error destination for messages that have expired or reached their redelivery limit

Hidden Properties

createOrder : INTEGER = 60

Create Order

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

setErrorDestinationOrder : INTEGER = 61

Set Error Destination Order

setErrorDestinationVerb : STRING = *Set error queue for*

Set Error Destination Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName, errorDestination, setErrorDestinationScript, setErrorDestinationVerb, setErrorDestinationOrder, unsetErrorDestinationVerb, unsetErrorDestinationOrder, unsetErrorDestinationScript*

Standard Properties Not To Expose

unsetErrorDestinationOrder : INTEGER = 37

Unset Error Destination Order

unsetErrorDestinationVerb : STRING = *Unset error queue from =*

Unset Error Destination Verb

createScript : STRING

Python script invoked to create this resource

destroyScript : STRING

Python script invoked to destroy this resource

modifyScript : STRING

Python script invoked to upgrade this resource

setErrorDestinationScript : STRING

Python script invoked to set error destination on this jms resource

unsetErrorDestinationScript : STRING

Python script invoked to unset error destination from this jms resource


wls.Cluster

Hierarchy `udm.BaseContainer >> udm.BaseConfigurationItem`

Interfaces `udm.Taggable`, [wls.WlsContainer](#), `python.PythonManagedContainer`, [wls.JmsTarget](#), `udm.ConfigurationItem`, `udm.Container`

WebLogic Cluster which defines groups of WebLogic servers that work together to increase scalability and reliability

Public Properties

 domain : <code>CI<wls.Domain></code>
The domain to which the WebLogic Cluster belongs. 'asContainment'=true, means a Cluster is 'contained' under a Domain
servers : <code>SET_OF_CI<wls.Server></code>
Servers in the WebLogic Cluster
tags : <code>SET_OF_STRING</code>
The tags to map deployables to containers.


wls.ConnectionFactory

Hierarchy [wls.JmsResource](#) >> `python.PythonManagedDeployed` >> `udm.BaseDeployed` >> `udm.BaseConfigurationItem`

Interfaces `udm.Deployed`, `udm.ConfigurationItem`

A connection factory defines a set of connection configuration parameters that are used to create connections for JMS clients. Connection factories can configure properties of the connections returned to the JMS client, and also provide configurable options for default delivery, transaction, and message flow control parameters

Public Properties

 container : <code>CI<udm.Container></code>
The container on which this deployed runs.
jmsModuleName : <code>STRING</code>
Existing or new Jms system module which will be used to hold this resource
jndiName : <code>STRING</code>
Global JNDI name used to look up the destination within the JNDI namespace
LoadBalancingParams_ServerAffinityEnabled : <code>BOOLEAN</code>
ServerAffinityEnabled
TransactionParams_XAConnectionFactoryEnabled : <code>BOOLEAN</code>
XAConnectionFactoryEnabled
deployable : <code>CI<udm.Deployable></code>
The deployable that this deployed is derived from.

Hidden Properties**createOrder** : INTEGER = 60

Create Order

createScript : STRING = *wls/jms/create-connection-factory.py*

Create Script

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyScript : STRING = *wls/jms/destroy-connection-factory.py*

Destroy Script

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName*

Standard Properties Not To Expose

modifyScript : STRING

Python script invoked to upgrade this resource

wls.ConnectionFactorySpec**Hierarchy** *wls.JmsResourceSpec* >> *udm.BaseDeployable* >> *udm.BaseConfigurationItem***Interfaces** *udm.Taggable*, *udm.Deployable*, *udm.ConfigurationItem*

Specification for a JMS connection factory

Public Properties**LoadBalancingParams_ServerAffinityEnabled** : **STRING**

ServerAffinityEnabled

TransactionParams_XAConnectionFactoryEnabled : **STRING**

XAConnectionFactoryEnabled

jmsModuleName : **STRING**

Existing or new Jms system module which will be used to hold this resource

jndiName : **STRING**

Global JNDI name used to look up the destination within the JNDI namespace

tags : **SET_OF_STRING**

The tags to map deployables to containers.

wls.DataSource**Hierarchy** [wls.Resource](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem**Interfaces** udm.Deployed, udm.ConfigurationItem

An object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections

Public Properties

 **container** : `CI<udm.Container>`

The container on which this deployed runs.

driverName : `STRING`

Full package name of JDBC driver class used to create the physical database connections in the connection pool

jndiNames : `STRING`

JNDI path to where this data source is bound

password : `STRING`

Password attribute passed to the JDBC driver when creating physical database connections

url : `STRING`

URL of the database to connect to.

username : `STRING`

Username attribute passed to the JDBC driver when creating physical database connections

JDBCConnectionPoolParams_CapacityIncrement : `INTEGER`

Number of connections created when new connections are added to the connection pool

JDBCConnectionPoolParams_InitialCapacity : `INTEGER`

Number of physical connections to create when creating the connection pool

JDBCConnectionPoolParams_MaxCapacity : `INTEGER`

Maximum number of physical connections that this connection pool can contain

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

properties : `MAP_STRING_STRING`

The map of properties passed to the JDBC driver that are used to create physical database connections

