



# BEAProducts

## **Using the WebLogic Server Scripting Tool for Offline Configuration**

WebLogic Server® Version 9.2  
BEA Workshop for WebLogic Platform™ Version 9.2  
WebLogic Portal® Version 9.2  
BEA AquaLogic Service Bus™ Version 2.5  
Document Revised: June 28, 2006

# Copyright

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved.

## Restricted Rights Legend

This software is protected by copyright, and may be protected by patent laws. No copying or other use of this software is permitted unless you have entered into a license agreement with BEA authorizing such use. This document is protected by copyright and may not be copied photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form, in whole or in part, without prior consent, in writing, from BEA Systems, Inc.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE DOCUMENTATION IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA SYSTEMS DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE DOCUMENT IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

## Trademarks and Service Marks

Copyright © 1995-2006 BEA Systems, Inc. All Rights Reserved. BEA, BEA JRockit, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop, Built on BEA, Jolt, JoltBeans, SteelThread, Top End, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA AquaLogic, BEA AquaLogic Data Services Platform, BEA AquaLogic Enterprise Security, BEA AquaLogic Interaction, BEA AquaLogic Interaction Analytics, BEA AquaLogic Interaction Collaboration, BEA AquaLogic Interaction Content Services, BEA AquaLogic Interaction Data Services, BEA AquaLogic Interaction Integration Services, BEA AquaLogic Interaction Process, BEA AquaLogic Interaction Publisher, BEA AquaLogic Interaction Studio, BEA AquaLogic Service Bus, BEA AquaLogic Service Registry, BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Kodo, BEA Liquid Data for WebLogic, BEA Manager, BEA MessageQ, BEA SALT, BEA Service Architecture Leveraging Tuxedo, BEA WebLogic Commerce Server, BEA WebLogic Communications Platform, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Java Adapter for Mainframe, BEA WebLogic JDriver, BEA WebLogic Log Central, BEA WebLogic Mobility Server, BEA WebLogic Network Gatekeeper, BEA WebLogic Personalization Server, BEA WebLogic Personal Messaging API, BEA WebLogic Platform, BEA WebLogic Portlets for Groupware Integration, BEA WebLogic Real Time, BEA WebLogic RFID Compliance Express, BEA WebLogic RFID Edge Server, BEA WebLogic RFID Enterprise Server, BEA WebLogic Server Process Edition, BEA WebLogic SIP Server, BEA WebLogic WorkGroup Edition, BEA Workshop for WebLogic Platform, BEA Workshop JSP, BEA Workshop JSP Editor, BEA Workshop Struts, BEA Workshop Studio, Dev2Dev, Liquid Computing, and Think Liquid are trademarks of BEA Systems, Inc. Accelerated Knowledge Transfer, AKT, BEA Mission Critical Support, BEA Mission Critical Support Continuum, and BEA SOA Self Assessment are service marks of BEA Systems, Inc.

All other names and marks are property of their respective owners.

# Contents

## Using the WebLogic Server Scripting Tool for Offline Configuration

Introducing WLST .....	1
Getting Started Using WLST Offline Configuration .....	2
Setting up Your Environment—WebLogic Platform 8.1 SP5 and Later .....	3
Setting up Your Environment—WebLogic Platform 8.1 SP4, SP3, or SP2 .....	4
Invoking WLST .....	5
WLST Offline Configuration Variables .....	7
WLST Offline Configuration Commands .....	8
Guidelines for Entering WLST Commands .....	9
Control Commands .....	9
addTemplate .....	10
closeDomain .....	11
closeTemplate .....	11
exit .....	12
readDomain .....	12
readTemplate .....	13
updateDomain .....	13
writeDomain .....	14
Browse Command (cd) .....	14
Edit Commands .....	15
assign .....	16

assignAll . . . . .	17
create . . . . .	18
delete . . . . .	19
get . . . . .	20
loadDB . . . . .	20
set . . . . .	21
setOption . . . . .	22
unassign . . . . .	23
unassignAll . . . . .	25
Information Commands . . . . .	26
dumpStack . . . . .	26
dumpVariables . . . . .	27
help . . . . .	27
ls . . . . .	28
prompt . . . . .	30
pwd . . . . .	31
startRecording . . . . .	31
stopRecording . . . . .	32

# Using the WebLogic Server Scripting Tool for Offline Configuration

This document describes how to use the BEA Products™ Scripting Tool (WLST) for offline configuration. Topics include:

- [Introducing WLST](#)
- [Getting Started Using WLST Offline Configuration](#)
- [WLST Offline Configuration Variables](#)
- [WLST Offline Configuration Commands](#)

## Introducing WLST

WLST is a command-line scripting interface that you use to configure WebLogic Server and WebLogic domains. Using WLST, WebLogic Server administrators and operators can perform administrative tasks and initiate WebLogic Server configuration changes interactively or by using an executable script.

**Note:** The WLST scripting environment is based on the Java scripting interpreter, Jython. For more information about the Jython language syntax, see <http://www.jython.org>.

WLST supports both online and offline configuration:

- *WLST online configuration*

Enables you to perform administrative tasks and initiate WebLogic Server configuration changes while connected to a running server. For more information about WLST online configuration, see *WebLogic Server Scripting Language*, delivered with the WLST online

<~runChNum>

configuration kit, *WebLogic Scripting Tool (WLST) for 7.0 & 8.1* (Artifact ID S13), in the dev2dev code samples project at:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=13>

- *WLST offline configuration*

Enables you to create a new domain or update an existing domain without connecting to a running WebLogic Server—supporting the same functionality as the Configuration Wizard.

