

Oracle® WebLogic Operations Control

Configuration Guide

10g Release 3 (10.3)

September 2008

Copyright © 2007, 2008, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Introduction

Guide to this Document	1-1
Related Documents	1-2

Overview

WLOC Components	2-1
Configuration Types	2-3
Configuration Workflow	2-3

Configuring the Controller and Agents

Overview	3-1
Controller	3-1
Agent	3-2
Agent Configurations	3-3
Configuring a Plain Agent	3-3
Configuring an ESX Agent	3-8
Modifying Agent Configurations	3-15
Controller Configurations	3-15
Configuring a Controller	3-15
Modifying Controller Configurations	3-21
Adding Agents to a Controller	3-22

Starting and Stopping the Controller and Agents

Starting an Agent	4-1
Starting the Controller	4-2
Accessing the WLOC Administration Console	4-2
Stopping an Agent	4-2
Stopping the Controller	4-3
Running the Controller and Agents as Windows Services.	4-3
Unexpected Shutdowns	4-3

Configuring Services and Process Groups

Overview	5-1
Service Process Management	5-3
Configuring a Service.	5-3
Configuring a Process Group	5-4
Resource Requirements	5-7
Minimums and Maximums	5-7
Ready Metrics for Processes	5-8
JVM Arguments for Processes.	5-10
Setting Communication Protocols.	5-11
Setting the JMX Port.	5-13
Determining the CLASSPATH	5-13
Creating Multiple Java Processes Using Templates.	5-14
Defining the Common JVM Parameters.	5-15
Specifying the Number of Processes.	5-15
Defining Variables.	5-15
Sample Template Using Variables	5-17
Service Deployment Process	5-19

Defining Policies

Policy Components	6-1
Creating Policies	6-3
Policy Types	6-4
Deployment Policies	6-5
Runtime Policies	6-5
Administrative Policies	6-6
Constraint Types	6-6
Action Types	6-11
Action Pipelines	6-13

Configuring Security

WLOC Users, Groups, and Security Roles	7-1
WLOC Boot User	7-2
Users and Groups	7-2
WLOC Security Roles	7-3
Secure Communications	7-4
Configuring Firewalls	7-5
Securing WLOC Administration Console to Controller Communication	7-7
Securing Controller to Agents Communication	7-7
Securing Agent to VMware Virtual Center Communication	7-9
Securing Agent to MBean Server Communication	7-10
Keystores	7-10
Controller Keystores	7-13
Agent Keystores	7-14
Password Encryption	7-16
File System Security	7-16

Logging, Auditing, and Monitoring

Logging	8-1
Configuring Logging	8-1
Viewing Log Messages	8-3
Log Message Format	8-4
Output to Standard Out and Standard Error	8-4
Log Message Attributes	8-5
Message Severity	8-6
Rotating Log Files	8-6
Debug Log Messages	8-7
Auditing WLOC Actions	8-7
Audit Event Types	8-8
Audit Format	8-9
Monitoring	8-10
Monitoring Service Performance	8-10
Viewing Events	8-10

Silent Mode Configurations

Running the Configuration Wizard in Silent-Mode	A-1
Plain Agent Template XML File	A-2
Plain Agent Configuration Values	A-3
ESX Agent Template XML File	A-5
ESX Agent Configuration Values	A-7
Controller Template XML File	A-10
Controller Configuration Values	A-12

Introduction

This document describes configuration tasks associated with using WebLogic Operations Control™ (WLOC) to manage the deployment and runtime performance of applications.

See [“Related Documents” on page 1-2](#) for a description of other WLOC documents.

Guide to this Document

This document includes the following sections:

- [Chapter 2, “Overview”](#) provides an overview of WLOC with respect to the configuration tasks explained in this document.
- [Chapter 3, “Configuring the Controller and Agents”](#) describes how to configure the WLOC Controller and Agents using the WLOC Configuration Wizard.
- [Chapter 4, “Starting and Stopping the Controller and Agents”](#) describes how to start and stop WLOC Controllers and Agents.
- [Chapter 5, “Configuring Services and Process Groups”](#) describes configuration tasks associated with defining and managing WLOC services.
- [Chapter 6, “Defining Policies”](#) describes the components of a policy, explains how policies are used in WLOC, and gives information about how they are created.
- [Chapter 7, “Configuring Security”](#) describes the WLOC security model and provides information about protecting access to WLOC and securing WLOC communications.