Hidden Properties**additionalPropertiesNotToExpose** : *STRING = jndiNames, url, driverName, username, password, properties*

Additional Properties Not To Expose

createOrder : *INTEGER = 60*

Create Order

createScript : *STRING = wls/ds/create-datasource.py*

Create Script

createVerb : *STRING = Create*

Create Verb

destroyOrder : *INTEGER = 40*

Destroy Order

destroyScript : *STRING = wls/ds/destroy-datasource.py*

Destroy Script

destroyVerb : *STRING = Destroy*

Destroy Verb

modifyOrder : *INTEGER = 40*

Modify Order

modifyVerb : *STRING = Upgrade*

Modify Verb

standardPropertiesNotToExpose : *STRING = id, name, type, _properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName*

Standard Properties Not To Expose

modifyScript : *STRING*

Python script invoked to upgrade this resource

wls.DataSourceSpec**Hierarchy** [wls.ResourceSpec](#) >> [udm.BaseDeployable](#) >> [udm.BaseConfigurationItem](#)**Interfaces** [udm.Tagable](#), [udm.Deployable](#), [udm.ConfigurationItem](#)

Specification for a datasource

Public Properties**JDBCConnectionPoolParams_CapacityIncrement** : **STRING**

Number of connections created when new connections are added to the connection pool

JDBCConnectionPoolParams_InitialCapacity : **STRING**

Number of physical connections to create when creating the connection pool

JDBCConnectionPoolParams_MaxCapacity : **STRING**

Maximum number of physical connections that this connection pool can contain

driverName : **STRING**

Full package name of JDBC driver class used to create the physical database connections in the connection pool

jndiNames : **STRING**

JNDI path to where this data source is bound

password : **STRING**

Password attribute passed to the JDBC driver when creating physical database connections

properties : **MAP_STRING_STRING**

The map of properties passed to the JDBC driver that are used to create physical database connections

tags : **SET_OF_STRING**

The tags to map deployables to containers.

url : **STRING**

URL of the database to connect to.

username : **STRING**

Username attribute passed to the JDBC driver when creating physical database connections

wls.Domain**Hierarchy** udm.BaseContainer >> udm.BaseConfigurationItem**Interfaces** udm.Taggable, python.PythonManagingContainer, python.PythonManagedContainer, udm.ConfigurationItem, udm.Container

WebLogic Domain which is a collection of WebLogic Server instances that is managed by a single Administration Server

Public Properties

adminServerName : *STRING* = *AdminServer*

The name of the admin server

 **clusters** : *SET_OF_CI*<*wls.Cluster*>

WebLogic clusters belonging to domain

host : *CI*<*overthere.Host*>

The host that runs the admin server

password : *STRING*

Password which is used to login to the WebLogic Domain.

port : *INTEGER* = *7001*

Port to be used by the AdminServer for this domain

startMode : *ENUM* [*NodeManager*, *Script*, *WindowsService*] = *NodeManager*

Tells how a managed server is start and stop, default is NodeManager, others are Script or Windows Service

username : *STRING*

Username which is used to login to the WebLogic Domain.

version : *ENUM* [*WEBLOGIC_10*, *WEBLOGIC_11*] = *WEBLOGIC_10*

Version of Oracle WebLogic Server

wlHome : *STRING*

The location of the WebLogic Server installation

domainHome : *STRING*

The location of the WebLogic domain. Defaults to './user_projects/domains/'

tags : *SET_OF_STRING*

The tags to map deployables to containers.

Hidden Properties

runWithDaemon : *BOOLEAN* = *true*

Set to true to execute commands with the Python daemon

wls.Ear

Hierarchy jee.Ear >> udm.BaseDeployableArchiveArtifact >> udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A JEE EAR archive

Public Properties

deploymentOrder : *STRING*

By default, new applications and modules are configured with a Deployment Order value of 100

placeholders : *SET_OF_STRING*

Placeholders detected in this artifact

redeploymentStrategy : *STRING*

indicates what redeployment strategy to use for upgrading the application

retireTimeout : *STRING*

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

scanPlaceholders : *BOOLEAN = true*

Scan Placeholders

stageMode : *STRING*

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : *STRING*

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

tags : *SET_OF_STRING*

The tags to map deployables to containers.

versionIdentifier : *STRING*

Version Identifier

versioned : *STRING*

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

textFileNamesRegex : *STRING = .+\. (cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

wls.EarModule

Hierarchy [wls.ExtensibleDeployedArtifact](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

Interfaces [udm.Artifact](#), [udm.Deployed](#), [udm.ConfigurationItem](#), [udm.DerivedArtifact](#)

Ear with values configured for a deployment

Public Properties



container : `CI<udm.Container>`

The container on which this deployed runs.