WLST offline configuration provides access to persisted configuration information. You can create new configuration information, and retrieve and change existing configuration values that are stored in the domain `config.xml` file or in a domain template JAR created using Template Builder.

This document describes how to use WLST for offline configuration.

Refer to the following resources for additional information:

- For an overview of WLST and how it works, see *WebLogic Server Scripting Language*, delivered with the WLST online configuration kit, *WebLogic Scripting Tool (WLST) for 7.0 & 8.1* (Artifact ID S13), in the dev2dev code samples project at:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=13>

- For more information about the Configuration Wizard and Template Builder, see *Creating WebLogic Configurations Using the Configuration Wizard*, at:

<http://e-docs.bea.com/platform/docs81/configwiz/index.html>

## Getting Started Using WLST Offline Configuration

The following sections describe how to get started using WLST offline configuration. The following topics are described:

- [Setting up Your Environment—WebLogic Platform 8.1 SP5 and Later](#)
- [Setting up Your Environment—WebLogic Platform 8.1 SP4, SP3, or SP2](#)
- [Invoking WLST](#)

**Note:** For information about getting started using WLST online configuration, see “Basic Steps for Using WLST” in *WebLogic Server Scripting Language*, delivered with the WLST online configuration kit, *WebLogic Scripting Tool (WLST) for 7.0 & 8.1* (Artifact ID S13), in the dev2dev code samples project at:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=13>

## Setting up Your Environment—WebLogic Platform 8.1 SP5 and Later

For WebLogic Platform 8.1 SP5 (or a later release of 8.1), to set up your environment for WLST offline configuration, perform the following steps:

1. Install the WebLogic Platform 8.1 SP5 (or later release of 8.1) software, as described in the *WebLogic Platform Installation Guide* at:

<http://e-docs.bea.com/platform/docs81/install/index.html>

**Note:** To take full advantage of the domain and extension templates available, a full WebLogic Platform installation is recommended. At a minimum, you need to install the WebLogic Server 8.1 SP5 software, as described in the *WebLogic Server and WebLogic Express Installation Guide* at:

<http://e-docs.bea.com/wls/docs81/install/index.html>

2. Download the WLST offline configuration kit (if you have not done so already), *Configuring WebLogic Server with WLST Offline* (Artifact ID S97), in the dev2dev code samples project at:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

The WLST offline configuration kit contains the following files:

- JAR files (`config.jar`, `comdev.jar`, and `3rdparty.jar`) required to execute the WLST offline configuration tool.

**Note:** These files are included as part of the WebLogic Platform 8.1 SP5 (or later 8.1 release) software installation. They are not required to be extracted from the ZIP file.

- Script files for invoking WLST offline configuration (`runWLSTOffline.cmd` and `runWLSTOffline.sh`).
- Sample script files which are summarized in “[Summary of Sample Scripts](#)” on page 7.
- `README.txt` file summarizing the steps required to get started using the tool.
- This document in PDF format (`wlst_offline.pdf`).

3. Extract the following files from the WLST offline configuration kit:

- `runWLSTOffline.cmd` and `runWLSTOffline.sh` script files to `WL_HOME\common\bin`.
- (Optional) Sample script files to the desired location.

<~runChNum>

4. Update the CLASSPATH environment variable to include the following WebLogic Server, Jython, and WLST files and directories:

```
WL_HOME\server\lib
WL_HOME\server\lib\weblogic.jar
WL_HOME\common\lib\jython.jar
WL_HOME\common\lib\config.jar
WL_HOME\common\lib\comdev.jar
WL_HOME\common\lib\3rdparty.jar
```

## Setting up Your Environment—WebLogic Platform 8.1 SP4, SP3, or SP2

For WebLogic Platform 8.1 SP4, SP3, or SP2, to set up your environment for WLST offline configuration, perform the following steps:

1. Install the WebLogic Platform 8.1 SP4, SP3, or SP2 software, as described in the *WebLogic Platform Installation Guide* at:

<http://e-docs.bea.com/platform/docs81/install/index.html>

**Note:** To take full advantage of the domain and extension templates available, a full WebLogic Platform installation is recommended. At a minimum, you need to install the WebLogic Server 8.1 SPx software, as described in the *WebLogic Server and WebLogic Express Installation Guide* at:

<http://e-docs.bea.com/wls/docs81/install/index.html>

2. Download and install Jython. For more information, see

<http://www.jython.org/download.html>.

**Note:** Download the WLST offline configuration kit (if you have not done so already), *Configuring WebLogic Server with WLST Offline* (Artifact ID S97), in the dev2dev code samples project at:

<https://codesamples.projects.dev2dev.bea.com/servlets/Scarab?id=97>

The WLST offline configuration kit contains the following files:

- JAR files (config.jar, comdev.jar, and 3rdparty.jar) required to execute the WLST offline configuration tool.
- Script files for invoking WLST offline configuration (runWLSToffline.cmd and runWLSToffline.sh).
- Sample script files which are summarized in “Summary of Sample Scripts” on page 7.