- [Chapter 8, “Logging, Auditing, and Monitoring”](#) describes how to monitor, log, and audit WLOC services and resources.
- [Appendix A, “Silent Mode Configurations,”](#) describes how to run the WLOC Configuration Wizard in silent mode.

Related Documents

In addition to this document, the WLOC documentation set includes the following:

- [Introducing WebLogic Operations Control](#)—Provides a high-level overview of WLOC.
- [Installation Guide](#)—Describes how to install and uninstall the WLOC components.
- [WLOC Administration Console Online Help](#)—The online help for WLOC’s graphical user interface. You can also access the WLOC Administration Console Help by clicking the Help link in the upper right corner of the Administration Console.
- [Securing a Production Environment](#)—Highlights essential security measures for you to consider before you deploy WLOC into a production environment.
- [Controller Configuration Schema Reference](#)—A reference to the XML Schema used to persist the configuration of the WLOC Controller component.
- [Agent Configuration Schema Reference](#)—A reference to the XML Schema used to persist the configuration of the WLOC Agent component.
- [Service Metadata Schema Reference](#)—A reference to the XML Schema used to persist the configuration of WLOC services.
- [Message Catalog](#)—A reference to messages generated by WLOC.

Overview

This section provides an overview of WLOC with respect to the configuration tasks explained in this document.

- [“WLOC Components” on page 2-1](#)
- [“Configuration Types” on page 2-3](#)
- [“Configuration Workflow” on page 2-3](#)

WLOC Components

A WLOC environment consists of the following components:

- **WLOC Controller**

The Controller is the central component that gathers data about the operating environment from Agents. It uses the gathered data to enforce policies and to deploy new services in order to honor the Service Level Agreements (SLAs) of all deployed services. The Controller hosts the WLOC Administration Console.

- **WLOC Agents**

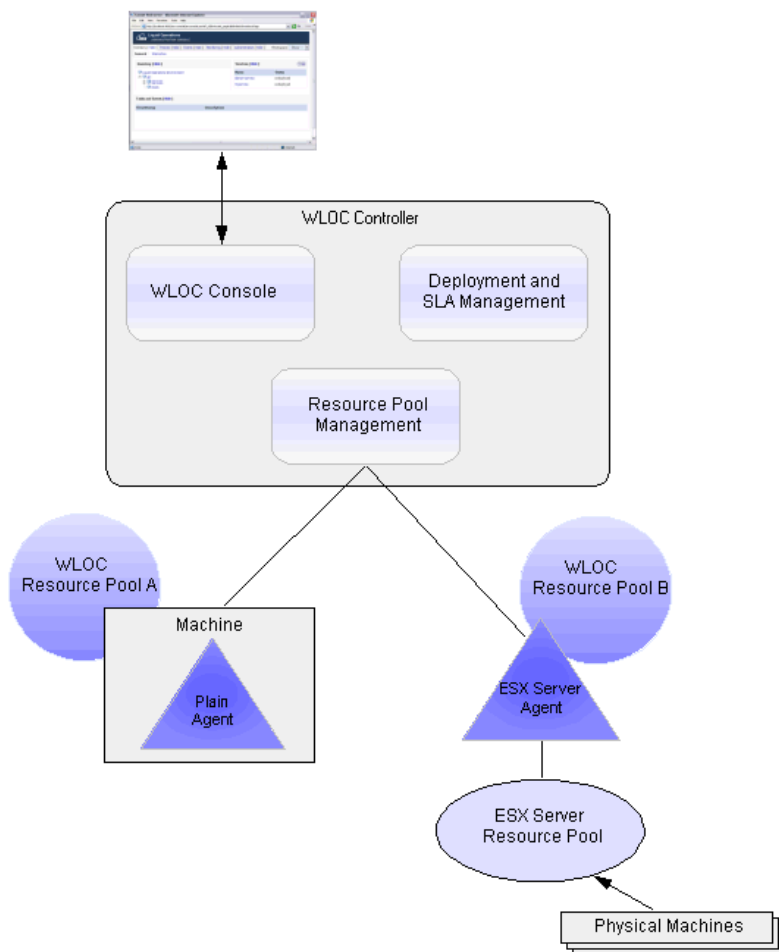
WLOC Agents provide information about the environment to the Controller, start and stop processes, and invoke other actions at the Controller’s request. A Plain Agent gathers data and manages processes on a single physical machine. An ESX Agent communicates with the VMware Virtual Center to gather data about the VMware resource pools in order to manage LiquidVM instances.