deploymentOrder : `INTEGER = 100`

By default, new applications and modules are configured with a Deployment Order value of 100

redeploymentStrategy : `ENUM [CLASSIC, STOP_START, SIDE_BY_SIDE] = CLASSIC`

indicates what redeployment strategy to use for upgrading the application

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

placeholders : `MAP_STRING_STRING`

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

retireTimeout : `INTEGER = -1`

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

stageMode : `ENUM [Stage, NoStage] = Stage`

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : `STRING`

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

versionIdentifier : `STRING`

Version Identifier

versioned : `BOOLEAN`

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

createOrder : INTEGER = 70

Create Order

createScript : STRING = *wls/application/deploy-application.py*

Create Script

createVerb : STRING = *Deploy*

Create Verb

destroyOrder : INTEGER = 30

Destroy Order

destroyScript : STRING = *wls/application/undeploy-application.py*

Destroy Script

destroyVerb : STRING = *Undeploy*

Destroy Verb

modifyOrder : INTEGER = 60

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, deployable, properties, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, startScript, startVerb, startOrder, stopScript, stopVerb, stopOrder, deploymentStrategy, placeholders, file, redeploymentStrategy, stopRetiredApplicationOrder, undeployRetiredApplicationOrder*

Standard Properties Not To Expose

startOrder : INTEGER = 90

Start Order

startScript : STRING = *wls/application/start-application.py*

Start Script

startVerb : STRING = *Start*

Start Verb

stopOrder : INTEGER = 10

Stop Order

stopRetiredApplicationOrder : INTEGER = 95

Stop Retired Application Order

stopScript : STRING = *wls/application/stop-application.py*

Stop Script

stopVerb : STRING = *Stop*

Stop Verb

undeployRetiredApplicationOrder : INTEGER = 98

Undeploy Retired Application Order

wlstPath : STRING = *AppDeployments*

Wlst Path

modifyScript : **STRING**

Python script invoked to upgrade this Java EE artifact

wls.EjbJar

Hierarchy jee.EjbJar >> udm.BaseDeployableArchiveArtifact >> udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A JEE EJB archive

Public Properties

deploymentOrder : **STRING**

By default, new applications and modules are configured with a Deployment Order value of 100

placeholders : **SET_OF_STRING**

Placeholders detected in this artifact

redeploymentStrategy : **STRING**

indicates what redeployment strategy to use for upgrading the application

retireTimeout : **STRING**

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

scanPlaceholders : **BOOLEAN** = *true*

Scan Placeholders

stageMode : **STRING**

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : **STRING**

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

tags : **SET_OF_STRING**

The tags to map deployables to containers.

versionIdentifier : **STRING**

Version Identifier

versioned : **STRING**

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

textFileNamesRegex : **STRING** = *.\.(\.cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

wls.EjbJarModule

Hierarchy [wls.ExtensibleDeployedArtifact](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

Interfaces [udm.Artifact](#), [udm.Deployed](#), [udm.ConfigurationItem](#), [udm.DerivedArtifact](#)

EJB with values configured for a deployment

Public Properties



container : [CI<udm.Container>](#)

The container on which this deployed runs.

deploymentOrder : [INTEGER](#) = *100*

By default, new applications and modules are configured with a Deployment Order value of 100

redeploymentStrategy : [ENUM \[CLASSIC, STOP_START, SIDE_BY_SIDE\]](#) = *CLASSIC*

indicates what redeployment strategy to use for upgrading the application

deployable : [CI<udm.Deployable>](#)

The deployable that this deployed is derived from.

placeholders : [MAP_STRING_STRING](#)

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

retireTimeout : [INTEGER](#) = *-1*

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

stageMode : [ENUM \[Stage, NoStage\]](#) = *Stage*

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : [STRING](#)

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

versionIdentifier : [STRING](#)

Version Identifier

versioned : [BOOLEAN](#)

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

createOrder : INTEGER = 70

Create Order

createScript : STRING = *wls/application/deploy-application.py*

Create Script

createVerb : STRING = *Deploy*

Create Verb

destroyOrder : INTEGER = 30

Destroy Order

destroyScript : STRING = *wls/application/undeploy-application.py*

Destroy Script

destroyVerb : STRING = *Undeploy*

Destroy Verb

modifyOrder : INTEGER = 60

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, deployable, properties, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, startScript, startVerb, startOrder, stopScript, stopVerb, stopOrder, deploymentStrategy, placeholders, file, redeploymentStrategy, stopRetiredApplicationOrder, undeployRetiredApplicationOrder*

Standard Properties Not To Expose

startOrder : INTEGER = 90

Start Order

startScript : STRING = *wls/application/start-application.py*

Start Script

startVerb : STRING = *Start*

Start Verb

stopOrder : INTEGER = 10

Stop Order

stopRetiredApplicationOrder : INTEGER = 95

Stop Retired Application Order

stopScript : STRING = *wls/application/stop-application.py*

Stop Script

stopVerb : STRING = *Stop*

Stop Verb

undeployRetiredApplicationOrder : INTEGER = 98

Undeploy Retired Application Order

wlstPath : STRING = *AppDeployments*

Wlst Path

modifyScript : *STRING*

Python script invoked to upgrade this Java EE artifact


wls.ExtensibleDeployedArtifact

Hierarchy python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Artifact, udm.Deployed, udm.ConfigurationItem, udm.DerivedArtifact

Base class for all deployments meant to contain Applications

Public Properties

 **container** : *CI<udm.Container>*

The container on which this deployed runs.