- `README.txt` file summarizing the steps required to get started using the tool.
  - This document in PDF format (`wlst_offline.pdf`).
3. Extract the following files from the WLST offline configuration kit:
- WLST JAR files, including `config.jar`, `comdev.jar`, and `3rdparty.jar`, to `WL_HOME\common\lib`
- Note:** It is recommended that you back up the existing JAR files.
- `runWLSTOffline.cmd` and `runWLSTOffline.sh` script files to `WL_HOME\common\bin`.
  - (Optional) Sample script files to the desired location.
4. Update the `CLASSPATH` environment variable to include the following WebLogic Server, Jython, and WLST files and directories:

**Note:** `JYTHON_HOME` refers to the root directory of your Jython installation.

```
WL_HOME\server\lib
WL_HOME\server\lib\weblogic.jar
JYTHON_HOME\jython.jar
WL_HOME\common\lib\config.jar
WL_HOME\common\lib\comdev.jar
WL_HOME\common\lib\3rdparty.jar
```

## Invoking WLST

You can invoke WLST in interactive or script mode:

- *Interactive mode*

WLST interprets commands interactively, supplied one-at-a-time from a command prompt. Interactive mode enables you to:

- Navigate and interrogate configuration objects in a domain.
- Verify configuration changes as you execute them.
- Verify and test command syntax.
- Record user interactions to a script.

You can also execute a script from this mode.

- *Script mode*

<~runChNum>

WLST interprets commands from a script file. Script mode enables you to:

- Invoke a sequence of WLST offline configuration commands without requiring manual input, much like a shell script.
- More conveniently implement loops, flow control, conditional statements, and variables.
- Automate configuration changes.

The methods for invoking WLST in each mode are described in the following sections. A summary of the sample scripts available is also provided in [“Summary of Sample Scripts” on page 7](#).

## Invoking WLST in Interactive Mode

To invoke WLST offline configuration in interactive mode, at a command prompt enter one of the following commands:

Windows: `WL_HOME\common\bin\runWLSToffline.cmd`

Unix: `WL_HOME/common/bin/runWLSToffline.sh`

WLST starts in interactive mode, and a welcome message and the WLST offline configuration prompt appears:

```
Welcome to Weblogic Scripting Shell
wls_offline:/>
```

**Note:** The first time the command is run, the Jython package manager processes the JAR files in your `CLASSPATH`.

You can begin entering commands, setting variables, or running a script to create a new or update an existing domain.

To run a WLST script in interactive mode, enter the `execfile` command, as follows:

```
execfile(WLST_script)
```

*WLST\_script* specifies the full pathname to the WLST script file. For a summary of the sample script files available, see [“Summary of Sample Scripts” on page 7](#).

For more information about the WLST commands and variables, see [“WLST Offline Configuration Commands” on page 8](#) and [“WLST Offline Configuration Variables” on page 7](#), respectively.

## Invoking WLST in Script Mode

To invoke WLST offline configuration in script mode, at a command prompt enter one of the following commands:

Windows: `WL_HOME\common\bin\runWLSToffline.cmd WLST_script`

Unix: `WL_HOME/common/bin/runWLSToffline.sh WLST_script`

The `WLST_script` argument specifies the full pathname to a WLST script file. For a summary of the sample script files available, see [“Summary of Sample Scripts” on page 7](#).

WLST starts in script mode, executing the specified script without requiring your input.

## Summary of Sample Scripts

[Table 1](#) summarizes the WLST offline configuration sample scripts that you can run or use as templates for creating additional scripts.

**Table 1 WLST Offline Configuration Example Scripts**

Script	Description
<code>domain_platform.py</code>	Creates a WebLogic domain using the Basic WebLogic Platform Domain template. The script demonstrates the basic steps required to create a domain from a domain template.
<code>domain_WLS.py</code>	Creates a WebLogic domain using the Basic WebLogic Server Domain template. The script demonstrates how to open a domain template, create and edit configuration objects, and write the domain configuration information to the specified directory.
<code>domain_extension.py</code>	Creates a WebLogic domain using the Basic WebLogic Server Domain template and extends it to support WebLogic Workshop. The script demonstrates how to extend a domain using an extension template.
<code>wliClusterDomain.py</code>	Creates a WebLogic domain using the Basic WebLogic Integration Domain template. The script demonstrates how to create, configure, and assign servers to a cluster.

## WLST Offline Configuration Variables

[Table 2](#) describes WLST offline configuration variables and their common usage. All variables are initialized to default values at the start of a user session and are changed according to the user interaction with WLST.

**Note:** While in interactive mode, you can display help information for the WLST offline configuration variables using the `help('variables')` command. For more information, see [“help” on page 27](#).

**Table 2 WLST Offline Configuration Variables**

Variable	Description	Default
<code>cmo</code>	<p>Specifies the Current Management Object. This variable is set to the configuration object to which you have navigated.</p> <p>You use this variable to perform any <code>get</code> or <code>set</code> method on the current configuration object. For example:</p> <pre>cmo.setPassword('weblogic')</pre> <p>Alternatively, you can use the WLST <code>get</code> and <code>set</code> commands, as described in <a href="#">“Edit Commands” on page 15</a>.</p>	Initialized to the root of all domain configuration objects.
<code>exitonerror</code>	Specifies whether to terminate script execution when WLST encounters an exception. This variable is not applicable when running WLST in interactive mode.	True
<code>isrecording</code>	Specifies whether user interactions are being saved in a file. This variable should not be set explicitly; it is set to True and False by the <code>startRecording()</code> and <code>stopRecording()</code> commands, respectively.	False

## WLST Offline Configuration Commands

WLST offline configuration commands (primitives) are executable functions that are extensions to the Jython language and that follow the Jython syntax.

The following sections describe WLST offline configuration commands and how to use them. These commands are available when using WLST for offline configuration in interactive and script mode.