- **Managed Java Processes**

Configuration tasks associated with Java processes includes service definitions, instructions for starting and stopping processes, and managing policies that govern service deployment and adaptive actions that ensure compliance with service requirements.

Figure 2-1 provides a high-level view of the relationship of WLOC components.

Figure 2-1 WLOC Components — Overview



Configuration Types

Configuration tasks can be considered of three types:

- **WLOC Controller Configurations**

A Controller configuration defines information used to start the Controller, connect to WLOC Agents, capture event logs, and notification methods to use when policy definitions are not being met by the managed application.

- **WLOC Agent Configurations**

A WLOC agent configuration consists of its operational settings (address, connection ports, log level, etc.) and information about its managed resource pool (CPU cycles, memory, and IP addresses).

- **Service Metadata**

Service metadata consists of:

- The organization of processes into WLOC services. Each service consists of logically related processes organized into process groups. Each process is a software stack starting from a Java Virtual Machine (JVM) and include the classes that are running in the JVM.
- The WLOC policies that specify deployment requirements and adaptive runtime actions that should be taken if deployment requirements are not being satisfied.

Configuration Workflow

This section provides a high-level and logical description of the configuration tasks described in this document.

- **Setting Up the Controller**

This involves installing the WLOC Controller and creating the Controller instance with initial configuration. For instructions on configuring a Controller, see [“Configuring a Controller” on page 3-15](#). In addition, instructions on performing silent mode Controller configurations are located in [Appendix A, “Silent Mode Configurations.”](#)

- **Establishing the WLOC Resource Environment**

- Installing WLOC Agents and creating Agent instances with initial configuration. For instructions on configuring a Plain Agent, see [“Configuring a Plain Agent” on page 3-3](#). For an ESX Agent, see [“Configuring an ESX Agent” on page 3-8](#). In

addition, instructions on performing silent mode Agent configurations are located in [Appendix A, “Silent Mode Configurations.”](#)

- **Establishing the WLOC Runtime Environment**

This includes starting the WLOC Controller and Agent(s) and accessing the WLOC Administration Console. This information is found in [“Starting and Stopping the Controller and Agents” on page 4-1.](#)

- **Defining WLOC Services**

This includes setting up WLOC services under management and defining the service’s initial deployment requirements as described in [“Configuring Services and Process Groups” on page 5-1.](#)

- **Establishing the Adaptive Runtime Policies**

This includes defining adaptive runtime policies to ensure service requirements are being met. For detailed information, see [“Defining Policies” on page 6-1.](#)

- **Managing Security**

This includes securing access to WLOC and configuring secure communications between WLOC components and managed processes. Information about securing access to WLOC is contained in [“Configuring Security” on page 7-1.](#) In addition, achieving secure communication between WLOC components is an essential part of Controller and Agent configurations.

- **Monitoring Managed Applications**

To ensure SLAs are being met, the runtime behavior of the managed applications are monitored to determine if additional policies are needed or existing policies require refinement. For information, see [“Logging, Auditing, and Monitoring” on page 8-1.](#)

Configuring the Controller and Agents

This section describes how to configure the WLOC Controller and Agents. It includes the following sections:

- [“Overview” on page 3-1](#)
- [“Agent Configurations” on page 3-3](#)
- [“Controller Configurations” on page 3-15](#)

Overview

This section describes the Controller, Plain Agent, and ESX Agent configurations.

Controller

The WLOC Controller configuration settings control the behavior of the Controller and how it communicates with the WLOC Agents.

The initial Controller configuration is defined by running the WLOC Configuration Wizard. This process stores the configuration settings in an XML file that is used to establish the Controller’s runtime configuration at startup. The XML file is named `loc-controller-config.xml` and, by default, is located in the following directory:

```
BEA_HOME\user_projects\controller\config
```

After the creation of the XML file, the Controller configuration can be changed using the WLOC Administration Console, by directly editing the XML file, or by re-running the configuration

wizard. Modifications made using the Administration Console take immediate effect, while changes made using the other two methods do not take effect until the Controller is restarted.