deploymentOrder : *INTEGER = 100*

By default, new applications and modules are configured with a Deployment Order value of 100

redeploymentStrategy : *ENUM [CLASSIC, STOP_START, SIDE_BY_SIDE] = CLASSIC*

indicates what redeployment strategy to use for upgrading the application

deployable : *CI<udm.Deployable>*

The deployable that this deployed is derived from.

placeholders : *MAP_STRING_STRING*

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

retireTimeout : *INTEGER = -1*

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

stageMode : *ENUM [Stage, NoStage] = Stage*

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : *STRING*

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

versionIdentifier : *STRING*

Version Identifier

versioned : *BOOLEAN*

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

createOrder : INTEGER = 70

Create Order

createScript : STRING = *wls/application/deploy-application.py*

Create Script

createVerb : STRING = *Deploy*

Create Verb

destroyOrder : INTEGER = 30

Destroy Order

destroyScript : STRING = *wls/application/undeploy-application.py*

Destroy Script

destroyVerb : STRING = *Undeploy*

Destroy Verb

modifyOrder : INTEGER = 60

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, deployable, properties, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, startScript, startVerb, startOrder, stopScript, stopVerb, stopOrder, deploymentStrategy, placeholders, file, redeploymentStrategy, stopRetiredApplicationOrder, undeployRetiredApplicationOrder*

Standard Properties Not To Expose

startOrder : INTEGER = 90

Start Order

startScript : STRING = *wls/application/start-application.py*

Start Script

startVerb : STRING = *Start*

Start Verb

stopOrder : INTEGER = 10

Stop Order

stopRetiredApplicationOrder : INTEGER = 95

Stop Retired Application Order

stopScript : STRING = *wls/application/stop-application.py*

Stop Script

stopVerb : STRING = *Stop*

Stop Verb

undeployRetiredApplicationOrder : INTEGER = 98

Undeploy Retired Application Order

wlstPath : STRING = *AppDeployments*

Wlst Path

modifyScript : *STRING*

Python script invoked to upgrade this Java EE artifact

wls.FilePersistentStore

Hierarchy [wls.PersistentStore](#) >> [wls.Resource](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)

Interfaces [udm.Deployed](#), [udm.ConfigurationItem](#)

Description unavailable

Public Properties



container : *CI<udm.Container>*

The container on which this deployed runs.

directory : *STRING*

Directory

synchronousWritePolicy : *STRING* = *Cache-Flush*

Synchronous Write Policy

deployable : *CI<udm.Deployable>*

The deployable that this deployed is derived from.

Hidden Properties**createOrder** : INTEGER = 60

Create Order

createScript : STRING = *wls/resources/create-file-persistence-store.py*

Create Script

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyScript : STRING = *wls/resources/destroy-file-persistence-store.py*

Destroy Script

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, _properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName*

Standard Properties Not To Expose

modifyScript : STRING

Python script invoked to upgrade this resource

wls.FilePersistentStoreSpec**Hierarchy** [wls.PersistentStoreSpec](#) >> [wls.ResourceSpec](#) >> [udm.BaseDeployable](#) >> [udm.BaseConfigurationItem](#)**Interfaces** [udm.Taggable](#), [udm.Deployable](#), [udm.ConfigurationItem](#)

Description unavailable (deployable)

Public Properties**directory** : STRING

Directory

synchronousWritePolicy : STRING

Synchronous Write Policy

tags : SET_OF_STRING

The tags to map deployables to containers.

wls.JmsDestination**Hierarchy** [wls.JmsResource](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >>

udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

Base class for all JMS destinations, which can have a error destination property defined on them

Public Properties



container : [CI<udm.Container>](#)

The container on which this deployed runs.

jmsModuleName : [STRING](#)

Existing or new Jms system module which will be used to hold this resource

jndiName : [STRING](#)

Global JNDI name used to look up the destination within the JNDI namespace

deployable : [CI<udm.Deployable>](#)

The deployable that this deployed is derived from.

errorDestination : [CI<wls.JmsDestination>](#)

Target error destination for messages that have expired or reached their redelivery limit

Hidden Properties

createOrder : INTEGER = 60

Create Order

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

setErrorDestinationOrder : INTEGER = 61

Set Error Destination Order

setErrorDestinationVerb : STRING = *Set error queue for*

Set Error Destination Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName, errorDestination, setErrorDestinationScript, setErrorDestinationVerb, setErrorDestinationOrder, unsetErrorDestinationVerb, unsetErrorDestinationOrder, unsetErrorDestinationScript*

Standard Properties Not To Expose

unsetErrorDestinationOrder : INTEGER = 37

Unset Error Destination Order

unsetErrorDestinationVerb : STRING = *Unset error queue from =*

Unset Error Destination Verb

createScript : STRING

Python script invoked to create this resource

destroyScript : STRING

Python script invoked to destroy this resource

modifyScript : STRING

Python script invoked to upgrade this resource

setErrorDestinationScript : STRING

Python script invoked to set error destination on this jms resource

unsetErrorDestinationScript : STRING

Python script invoked to unset error destination from this jms resource

wls.JmsResource

Hierarchy python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

Base deployed type for all JMS related resources

Public Properties



container : *CI<udm.Container>*

The container on which this deployed runs.

jmsModuleName : *STRING*

Existing or new Jms system module which will be used to hold this resource

jndiName : *STRING*

Global JNDI name used to look up the destination within the JNDI namespace

deployable : *CI<udm.Deployable>*

The deployable that this deployed is derived from.