WLST offline configuration commands are divided into the following four categories:

- [Control Commands](#)—operate on a domain or template.
- [Browse Command \(cd\)](#)—navigate the hierarchy of configuration objects.
- [Edit Commands](#)—interrogate and edit configuration objects.

- **Information Commands**—interrogate domains, servers, and variables, and provide configuration object and WLST-related information.

The commands in each category are described in the following sections. Before proceeding, it is recommended that you review [“Guidelines for Entering WLST Commands” on page 9](#).

## Guidelines for Entering WLST Commands

Please review the following guidelines for entering WLST commands:

- Command names and arguments are case sensitive.
- Arguments must be enclosed in single or double quotes. For example, `'newServer'`.
- Configuration object pathnames must be specified using the forward slash character (/) as the delimiter on both Windows and Unix, according to the Jython syntax. To specify a configuration object that includes a forward slash (/) in its name, include the configuration object name in parentheses. For example:

```
cd('JMSQueue/(jms/REGISTRATION_MDB_QUEUE)')
```

- You can only create and access security information when you are creating a new domain using a domain template using WLST. When you are updating a domain, you cannot access security information using WLST.
- While in interactive mode, you can display help information for the WLST offline configuration commands using the `help()` command. For more information, see [“help” on page 27](#).

## Control Commands

Similar to the Configuration Wizard, the control commands for WLST offline configuration enable you to perform the following tasks:

- Create a new domain from a domain template
- Update or extend an existing domain

[Table 3](#) lists the control commands for WLST offline configuration used to create a new or update an existing domain, or to exit WLST.

**Table 3 Control Commands for WLST Offline Configuration**

In order to...	Use this command...	To...	For more information, see...
Create a new domain from a domain template	<a href="#">readTemplate</a>	Open an existing domain template for domain creation.	<a href="#">“readTemplate” on page 13</a>
	<a href="#">writeDomain</a>	Write the domain configuration information to the specified directory.	<a href="#">“writeDomain” on page 14</a>
	<a href="#">closeTemplate</a>	Close the current domain template.	<a href="#">“closeTemplate” on page 11</a>
Update an existing domain	<a href="#">readDomain</a>	Open an existing domain for updating.	<a href="#">“readDomain” on page 12</a>
	<a href="#">addTemplate</a>	Extend the current domain using the specified application or service extension template.	<a href="#">“addTemplate” on page 10</a>
	<a href="#">updateDomain</a>	Update and save the current domain.	<a href="#">“updateDomain” on page 13</a>
	<a href="#">closeDomain</a>	Close the current domain.	<a href="#">“closeDomain” on page 11</a>
Exit WLST	<a href="#">exit</a>	Disconnect WLST from the interactive session and close the scripting shell.	<a href="#">“exit” on page 12</a>

Each command is described in detail in the following sections. Commands are described alphabetically.

## addTemplate

Extends the current domain using the specified application or service extension template.

### Syntax

```
addTemplate(templateFileName)
```

Argument	Definition
<i>templateFileName</i>	The name of the application or service extension template.

## Examples

The following command extends the current domain using the specified extension template:

```
wls_offline:/> readDomain('c:/bea/user_projects/domains/wlw')
wls_offline:/wlw> addTemplate('c:/bea/weblogic81/common/templates/
applications/wlp.jar')
```

## closeDomain

Closes the current domain. The domain is no longer available for editing once it is closed.

### Syntax

```
closeDomain()
```

## Examples

The following command closes the current domain:

```
wls_offline:/> readDomain('c:/bea/user_projects/domains/medrec')
...
wls_offline:/medrec> updateDomain()
wls_offline:/medrec> closeDomain()
```

## closeTemplate

Closes the current domain template. The domain template is no longer available once it is closed.

### Syntax

```
closeTemplate()
```

<~runChNum>

## Examples

The following command closes the current domain template:

```
wls_offline:/> readTemplate('c:/bea/weblogic81/common/templates/domains/
medrec.jar')
...
wls_offline:/medrec> writeDomain('c:/bea/user_projects/domains/medrec')
wls_offline:/medrec> closeTemplate()
```

## exit

Disconnects WLST from the user session and closes the scripting shell.

### Syntax

```
exit()
```

### Example

The following example instructs WLST to disconnect from the user session and close the scripting shell:

```
wls_offline:/> exit()
Exiting WLS scripting shell ...
c:\>
```

## readDomain

Opens an existing domain for updating.

### Syntax

```
readDomain(domainDirName)
```

Argument	Definition
<i>domainDirName</i>	Specifies the directory name of the domain that you wish to open.



## Example

The following command opens the specified domain for editing:

```
wls_offline:/> readDomain('c:/bea/user_projects/domains/medrec')
wls_offline:/medrec>
```

## readTemplate

Opens an existing domain template for domain creation.

### Syntax

```
readTemplate(templateFileName)
```

Argument	Definition
<i>templateFileName</i>	The name of the JAR file corresponding to the domain template.

## Examples

The following command opens the specified domain template for domain creation:

```
wls_offline:/> readTemplate('c:/bea/weblogic81/common/templates/domains
/medrec.jar')
wls_offline:/medrec>
```

## updateDomain

Updates and saves the current domain. The domain continues to be editable after you update and save it.

### Syntax

```
updateDomain()
```

<~runChNum>

## Examples

The following command updates and saves the current domain:

```
wls_offline:/> readDomain('c:/bea/user_projects/domains/medrec')
...
wls_offline:/medrec> updateDomain()
```

## writeDomain

Writes the domain configuration information to the specified directory. The domain continues to be editable after you execute this command.