Changes to a Controller configuration using the Configuration Wizard or the WLOC Administration Console are captured in the Controller's audit log located in the *BEA_HOME\user_projects\controller\logs* directory, where *user_projects* is the directory specified while running the Configuration Wizard.

Agent

A WLOC Agent configuration controls the behavior of the Agent and how it communicates with the WLOC Controller and the managed application. An Agent is required to manage each resource pool in the WLOC environment. The Agent discovers information about the resources available and maintains that information in its configuration.

There are two types of Agents:

- **Plain Agent**—a Plain Agent manages a resource pool on a physical machine. Configuration settings include the amount of CPU to allocate for WLOC and path names to software that is available to WLOC services.
- **ESX Agent**—an ESX Agent manages a resource pool on a virtual machine that has been configured by hypervisor software. The Agent communicates with the VirtualCenter server to discover the capabilities of the resource pool and allocates all resources in the resource pool as WLOC resources. ESX Agent configuration information includes the path to available ISO images and NFS shares.

An Agent configuration is stored in an XML file named *loc-agent-config.xml* located in the *BEA_HOME\user_projects\agent1\config* directory, where *agent1* is the directory where the agent was installed.

The creation of an Agent instance and its initial configuration must be performed with the WLOC Configuration Wizard. Thereafter, the configuration can be modified using the Administration Console or by directly editing its configuration file.

Changes to an Agent configuration using the WLOC Administration Console are captured in the Agent's audit log located in the *BEA_HOME\user_projects\agent1\logs* directory, where *user_projects\agent1* is the directory specified while running the Configuration Wizard.

Agent Configurations

This section provides step-by-step instructions for running the WLOC Configuration Wizard in GUI mode. You can also use the configuration wizard in silent mode, which is a noninteractive method of configuring your software that requires the use of an XML properties file for selecting configuration options. For details about performing silent-mode configurations, see [Appendix A, “Silent Mode Configurations”](#).

Note: Running the Configuration Wizard in console mode is currently not supported.

Configuring a Plain Agent

Follow these steps to create a plain Agent instance and its initial configuration:

1. Invoke the Configuration Wizard as described in [Table 3-1](#) and click **Next** in the **Welcome** window.

Table 3-1 Invoking the WLOC Configuration Wizard

On this platform . . .	Perform the following steps . . .
Windows	<ul style="list-style-type: none"> • From the Windows Start Menu, choose Start > All Programs > WebLogic Operations Control 10gR3>WLOC Configuration Wizard • From an MS-DOS Command Prompt window, go to the <code>BEA_HOME\WLOC_HOME\common\bin\</code> directory and enter <code>config.cmd</code> at the prompt. In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation, for example <code>C:\bea\wloc_10.3</code>.
UNIX or Linux	<p>Go to the <code>BEA_HOME/WLOC_HOME/common/bin/</code> directory and enter <code>config.sh</code> at the prompt.</p> <p>In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation.</p>

2. In the **Choose Controller or Agent** window, select **Create a new Agent for this host** and click **Next**.
3. In the **Enter Agent Directory Location** window, accept the default location or specify a different directory and click **Next**.

By default, the Configuration Wizard creates the Agent in the `BEA_HOME/user_projects/agent1` directory, but you can specify any name and directory location you choose.

4. In the **Configure Agent Connection Details** window, complete the fields as described in [Table 3-2](#).

Table 3-2 Configure Agent Connection Details

In this field . . .	Enter the following . . .
Agent Name	<p>Name for the Agent. The Administration Console displays this as the Agent name.</p> <p>Note: Managing multiple Agents through the Administration Console requires that each Agent has a unique name.</p>
Agent Host	<p>Fully-qualified name of the machine hosting the agent, for example: <code>agentbox.east.example.com</code>.</p>
Agent Port	<p>HTTP port number that the Agent uses when communicating with the Controller in unsecure mode; default: 8001.</p> <p>Note: In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>
Agent Secure Port	<p>HTTPS port number that the Agent uses when communicating with the Controller in secure mode using SSL; default: 8002.</p> <p>Note: In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>

Table 3-2 Configure Agent Connection Details (Continued)