Hidden Properties

createOrder : *INTEGER = 60*

Create Order

createVerb : *STRING = Create*

Create Verb

destroyOrder : *INTEGER = 40*

Destroy Order

destroyVerb : *STRING = Destroy*

Destroy Verb

modifyOrder : *INTEGER = 40*

Modify Order

modifyVerb : *STRING = Upgrade*

Modify Verb

standardPropertiesNotToExpose : *STRING = id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName*

Standard Properties Not To Expose

createScript : *STRING*

Python script invoked to create this resource

destroyScript : *STRING*

Python script invoked to destroy this resource

modifyScript : *STRING*

Python script invoked to upgrade this resource

wls.JmsResourceSpec

Hierarchy udm.BaseDeployable >> udm.BaseConfigurationItem
Interfaces udm.Taggable, udm.Deployable, udm.ConfigurationItem

Base deployable type for all JMS related resources

Public Properties

tags : SET_OF_STRING


The tags to map deployables to containers.

wls.JmsServer

Hierarchy udm.BaseContainer >> udm.BaseConfigurationItem
Interfaces udm.Taggable, python.PythonManagedContainer, udm.ConfigurationItem, [wls.JmsTarget](#), udm.Container

WebLogic JMS server, that act as management containers for the queues and topics in JMS modules that are targeted to them

Public Properties

 **server** : CI<[wls.Server](#)>

A server instance or migratable target this JMS server is deployed to. 'asContainment'=true, means a JmsServer is 'contained' under a Server

tags : SET_OF_STRING

The tags to map deployables to containers.

wls.JmsTarget

null

wls.MailSession

Hierarchy [wls.Resource](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem
Interfaces udm.Deployed, udm.ConfigurationItem

Mail sessions facilitate the process of using the JavaMail APIs, which provide applications and other J2EE modules with access to Internet Message Access Protocol (IMAP)- and Simple Mail Transfer Protocol (SMTP)-capable mail servers on your network or the Internet

Public Properties

 **container** : `CI<udm.Container>`

The container on which this deployed runs.

jndiName : `STRING`

The JNDI name that modules use to access this mail session

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

javaMailProperties : `MAP_STRING_STRING`

Java Mail Properties

Hidden Properties

additionalPropertiesNotToExpose : `STRING = jndiName,javaMailProperties`

Additional Properties Not To Expose

createOrder : `INTEGER = 60`

Create Order

createScript : `STRING = wls/resources/create-mail-session.py`

Create Script

createVerb : `STRING = Create`

Create Verb

destroyOrder : `INTEGER = 40`

Destroy Order

destroyScript : `STRING = wls/resources/destroy-mail-session.py`

Destroy Script

destroyVerb : `STRING = Destroy`

Destroy Verb

modifyOrder : `INTEGER = 40`

Modify Order

modifyVerb : `STRING = Upgrade`

Modify Verb

standardPropertiesNotToExpose : `STRING = id, name, type, _properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName`

Standard Properties Not To Expose

modifyScript : `STRING`

Python script invoked to upgrade this resource

wls.MailSessionSpec

Hierarchy `wls.ResourceSpec >> udm.BaseDeployable >> udm.BaseConfigurationItem`

Interfaces `udm.Taggable, udm.Deployable, udm.ConfigurationItem`

Specification for a mail session

Public Properties

javaMailProperties : `MAP_STRING_STRING`

Java Mail Properties

jndiName : `STRING`

The JNDI name that modules use to access this mail session


tags : `SET_OF_STRING`

The tags to map deployables to containers.

wls.PersistentStore**Hierarchy** [wls.Resource](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem**Interfaces** udm.Deployed, udm.ConfigurationItem

Description unavailable

Public Properties

 **container** : `CI<udm.Container>`

The container on which this deployed runs.

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

Hidden Properties**createOrder** : INTEGER = 60

Create Order

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, _properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName*

Standard Properties Not To Expose

createScript : STRING

Python script invoked to create this resource

destroyScript : STRING

Python script invoked to destroy this resource

modifyScript : STRING

Python script invoked to upgrade this resource

wls.PersistentStoreSpec**Hierarchy** [wls.ResourceSpec](#) >> [udm.BaseDeployable](#) >> [udm.BaseConfigurationItem](#)**Interfaces** [udm.Taggable](#), [udm.Deployable](#), [udm.ConfigurationItem](#)

Description unavailable (deployable)

Public Properties**tags** : SET_OF_STRING

The tags to map deployables to containers.