**Note:** The name of the domain is derived from the name of the domain directory. For example, for a domain saved to `c:/bea/user_projects/domains/myMedrec`, the domain name is `myMedrec`.

## Syntax

```
writeDomain(domainDir)
```

Argument	Definition
<i>domainDir</i>	The name of the directory to which you want to write the domain configuration information.

## Examples

The following command writes the domain configuration information to the specified directory:

```
wls_offline:/> readTemplate('c:/bea/weblogic81/common/templates/domains
/medrec.jar')
...
wls_offline:/medrec> writeDomain('c:/bea/user_projects/domains/medrec')
```

## Browse Command (cd)

The `cd` command navigates the hierarchy of configuration objects. This command uses a model that is similar to navigating a file system in a Windows or UNIX command shell. You can navigate to configuration objects in the current hierarchy and to any child or instance.

To navigate back to a parent resource, enter the `cd('..')` command. To get back to the root configuration object after navigating to a resource that is deep in the hierarchy, enter the `cd('/')` command.

The `cd` command returns a stub of the configuration object instance, if one exists.

**Note:** The `cmo` variable is initialized to the root of all domain configuration objects. As you navigate, it is updated to the current configuration object.

## Syntax

`cd(path)`

Argument	Definition
<i>path</i>	Path to the configuration object in the namespace.

## Example

The following example shows how to navigate the hierarchy of configuration objects:

```
wls_offline:/medrec> cd('Server')
wls_offline:/medrec/Server> cd('MedRecServer')
wls_offline:/medrec/Server/MedRecServer> cd('../..')
wls_offline:/medrec>
```

## Edit Commands

[Table 4](#) lists the WLST offline configuration Edit commands that you use to interrogate and edit configuration objects.

**Table 4 Edit Commands for WLST Offline Configuration**

This command...	Enables you to...	For more information, see...
<a href="#">assign</a>	Assign configuration objects to one or more destinations.	<a href="#">“assign” on page 16</a>
<a href="#">assignAll</a>	Assign all applications or services to one or more destinations.	<a href="#">“assignAll” on page 17</a>
<a href="#">create</a>	Create a child object with the specified name and type for the current configuration object.	<a href="#">“create” on page 18</a>

<~runChNum>

**Table 4 Edit Commands for WLST Offline Configuration (Continued)**

This command...	Enables you to...	For more information, see...
<code>delete</code>	Delete an instance of a child configuration with the specified name and type.	<a href="#">“delete” on page 19</a>
<code>get</code>	Return the value of the specified attribute for the current configuration object.	<a href="#">“get” on page 20</a>
<code>loadDB</code>	Load SQL files into a database.	<a href="#">“loadDB” on page 20</a>
<code>set</code>	Set the specified attribute value for the current configuration object.	<a href="#">“set” on page 21</a>
<code>setOption</code>	Set options related to a domain creation or update	<a href="#">“setOption” on page 22</a>
<code>unassign</code>	Unassign configuration objects from one or more destinations.	<a href="#">“unassign” on page 23</a>
<code>unassignAll</code>	Unassign all applications or services from one or more destinations.	<a href="#">“unassignAll” on page 25</a>

## assign

Assigns configuration objects to one or more destinations.

### Syntax

```
assign(sourceType, sourceName, destinationType, destinationName)
```

Argument	Definition
<i>sourceType</i>	Type of configuration object to be assigned. This value can be set to <code>Application</code> , <code>Server</code> , or the name of a security type (such as <code>User</code> ) or service (such as <code>JDBCConnectionPool</code> ). Guidelines for setting this value are provided below.
<i>sourceName</i>	Name of the configuration object to be assigned. Multiple names must be separated by commas.
<i>destinationType</i>	Type of destination. Guidelines for setting this value are provided below.
<i>destinationName</i>	Name of the destination. Multiple names must be separated by commas.

Use the following guidelines for setting the `sourceType` and `destinationType`:

- When assigning **applications** or **application components**, set the values as follows:
  - `sourceType`: `Application`
  - `destinationType`: `Target`
- When assigning **services**, set the values as follows:
  - `sourceType`: Name of the specific server, such as `JDBCConnectionPool`.
  - `destinationType`: `Target`
- When assigning **servers** to **clusters**, set the values as follows:
  - `sourceType`: `Server`.
  - `destinationType`: `Cluster`.
- When assigning **security types**, set the values as follows:
  - `sourceType`: Name of the security type, such as `User`.
  - `destinationType`: Name of the destination security type, such as `Group`.

## Example

The following example performs the following assignments:

- Assigns the servers `myServer` and `myServer2` to the cluster `myCluster`.
- Assigns the application `MedRecEAR` to the target server `newServer`.
- Assigns the user `newUser` to the group `Monitors`.

```
wls_offline://medrec> assign("Server", "myServer,myServer2", "Cluster",
"myCluster")
wls_offline://medrec> assign("Application", "MedRecEAR", "Target",
"newServer")
wls_offline://medrec> assign("User", "newUser", "Group", "Monitors")
```

## assignAll

Assigns all applications or services to one or more destinations.

**Note:** Note that you must assign JMS server and JMS distributed destinations using the `assign()` command, as described in [“assign” on page 16](#).

<~runChNum>

## Syntax

`assignAll(sourceType, destinationType, destinationName)`

Argument	Definition
<i>sourceType</i>	Type of applications or services to be assigned. This value can be set to Applications or Services.
<i>destinationType</i>	Type of destination. This value must be set to Target.
<i>destinationName</i>	Name(s) of the destination. Multiple names must be separated by commas.