In this field . . .	Enter the following . . .
Transfer Encryption Passphrase/ Confirm Transfer Encryption Passphrase	<p>Passphrase used to apply encryption, beyond the Security Mode setting, to certain sensitive data passed between the Controller and Agent. The password must be a minimum of 8 characters.</p> <p>If Security Mode is set to Unsecure, this setting will still encrypt the most sensitive data.</p> <p>Note: This passphrase must be entered when adding the Agent to the Controller or communication between the Controller and Agent will fail.</p> <p>For development environments, it is sufficient to accept the defaults. You do not need to know these passphrases.</p>
Security Mode	<p>Security mode used for connections with the Controller.</p> <ul style="list-style-type: none"> • Unsecure — Communication with the Controller uses HTTP protocol. Unsecure mode is sufficient for development environments. • Secure — Communication with the Controller uses SSL protocol. Secure mode ensures confidentiality and integrity of the communication and requires setting up trust as an explicit step between the Controller and the Agent. Oracle recommends that you use Secure mode for production environments. <p>For more information, see “Securing Controller to Agents Communication” on page 7-7.</p> <p>Note: Both the Controller and Agent must be set to the same security mode.</p>

5. In the **Configure Agent Logging** window, complete the fields as described in [Table 3-3](#).

Table 3-3 Configure Agent Logging

In this field . . .	Specify the following . . .
Logfile severity	Severity of events written to the log file. The default is INFO. In the order of severity from least severe to most severe, the log levels are: TRACE, DEBUG, INFO, NOTICE, WARNING, ERROR, CRITICAL, ALERT, EMERGENCY Severity levels are inclusive. When set to INFO, the log will include NOTICE, WARNING, ERROR, CRITICAL, ALERT, and EMERGENCY events.
Stdout severity	Severity of events that are written to stdout. The default is INFO. Stdout severity uses the same severity levels as the log file.
Logfile location	Filename and directory location for the Agent log file. You can accept the default or specify a different directory and/or log file name. Default: <i>BEA_HOME\user_projects\agent1\logs\Agent.log</i>

6. In the **Configure Agent Keystore Passwords** window, you are prompted for the Agent keystore passwords used for internal WLOC communications. In most cases, this depends on whether you are using WLOC in a production or development environment, as described in [Table 3-4](#). Click **Next** after completing this window.

Table 3-4 Configure Agent Keystore Passwords

In this environment . . .	Do the following . . .
Development	Click Next to use the default keystores passwords.
Production	Enter the passwords that will be used to secure the keystores used for production-level communications between WLOC components. Make a note of these passwords. They will be required later when setting trust by importing certificates into the keystores. For more information about the WLOC Agent keystores, see “Keystores” on page 7-10 .

7. In the **Configure Agent Type** window, select **Plain Agent** and click **Next**.

8. In the **Configure Plain Agent (1 of 2)** window, complete the fields as described in [Table 3-5](#) and click **Next**.

Table 3-5 Plain Agent Configuration (1 of 2)

In this field . . .	Enter the following . . .
Resource Pool Name	A unique name for the resource pool managed by the Agent. Note: Managing multiple resource pools through the Administration Console requires that each resource pool has a unique name.
Description	An arbitrary description of the resource pool. This description appears in the console.
CPU capacity (MHz)	(Optional) CPU capacity (in normalized megahertz) available to the resource pool.
Stdout directory	Complete pathname for the directory to which the JVM standard output stream, stdout, should be directed. Default: <i>BEA_HOME</i> \user_projects\agent1\stdout
Stderr directory	Complete pathname for the directory to which the standard error stream, stderr, output should be directed. Default: <i>BEA_HOME</i> \user_projects\agent1\stderr

9. In the **Configure Plain Agent (2 of 2)** window, specify the software that is available to the Agent by clicking **Add** and completing each field as described in [Table 3-6](#). When you are finished, click **Next**.

Note: You may skip this step and use the Administration Console to provide the information at a later time.

Table 3-6 Configure Plain Agent (2 of 2)

In this field . . .	Enter the following . . .
Name	Name for the software as it will appear in the console.
Description	An arbitrary description of the software as it will appear in the console.
Path	The complete path to the directory containing the software. For example, to specify WLS, use <i>WL_HOME</i> /server/lib, where <i>WL_HOME</i> is the directory in which WLS is installed

- 10. In the **Create Agent Configuration** window, click **Create**. Progress messages are displayed as the Agent is created.
- 11. When the Agent has been successfully created, click **Done** to exit the Configuration Wizard.