wls.Queue**Hierarchy** [wls.AbstractQueue](#) >> [wls.JmsDestination](#) >> [wls.JmsResource](#) >> [python.PythonManagedDeployed](#) >> [udm.BaseDeployed](#) >> [udm.BaseConfigurationItem](#)**Interfaces** [udm.Deployed](#), [udm.ConfigurationItem](#)

A JMS Queue Defines a point-to-point destination type, which are used for asynchronous peer communications. A message delivered to a queue is distributed to only one consumer

Public Properties



container : `CI<udm.Container>`

The container on which this deployed runs.

jmsModuleName : `STRING`

Existing or new Jms system module which will be used to hold this resource

jndiName : `STRING`

Global JNDI name used to look up the destination within the JNDI namespace

DeliveryFailureParams_RedeliveryLimit : `INTEGER = -1`

Number of redelivery tries a message can have before it is moved to the error destination

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

errorDestination : `CI<wls.JmsDestination>`

Target error destination for messages that have expired or reached their redelivery limit

Hidden Properties

createOrder : INTEGER = 60

Create Order

createScript : STRING = *wls/jms/create-queue.py*

Create Script

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyScript : STRING = *wls/jms/destroy-queue.py*

Destroy Script

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

setErrorDestinationOrder : INTEGER = 61

Set Error Destination Order

setErrorDestinationScript : STRING = *wls/jms/set-error-queue.py*

Set Error Destination Script

setErrorDestinationVerb : STRING = *Set error queue for*

Set Error Destination Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName, errorDestination, setErrorDestinationScript, setErrorDestinationVerb, setErrorDestinationOrder, unsetErrorDestinationVerb, unsetErrorDestinationOrder, unsetErrorDestinationScript*

Standard Properties Not To Expose

unsetErrorDestinationOrder : INTEGER = 37

Unset Error Destination Order

unsetErrorDestinationScript : STRING = *wls/jms/unset-error-queue.py*

Unset Error Destination Script

unsetErrorDestinationVerb : STRING = *Unset error queue from =*

Unset Error Destination Verb

modifyScript : STRING

Python script invoked to upgrade this resource

wls.QueueSpec

Hierarchy [wls.JmsResourceSpec](#) >> udm.BaseDeployable >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.Deployable, udm.ConfigurationItem

Specification for a JMS Queue

Public Properties

DeliveryFailureParams_RedeliveryLimit : [STRING](#)

Number of redelivery tries a message can have before it is moved to the error destination

jmsModuleName : [STRING](#)

Existing or new Jms system module which will be used to hold this resource

jndiName : [STRING](#)

Global JNDI name used to look up the destination within the JNDI namespace

tags : [SET_OF_STRING](#)

The tags to map deployables to containers.

wls.Resource

Hierarchy python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

Base deployed of all Resources

Public Properties



container : [CI<udm.Container>](#)

The container on which this deployed runs.

deployable : [CI<udm.Deployable>](#)

The deployable that this deployed is derived from.

Hidden Properties**createOrder** : INTEGER = 60

Create Order

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, _properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName*

Standard Properties Not To Expose

createScript : STRING

Python script invoked to create this resource

destroyScript : STRING

Python script invoked to destroy this resource

modifyScript : STRING

Python script invoked to upgrade this resource

wls.ResourceSpec**Hierarchy** udm.BaseDeployable >> udm.BaseConfigurationItem**Interfaces** udm.Tagable, udm.Deployable, udm.ConfigurationItem

Base deployable of all Resources

Public Properties**tags** : SET_OF_STRING

The tags to map deployables to containers.

wls.Server**Hierarchy** udm.BaseContainer >> udm.BaseConfigurationItem**Interfaces** udm.Tagable, [wls.WlsContainer](#), python.PythonManagedContainer, [wls.JmsTarget](#), udm.ConfigurationItem, udm.Container

WebLogic Server that runs in its own Java Virtual Machine (JVM) and has its own configuration

Public Properties



domain : `CI<wls.Domain>`

WebLogic domain to which this server belongs. 'asContainment'=true, means a Server is 'contained' under a Domain

port : `INTEGER`

Port for the server runs on

host : `CI<overthere.Host>`

Host on which this server is running, needed to perform no-stage deployments and to start the server via a script

startCommand : `STRING`

Command that should be executed to start the managed server.

stopCommand : `STRING`

Command that should be executed to stop the managed server.

tags : `SET_OF_STRING`

The tags to map deployables to containers.

wls.SharedLibraryWar

Hierarchy `jee.War >> udm.BaseDeployableArchiveArtifact >> udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem`

Interfaces `udm.Taggable, udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact`

A JEE library archive

Public Properties**deploymentOrder** : *STRING*

By default, new applications and modules are configured with a Deployment Order value of 100

placeholders : *SET_OF_STRING*

Placeholders detected in this artifact

redeploymentStrategy : *STRING*

indicates what redeployment strategy to use for upgrading the application

retireTimeout : *STRING*

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

scanPlaceholders : *BOOLEAN = true*

Scan Placeholders

stageMode : *STRING*

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : *STRING*

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

tags : *SET_OF_STRING*

The tags to map deployables to containers.