## Example

The following example assigns all services to the servers `adminServer` and `cluster1`.

```
wls_offline:/mydomain> assignAll("Services", "Target",  
"adminServer,cluster1")
```

The following services, if present, are assigned to the specified targets:

MigratableRMIService, Shutdownclass, Startupclass, FileT3, RMCFactory, MailSession, MessagingBridge, JMSConnectionFactory, JDBCConnectionPool, JDBCMultipool, JDBCTxDatasource, JDBCDataSource, JDBCPoolComp, JoltConnectionPool, WLECCConnectionPool, and WTCServer.

## create

Creates a child object with the specified name and type for the current configuration object. The `create` command returns a stub for the newly created configuration object.

**Note:** Child types must be created under an instance of their parent type.

## Syntax

`create(Name, childObjectType)`

Argument	Definition
<i>Name</i>	Name for the configuration object that you are creating.

Argument	Definition (Continued)
<i>childObjectType</i>	Type of child object that you are creating for the current configuration object. You can create instances of any type defined in the <code>config.xml</code> file except custom security types.

## Example

The following example creates a child object of type `Server` named `newServer` for the current configuration object, storing the stub as `server1`:

```
wls_offline:/medrec> server1=create('newServer','Server')
wls_offline:/medrec> server1.getName()
'newServer'
```

## delete

Deletes an instance of a child object with the specified name and type for the current configuration object.

## Syntax

```
delete (Name, childObjectType)
```

Argument	Definition
<i>Name</i>	Name of the child object to delete for the current configuration object.
<i>childObjectType</i>	Type of the child object to be deleted. You can delete instances of any type defined in the <code>config.xml</code> file.

## Example

The following example deletes the child object of type `Server` named `newServer` for the current configuration object:

```
wls_offline:/medrec> delete('newServer','Server')
```

<~runChNum>

## get

Returns the value of the specified attribute for the current configuration object.

**Note:** You can list all attributes and their current values using the `a` argument to the `ls` command. For more information, see [“ls” on page 28](#).

Alternatively, you use the `cmo` variable to perform any `get` method on the current configuration object. For example:

```
cmo.getListenPort()
```

For more information about the `cmo` variable, see [“WLST Offline Configuration Variables” on page 7](#).

## Syntax

```
get(attributeName)
```

Argument	Definition
<i>attributeName</i>	Name of the attribute to be displayed.

## Example

The following example returns the value of the `AdministrationPort` for the current configuration object:

```
wls_offline:/medrec> get('AdministrationPort')
9002
```

## loadDB

Loads SQL files into a database.

Before executing this command, ensure that the following conditions are true:

- The appropriate database is running.
- SQL files exist for the specified database and version.

To verify that the appropriate SQL files exist, open the domain template and locate the relevant SQL file list, `jdbc.index`, in the `_jdbc_` directory. For example, for PointBase version 4.4, the SQL file list is located at `_jdbc_\Pointbase\44\jdbc.index`.



The command fails if the above conditions are not met.

## Syntax

```
loadDB(DBVersion, connectionPoolName)
```

Argument	Definition
<i>DBVersion</i>	Version of the database for which the SQL files are intended to be used.
<i>ConnectionPoolName</i>	Name of the JDBC connection pool to be used to load SQL files.

## Example

The following example loads SQL files, intended for version 4.4 of the database, using the `myPool-PointBase` JDBC connection pool:

```
wls_offline:/mydomain> loadDB('4.4', 'myPool-PointBase')
```

## set

Sets the specified attribute value for the current configuration object.

**Note:** You can list all attributes and their current values using the `a` argument to the `ls` command. For more information, see [“ls” on page 28](#).

When you use offline WLST to set password attributes, you need to use the encrypted attribute. For example, to update `CustomIdentityKeyStorePassPhrase`, you have to use the `CustomIdentityKeyStorePassPhraseEncrypted` attribute. In this example, if you do not use the encrypted attribute, the "Unable to find attribute `CustomIdentityKeyStorePassPhrase`" message appears.

Alternatively, you use the `cmo` variable to perform any `set` method on the current configuration object. For example:

```
cmo.setPassword('weblogic')
```

For more information about the `cmo` variable, see [“WLST Offline Configuration Variables” on page 7](#).

## Syntax

```
set(attributeName, attributeValue)
```

<~runChNum>

Argument	Definition
<i>attributeName</i>	Name of the attribute to be set.
<i>attributeValue</i>	Value of the attribute to be set. This value does not need to be enclosed in single or double quotes are required.

## Example

The following example sets the `ArchiveConfigurationCount` attribute of `DomainMBean` to 10:

```
wls_offline:/mydomain/config> set('ArchiveConfigurationCount',10)
wls_offline:/mydomain/config>
```

## setOption

Sets options related to a domain creation or update.