Configuring an ESX Agent

Follow these steps to create an ESX Agent instance and its initial configuration:

Note: Running the Configuration Wizard in console mode is currently not supported.

- 1. Invoke the Configuration Wizard as described in [Table 3-7](#) and click **Next** in the **Welcome** window.

Table 3-7 Invoking the WLOC Configuration Wizard

On this platform . . .	Perform the following steps . . .
Windows	<ul style="list-style-type: none">• From the Windows Start Menu, choose Start > All Programs > WebLogic Operations Control 10gR3>WLOC Configuration Wizard• From an MS-DOS Command Prompt window, go to the <i>BEA_HOME\WLOC_HOME\common\bin\</i> directory and enter <i>config.cmd</i> at the prompt. In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation, for example <i>C:\bea\wloc_10.3</i>.
UNIX or Linux	<p>Go to the <i>BEA_HOME/WLOC_HOME/common/bin/</i> directory and enter <i>config.sh</i> at the prompt.</p> <p>In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation.</p>

- 2. In the **Choose Controller or Agent** window, select **Create a new Agent for this host** and click **Next**.
- 3. In the **Enter Agent Directory Location** window, accept the default location or specify a different directory under *user_projects* and click **Next**.
- 4. In the **Configure Agent Connection Details** window, complete the fields as described in [Table 3-8](#).

Table 3-8 Configure Agent Connection Details

In this field . . .	Enter the following . . .
Agent Name	Name for the Agent. The Administration Console displays this as the Agent name.
Agent Host	Fully-qualified host name where the Agent resides; example: <code>agentbox.east.example.com</code> .
Agent Port	<p>HTTP port number used when the Agent and Controller are connecting in unsecure mode; default: 8001.</p> <p>Note: In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>
Agent Secure Port	<p>HTTPS port number used when the Agent and Controller are connecting in secure mode using SSL; default: 8002.</p> <p>Note: In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>
Transfer Encryption Passphrase/ Confirm Transfer Encryption Passphrase	<p>Passphrase used to apply encryption beyond the Security Mode setting to certain sensitive data passed between the Controller and Agent. The password must be a minimum of 8 characters.</p> <p>Even if Security Mode is Unsecure, this setting will encrypt the most sensitive data.</p> <p>Note: This passphrase must be entered when adding the Agent to the Controller or communication between the Controller and Agent will fail.</p> <p>For development environments, it is sufficient to accept the defaults. You do not need to know these passphrases.</p>
Security Mode	<p>Security mode used for connections with the Controller.</p> <ul style="list-style-type: none"> • Unsecure — Communication with the Controller uses HTTP protocol. Unsecure mode is sufficient for development environments. • Secure — Communication with the Controller uses SSL protocol. Secure mode ensures confidentiality and integrity of the communication and requires setting up trust as an explicit step between the Controller and the Agent. Oracle recommends that you use Secure mode for production environments. <p>For more information, see “Securing Controller to Agents Communication” on page 7-7.</p> <p>NOTE: Both the Controller and Agent must be set to the same security mode.</p>

5. In the **Configure Agent Logging** window, complete the fields as described in [Table 3-9](#).

Table 3-9 Configure Agent Logging

In this field . . .	Specify the following . . .
Logfile severity	Severity of events to log, default: INFO In the order of severity from least severe to most severe, the log levels are: TRACE, DEBUG, INFO, NOTICE, WARNING, ERROR, CRITICAL, ALERT, EMERGENCY Severity levels are inclusive. When set to INFO, the log will include NOTICE, WARNING, ERROR, CRITICAL, ALERT, and EMERGENCY events.
Stdout severity	Specify the severity level of events that should be written to stdout; default: INFO. Uses same event levels as the log file.
Logfile location	Accept the default or specify a different directory and/or log file name. Default: <code>BEA_HOME\user_projects\agent1\logs\Agent.log</code>

6. In the **Configure Agent Keystore Passwords** window, you are prompted for the Agent keystore passwords used for internal WLOC communications. In most cases, this depends on whether you are using WLOC in a production or development environment, as described in [Table 3-10](#).