versionIdentifier : *STRING*

Version Identifier

versioned : *STRING*

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties**textFileNamesRegex** : *STRING = .+\. (cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

wls.SharedLibraryWarModule

Hierarchy [wls.ExtensibleDeployedArtifact](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Artifact, udm.Deployed, udm.ConfigurationItem, udm.DerivedArtifact

The Java EE library feature provides an easy way to share one or more types of Java EE modules among multiple Enterprise Applications. In particular, a Java EE library is a stand-alone EJB or Web Application module, multiple EJB or Web Application modules packaged in an Enterprise Application (EAR), or a single plain JAR file that is registered with the Java EE application container upon deployment

Public Properties



container : `CI<udm.Container>`

The container on which this deployed runs.

deploymentOrder : `INTEGER = 100`

By default, new applications and modules are configured with a Deployment Order value of 100

redeploymentStrategy : `ENUM [CLASSIC, STOP_START, SIDE_BY_SIDE] = CLASSIC`

indicates what redeployment strategy to use for upgrading the application

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

placeholders : `MAP_STRING_STRING`

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

retireTimeout : `INTEGER = -1`

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

stageMode : `ENUM [Stage, NoStage] = Stage`

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : `STRING`

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

versionIdentifier : `STRING`

Version Identifier

versioned : `BOOLEAN`

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

createOrder : INTEGER = 70

Create Order

createScript : STRING = *wls/application/deploy-application.py*

Create Script

createVerb : STRING = *Deploy*

Create Verb

destroyOrder : INTEGER = 30

Destroy Order

destroyScript : STRING = *wls/application/undeploy-application.py*

Destroy Script

destroyVerb : STRING = *Undeploy*

Destroy Verb

libraryModule : STRING = *true*

Library Module

modifyOrder : INTEGER = 60

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, deployable, properties, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, startScript, startVerb, startOrder, stopScript, stopVerb, stopOrder, deploymentStrategy, placeholders, file, redeploymentStrategy, stopRetiredApplicationOrder, undeployRetiredApplicationOrder*

Standard Properties Not To Expose

startOrder : INTEGER = 90

Start Order

startScript : STRING = *wls/application/start-application.py*

Start Script

startVerb : STRING = *Start*

Start Verb

stopOrder : INTEGER = 10

Stop Order

stopRetiredApplicationOrder : INTEGER = 95

Stop Retired Application Order

stopScript : STRING = *wls/application/stop-application.py*

Stop Script

stopVerb : STRING = *Stop*

Stop Verb

undeployRetiredApplicationOrder : INTEGER = 98

Undeploy Retired Application Order

wlstPath : *STRING* = *Libraries*

Wlst Path

modifyScript : *STRING*

Python script invoked to upgrade this Java EE artifact

wls.UniformDistributedQueue

Hierarchy [wls.AbstractUniformDistributedQueue](#) >> [wls.JmsDestination](#) >> [wls.JmsResource](#) >>
python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Deployed, udm.ConfigurationItem

A distributed Queue defines a set of queues that are distributed on multiple JMS servers, but which are accessible as a single, logical queue to JMS clients

Public Properties

 **container** : *CI*<*udm.Container*>

The container on which this deployed runs.

jmsModuleName : *STRING*

Existing or new Jms system module which will be used to hold this resource

jndiName : *STRING*

Global JNDI name used to look up the destination within the JNDI namespace

DeliveryFailureParams_RedeliveryLimit : *INTEGER* = *-1*

Number of redelivery tries a message can have before it is moved to the error destination

deployable : *CI*<*udm.Deployable*>

The deployable that this deployed is derived from.

errorDestination : *CI*<*wls.JmsDestination*>

Target error destination for messages that have expired or reached their redelivery limit

Hidden Properties

createOrder : INTEGER = 60

Create Order

createScript : STRING = *wls/jms/create-udd-queue.py*

Create Script

createVerb : STRING = *Create*

Create Verb

destroyOrder : INTEGER = 40

Destroy Order

destroyScript : STRING = *wls/jms/destroy-udd-queue.py*

Destroy Script

destroyVerb : STRING = *Destroy*

Destroy Verb

modifyOrder : INTEGER = 40

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

setErrorDestinationOrder : INTEGER = 61

Set Error Destination Order

setErrorDestinationScript : STRING = *wls/jms/set-udd-error-queue.py*

Set Error Destination Script

setErrorDestinationVerb : STRING = *Set error queue for*

Set Error Destination Verb

standardPropertiesNotToExpose : STRING = *id, name, type, properties, deployable, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, jndiName, jmsModuleName, subDeploymentName, errorDestination, setErrorDestinationScript, setErrorDestinationVerb, setErrorDestinationOrder, unsetErrorDestinationVerb, unsetErrorDestinationOrder, unsetErrorDestinationScript*

Standard Properties Not To Expose

unsetErrorDestinationOrder : INTEGER = 37

Unset Error Destination Order

unsetErrorDestinationScript : STRING = *wls/jms/unset-udd-error-queue.py*

Unset Error Destination Script

unsetErrorDestinationVerb : STRING = *Unset error queue from =*

Unset Error Destination Verb

modifyScript : STRING

Python script invoked to upgrade this resource

wls.UniformDistributedQueueSpec

Hierarchy [wls.JmsResourceSpec](#) >> udm.BaseDeployable >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.Deployable, udm.ConfigurationItem