## Syntax

```
setOption(optionName,optionValue)
```

Argument	Definition
<i>optionName</i>	<p>Name of the option to set.</p> <p>Available options for <b>domain creation</b> include:</p> <ul style="list-style-type: none"> <li>• <code>CreateStartMenu</code>—specifies whether to create a Start Menu shortcut on a Windows platform.</li> <li>• <code>JavaHome</code>—specifies the home directory for the JVM to be used when starting the server.</li> <li>• <code>OverwriteDomain</code>—specifies whether to allow an existing domain to be overwritten. The default is <code>false</code>.</li> <li>• <code>ServerStartMode</code>—specifies the mode to use when starting the server for the newly created domain. This value can be <code>dev</code> (development) or <code>prod</code> (production).</li> </ul> <p>Available options for <b>domain updates</b> include:</p> <ul style="list-style-type: none"> <li>• <code>AllowCasualUpdate</code>—specifies whether allow a domain to be updated without adding an extension template. The default is <code>true</code>.</li> <li>• <code>ReplaceDuplicates</code>—specifies whether to keep original configuration elements in the domain or replace the elements with corresponding ones from an extension template when there is a conflict.</li> </ul> <p>Available options for both <b>domain creation</b> and <b>domain updates</b> include:</p> <ul style="list-style-type: none"> <li>• <code>AppDir</code>—specifies the application directory to be used when a separate directory is desired for applications, as specified by the template.</li> <li>• <code>AutoDeploy</code>—specifies whether to activate auto deployment when a cluster or multiple Managed Servers are created. This option defaults to <code>true</code>. To deactivate this feature, set the option to <code>'False'</code> on the first line of your script.</li> </ul>
<i>optionValue</i>	Value for the option.

## Example

The following example sets the `CreateStartMenu` option to `false`:

```
wls_offline:> setOption('CreateStartMenu', 'false')
```

## unassign

Unassign configuration objects from one or more destinations.

## Syntax

```
unassign(sourceType, sourceName, destinationType, destinationName)
```

Argument	Definition
<i>sourceType</i>	Type of configuration object to be unassigned. This value can be set to Application, Server, or the name of a security type (such as User) or service (such as JDBCConnectionPool). Guidelines for setting this value are provided below.
<i>sourceName</i>	Name of the configuration object to be unassigned. Multiple names must be separated by commas.
<i>destinationType</i>	Type of destination. Guidelines for setting this value are provided below.
<i>destinationName</i>	Name of the destination. Multiple names must be separated by commas.

Use the following guidelines for setting the *sourceType* and *destinationType*:

- When unassigning **applications** or **application components**, set the values as follows:
  - *sourceType*: Application
  - *destinationType*: Target
- When unassigning **services**, set the values as follows:
  - *sourceType*: Name of the specific server, such as JDBCConnectionPool.
  - *destinationType*: Target
- When assigning **servers** from **clusters**, set the values as follows:
  - *sourceType*: Server.
  - *destinationType*: Cluster.
- When unassigning **security types**, set the values as follows:
  - *sourceType*: Name of the security type, such as User.
  - *destinationType*: Name of the destination security type, such as Group.

## Example

The following example unassigns the following resources:

- Unassigns the servers `myServer` and `myServer2` from the cluster `myCluster`.
- Unassigns the application `MedRecEAR` from the target server `newServer`.
- Unassigns the user `newUser` from the group `Monitors`.

```
wls_offline:/medrec> unassign("Server", "myServer,myServer2", "Cluster",
"myCluster")
wls_offline:/medrec> unassign("Application", "MedRecEAR", "Target",
"newServer")
wls_offline:/medrec> unassign("User", "newUser", "Group", "Monitors")
```

## unassignAll

Unassigns all applications or services from one or more destinations.

### Syntax

```
unassignAll(sourceType, destinationType, destinationName)
```

Argument	Definition
<i>sourceType</i>	Type of applications or services to be unassigned. This value can be set to Applications or Services.
<i>destinationType</i>	Type of destination. This value must be set to 'Target'.
<i>destinationName</i>	Name(s) of the destination. Multiple names must be separated by commas.

### Example

The following example unassigns all services from the servers `adminServer` and `cluster1`.

```
wls_offline:/medrec> unassignAll("Services", "Target",
"adminServer,cluster1")
```

The following services, if present, are unassigned from the specified targets:

MigratableRMIService, Shutdownclass, Startupclass, FileT3, RMCFactory, MailSession, MessagingBridge, JMSConnectionFactory, JDBCConnectionPool, JDBCMultipool, JDBCTxDatasource, JDBCDataSource, JDBCPoolComp, JoltConnectionPool, WLECCConnectionPool, and WTCServer.

## Information Commands

[Table 5](#) lists the information commands for WLST offline configuration that are useful for interrogating domains, servers, and variables, and providing configuration object and WLST-related information.

**Table 5 Information Commands for WLST Offline Configuration**

This command...	Enables you to...	For more information, see...
<code>dumpStack</code>	Display any stack trace that was produced while performing a WLST action.	<a href="#">“dumpStack” on page 26</a>
<code>dumpVariables</code>	Display all variables used by WLST, including their name and value.	<a href="#">“dumpVariables” on page 27</a>
<code>help</code>	Provide syntax and usage information for all categories of WLST commands, when no arguments are specified, or for a single command, when a command or variable name argument is specified.	<a href="#">“help” on page 27</a>
<code>ls</code>	List all child objects or attributes for the current configuration object.	<a href="#">“ls” on page 28</a>
<code>prompt</code>	Toggle the display of the configuration object navigation path information at the prompt, when entered without an argument.	<a href="#">“prompt” on page 30</a>
<code>pwd</code>	Display the current location in the hierarchy of the configuration tree.	<a href="#">“pwd” on page 31</a>
<code>startRecording</code>	Record all user interactions with WLST; useful for capturing commands.	<a href="#">“startRecording” on page 31</a>
<code>stopRecording</code>	Stop recording WLST commands.	<a href="#">“stopRecording” on page 32</a>

Each command is described in detail in the following sections.

### dumpStack

Displays any stack trace that was produced while performing a WLST action.