Table 3-10 Configure Agent Keystore Passwords

In this environment . . .	Do the following . . .
Development	Click Next to use the default keystore passwords.
Production	Enter the passwords that will be used to secure the keystores used for production-level communications between WLOC components. Make a note of these passwords. They will be required later when setting trust by importing certificates into the keystores. For more information about the WLOC Agent keystores, see “Keystores” on page 7-10 .

7. In the **Configure Agent Type** window, select **ESX Agent** and click **Next**.
8. In the **Configure ESX Agent (1 of 6)** window, complete the fields as described in [Table 3-11](#) and click **Next**.

Table 3-11 Configure ESX Agent (1 of 6)

In this field . . .	Do the following . . .
Name	Accept the default name or enter a different one.
Description	Accept or modify the default.

9. In the **Configure ESX Agent (2 of 6)** window, complete the fields as described in [Table 3-12](#) and click **Next**.

Table 3-12 Configure ESX Agent (2 of 6)

In this field . . .	Do the following . . .
Virtual Center Host	Enter the IP address or name of the VirtualCenter Server.
Username	Enter the user name of a VirtualCenter administrator.
Password/Confirm Password	Specify a password for the administrator.
VMWare SSL Certificate	<p>Select the Connect to the Virtual Center to retrieve SSL certificate check box.</p> <p>When selected, the Virtual Center's public key certificate is obtained and added to the ESX Agent's trust keystore. This is needed to establish trust between an ESX Agent and Virtual Center.</p>
ESX Agent Configuration Type - Dynamic or Static	<p>Select the Configure the ESX Agent dynamically by connecting to the Virtual Center check box if you want the wizard to connect to the Virtual Center host and obtain information for the remaining configuration windows. Otherwise, you will be required to manually enter this information.</p> <p>Accept the default selection of the Use secure connection when connecting to the Virtual Center check box if the Virtual Center has been configured to use secure connections. Clear the check box if the Virtual Center uses only insecure connections.</p>

If select the **Configure the ESX Agent dynamically...** check box, the wizard will attempt to connect to the Virtual Center. If the connection is successful, click **Next** and you will be able to complete subsequent fields using dropdown lists. Otherwise, you are prompted to manually complete the fields.

Note: To connect to Virtual Center and retrieve configuration information using an unsecure connection, the Virtual Center must be set to support access to the SDK using HTTP. Otherwise, a message like the following appears when the connection is attempted:

"Failed to connect to VMware. It appears that the webservices stack is not running on the specified port".

See Virtual Center documentation for more information.

10. In the **Configure ESX Agent (3 of 6)** window, complete the fields as described in [Table 3-13](#) and click **Next**.

Table 3-13 Configure ESX Agent (3 of 6)

In this field . . .	Enter the following . . .
Data Center Name	Name of the Datacenter that contains the resource pool managed by this Agent.
Compute Resource (ESX host or cluster)	The ESX Server host or cluster name.
Resource Pool Name	The Resource Pool containing the LiquidVM instances to be managed by this Agent.
Resource Pool Description	An arbitrary description of the resource pool.

11. In the **Configure ESX Agent (4 of 6)** window, click **Add** and specify the networking information as described in [Table 3-14](#). If the Agent is managing LiquidVM instances on different network segments, you must specify the network settings for each network segment. Then click **Next**.

Table 3-14 Configure ESX Agent (4 of 6)

In this field . . .	Enter the following . . .
Name	<p>The Virtual Machine Port Group to which the LiquidVM instance is assigned.</p> <p>If you use the VMWare Infrastructure client, this can be obtained by displaying the host's Configuration tab and then selecting Networking in the Hardware list.</p> <p>Note: If you are using a cluster of ESX hosts, all hosts must have a Virtual Machine Port Group with some name and the group must be mapped a physical adapter connected to the same physical network.</p>
JVM IP Address	<p>One or more IP addresses reserved for the LiquidVM instances. Specify multiple addresses on the same line separated using a comma (,).</p> <p>Example: 182.343.121.122,182.343.121.123,182.343.121.122</p> <p>A LiquidVM instance will use only one of the IP addresses specified.</p>
Description	An arbitrary description.
Gateway IP Address	<p>The Gateway address used by the LiquidVM instance.</p> <p>The address can be determined from the physical network adapter to which the Virtual Machine Port Group is mapped.</p> <p>If not specified, the LiquidVM instance will use the default gateway based its IP address.</p>
Netmask	<p>The Netmask used by the LiquidVM instance.</p> <p>The Netmask can be determined from the physical network adapter to which the Virtual Machine Port Group is mapped.</p> <p>If not specified, the LiquidVM instance will use the default netmask.</p>
DNS Server Address	<p>The primary and alternate DNS Server used by the ESX Server host. Specify multiple addresses on the same line separated using a comma (,).</p> <p>Example: 10.344.22.86,10.170.43.81</p> <p>The LiquidVM instance cannot use remote DNS lookup if this is not specified.</p>
Domain Name	The domain name used by the LiquidVM instance.