Specification for a JMS uniform distributed queue

Public Properties

DeliveryFailureParams_RedeliveryLimit : [STRING](#)

Number of redelivery tries a message can have before it is moved to the error destination

jmsModuleName : [STRING](#)

Existing or new Jms system module which will be used to hold this resource

jndiName : [STRING](#)

Global JNDI name used to look up the destination within the JNDI namespace

tags : [SET_OF_STRING](#)

The tags to map deployables to containers.

wls.War

Hierarchy jee.War >> udm.BaseDeployableArchiveArtifact >> udm.BaseDeployableFileArtifact >> udm.BaseDeployableArtifact >> udm.BaseDeployable >> udm.BaseConfigurationItem

Interfaces udm.Taggable, udm.Deployable, udm.SourceArtifact, udm.ArchiveArtifact, udm.Artifact, udm.DeployableArtifact, udm.ConfigurationItem, udm.FileArtifact

A JEE WAR archive

Public Properties**deploymentOrder** : *STRING*

By default, new applications and modules are configured with a Deployment Order value of 100

placeholders : *SET_OF_STRING*

Placeholders detected in this artifact

redeploymentStrategy : *STRING*

indicates what redeployment strategy to use for upgrading the application

retireTimeout : *STRING*

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

scanPlaceholders : *BOOLEAN = true*

Scan Placeholders

stageMode : *STRING*

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : *STRING*

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

tags : *SET_OF_STRING*

The tags to map deployables to containers.

versionIdentifier : *STRING*

Version Identifier

versioned : *STRING*

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties**textFileNamesRegex** : *STRING = .+\. (cfg | conf | config | ini | properties | props | txt | asp | aspx | htm | html | jsf | jsp | xht | xhtml | sql | xml | xsd | xsl | xslt)*

Regular expression that matches file names of text files

wls.WarModule

Hierarchy [wls.ExtensibleDeployedArtifact](#) >> python.PythonManagedDeployed >> udm.BaseDeployed >> udm.BaseConfigurationItem

Interfaces udm.Artifact, udm.Deployed, udm.ConfigurationItem, udm.DerivedArtifact

War with values configured for a deployment

Public Properties



container : `CI<udm.Container>`

The container on which this deployed runs.

deploymentOrder : `INTEGER = 100`

By default, new applications and modules are configured with a Deployment Order value of 100

redeploymentStrategy : `ENUM [CLASSIC, STOP_START, SIDE_BY_SIDE] = CLASSIC`

indicates what redeployment strategy to use for upgrading the application

deployable : `CI<udm.Deployable>`

The deployable that this deployed is derived from.

placeholders : `MAP_STRING_STRING`

A Map containing all the placeholders mapped to their values. Special values are <ignore> or <empty>

retireTimeout : `INTEGER = -1`

timeout interval(in secs) before the previous application version is undeployed for side by side redeployment strategy

stageMode : `ENUM [Stage, NoStage] = Stage`

indicates whether the artifact will be deployed as staged or nostage mode

stagingDirectory : `STRING`

absolute directory path where the artifact will be uploaded and used by the servers for nostage deployment mode

versionIdentifier : `STRING`

Version Identifier

versioned : `BOOLEAN`

indicates wither this artifact is to be deployed as a versioned application

Hidden Properties

createOrder : INTEGER = 70

Create Order

createScript : STRING = *wls/application/deploy-application.py*

Create Script

createVerb : STRING = *Deploy*

Create Verb

destroyOrder : INTEGER = 30

Destroy Order

destroyScript : STRING = *wls/application/undeploy-application.py*

Destroy Script

destroyVerb : STRING = *Undeploy*

Destroy Verb

modifyOrder : INTEGER = 60

Modify Order

modifyVerb : STRING = *Upgrade*

Modify Verb

standardPropertiesNotToExpose : STRING = *id, name, type, deployable, properties, container, createScript, createVerb, createOrder, modifyScript, modifyVerb, modifyOrder, destroyScript, destroyVerb, destroyOrder, startScript, startVerb, startOrder, stopScript, stopVerb, stopOrder, deploymentStrategy, placeholders, file, redeploymentStrategy, stopRetiredApplicationOrder, undeployRetiredApplicationOrder*

Standard Properties Not To Expose

startOrder : INTEGER = 90

Start Order

startScript : STRING = *wls/application/start-application.py*

Start Script

startVerb : STRING = *Start*

Start Verb

stopOrder : INTEGER = 10

Stop Order

stopRetiredApplicationOrder : INTEGER = 95

Stop Retired Application Order

stopScript : STRING = *wls/application/stop-application.py*

Stop Script

stopVerb : STRING = *Stop*

Stop Verb

undeployRetiredApplicationOrder : INTEGER = 98

Undeploy Retired Application Order

wlstPath : STRING = *AppDeployments*

Wlst Path

modifyScript : **STRING**

Python script invoked to upgrade this Java EE artifact

wls.WlsContainer

null