## Syntax

```
dumpStack()
```

## Example

This example instructs WLST to display the stack trace.

```
wls_offline:/medrec> dumpStack()
com.bea.plateng.domain.script.jython.WLSTException:
java.lang.reflect.InvocationTargetException
...
```

## dumpVariables

Displays all the variables used by WLST, including their name and value.

## Syntax

```
dumpVariables()
```

## Example

This example instructs WLST to display all the current variables and their values.

```
wls_offline:/platform>dumpVariables()
cmo: Proxy for medrec: Name=medrec, Type=DomainConfig
exitonerror: true
isrecording: false
```

## help

Provides syntax and usage information for all categories of WLST commands, when no arguments are specified, or for a single command or variable, when a command or variable name argument is specified.

The `help` command will support a query; for example, `help('get*')` displays the syntax and usage information for all commands that begin with `get`.

## Syntax

```
help([ 'name' ])
```

<~runChNum>

Argument	Definition
<i>name</i>	Optional. Command or variable name for which information is requested.

### Example

In the following example, information about using the `get` command is requested:

```
wls:/medrec> help('get')
```

The command returns the following:

```
Description:
Get any attribute for the currently navigated configuration object.
Syntax:
get(AttributeName)
AttributeName = current configuration object's attribute name.
Example:
wls_offline:/medrec/Server/MedRecServer>get('AcceptBacklog')
50
wls_offline:/medrec/Server/MedRecServer>
```

### ls

Lists all the child objects or attributes for current configuration object.

You can optionally control the output by specifying an optional argument.

[Table 6](#) describes the `ls` command output information.

**Table 6** `ls` Command Output Information

Output	Definition
d	A configuration object with which you can use the <code>cd</code> command; analogous to a directory in a UNIX or Windows file system.
r	Readable property.
w	Writeable property.
x	Executable operation.



## Syntax

```
ls(['a' | 'c'])
```

Argument	Definition
a	Optional. Display all the attribute names and values for the current configuration object. If the attribute is encrypted, WLST displays six asterisks (*****).
c	Optional. Display all the child objects that are contained in the current configuration object. This argument is the default.

## Example

The following command displays the child objects for the current configuration object:

```
wls_offline:/medrec> ls()
drw-   Application
drw-   Cluster
drw-   FileRealm
drw-   JDBCConnectionPool
drw-   JDBCTxDataSource
drw-   JMSConnectionFactory
drw-   MailSession
drw-   PasswordPolicy
drw-   Realm
drw-   Security
drw-   Server
```

The following command displays all the attribute names and values for the current configuration object:

```
wls_offline:/medrec> ls('a')
-rw-   AdministrationPort                9002
-rw-   AdministrationPortEnabled          false
-rw-   AutoConfigurationSaveEnabled       false
-rw-   ConfigurationVersion               8.1.2.0
-rw-   ConsoleContextPath                 console
-rw-   ConsoleEnabled                     true
-rw-   LastModificationTime               0
-rw-   Name                               medrec
```

<~runChNum>

```
-rw-      Notes                               null
-rw-      ProductionModeEnabled              false
wls_offline:/medrec>
```

prompt

Toggles the display of the configuration object navigation path information at the prompt, when entered without an argument.

This command is useful when the prompt becomes too long due to the length of the configuration object navigation path. For WLST offline configuration, when you disable the prompt details, the prompt displays as follows: `wls_offline: />`. In this case, to determine your current location in the hierarchy, you can use the `pwd` command, as described in “[pwd](#)” on page 31. By default, the WLST prompt displays the configuration object navigation path information.

Syntax

```
prompt(['off' | 'on'])
```

Argument	Definition
'off'   'on'	Optional. Hides or displays WLST prompt, as follows: <ul style="list-style-type: none"><li>The <code>off</code> argument hides the WLST prompt and defaults to the Jython prompt. You can create a new prompt using Jython syntax.</li><li>The <code>on</code> argument displays the default prompt, including the configuration object navigation path information.</li></ul>

Example

The following example instructs WLST to hide and redisplay the configuration object navigation path information at the prompt:

```
wls_offline:/medrec/Server/MedRecServer> prompt()
wls_offline:/>
wls_offline:/> prompt()
wls_offline:/medrec/Server/MedRecServer
```

The following example instructs WLST to hide the prompt and default to the Jython prompt, change the Jython prompt, and redisplay the WLST prompt:

```
wls_offline:/medrec/Server/MedRecServer> prompt('off')
>>>
>>>sys.ps1="myprompt>"
myprompt>
myprompt> prompt('on')
wls:/medrec/Server/MedRecServerfs>
```

## pwd

Displays the current location in the hierarchy of the configuration tree.

### Syntax

```
pwd()
```

### Example

The following command displays the current location in the hierarchy of the configuration object:

```
wls_offline:/medrec> pwd()
/medrec
```

## startRecording

Records all user interactions with WLST; useful for capturing commands.

### Syntax

```
startRecording(recordFilePath)
```

Argument	Definition
<i>recordFilePath</i>	File pathname for storing WLST commands.

<~runChNum>

## Example

The following example instructs WLST to begin recording WLST commands in the `record.py` file:

```
wls_offline:> startRecording('c:/myScripts/record.py')
Starting recording to c:\myScripts\record.py
```

## stopRecording

Stops recording WLST commands.

## Syntax

```
stopRecording()
```

## Example

The following example instructs WLST to stop recording WLST commands:

```
wls_offline:> stopRecording()
Stopping recording to c:\myScripts\record.py
```