12. In the **Configure ESX Agent (5 of 6)** window, specify each ISO being used by clicking **Add** and completing the fields as described in [Table 3-15](#). Then click **Next**.

Table 3-15 Configure ESX Agent (5 of 6)

In this field . . .	Enter the following . . .
Name	The ISO name.
Description	An arbitrary description.
ISO Software Path	<p>The location of the ISO software, including the datastore. If you selected Configure the ESX Agent dynamically... in step 9, you can browse the ESX Server storage for the specific ISO.</p> <p>To specify it manually, the syntax is [datastore] /path/filename</p> <p>Examples:</p> <pre>[storage1] /wlsve922.iso</pre> <pre>[storage1] /wlsve/wlsve922.iso</pre>
Version	The LVM version (1.0, 1.1, or 1.2).

13. In the **Configure ESX Agent (6 of 6)** window, define each NFS share being used by clicking **Add** and completing the fields as described in [Table 3-16](#). Then click **Next**.

Table 3-16 Configure ESX Agent (6 of 6)

In this field . . .	Enter the following . . .
Name	The NFS share name.

Table 3-16 Configure ESX Agent (6 of 6)

In this field . . .	Enter the following . . .
Description	An arbitrary description.
NFS Software Path	<p>The NFS share path using the following syntax:</p> <pre><ip_address>: <path>,uid=<uid_num>,gid=<gid_num></pre> <p>where</p> <p><ip_address> — IP Address of the NFS share host</p> <p><path> — path to the share</p> <p><uid_num> — userid number</p> <p><gid_num> — group id number</p> <p>Example:</p> <pre>182.34.234.92:\WLOC\shares\domainw,uid=10947,gid=3498</pre>

14. In the **Create Agent** configuration window, click **Create**. Progress messages then appear.
15. When it becomes active, click the **Done** button.

Modifying Agent Configurations

There are two ways to modify an existing Agent configuration:

- Manually edit the Agent’s configuration file. The structure of the XML document as well as the configuration definitions can be obtained by accessing the Agent schema file. This file is `BEA_HOME\WLOC_HOME\schemas\loc-agent.xsd`.
- Use the WLOC Administration Console as described in the [WLOC Administration Console Online Help](#).

Controller Configurations

The following topics describe how to create a controller and modify an existing controller configuration.

Configuring a Controller

Follow these steps to create and configure the WLOC Controller:

Note: Running the Configuration Wizard in console mode is currently not supported.

1. Invoke the Configuration Wizard as described in [Table 3-17](#) and click **Next** in the **Welcome** window.

Table 3-17 Invoking the WLOC Configuration Wizard

On this platform . . .	Perform the following steps . . .
Windows	<ul style="list-style-type: none">• From the Windows Start Menu, choose Start > All Programs > WebLogic Operations Control 10gR3>WLOC Configuration Wizard• From an MS-DOS Command Prompt window, go to the <i>BEA_HOME\WLOC_HOME\common\bin\</i> directory and enter <i>config.cmd</i> at the prompt. In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation, for example <i>C:\bea\wloc_10.3</i>.
UNIX or Linux	<p>Go to the <i>BEA_HOME/WLOC_HOME/common/bin/</i> directory and enter <i>config.sh</i> at the prompt.</p> <p>In this pathname, <i>BEA_HOME</i> is the BEA Home directory and <i>WLOC_HOME</i> is the product installation directory that you selected during the WLOC installation.</p>

2. In the **Welcome** window, click **Next**.
3. In the **Choose Controller or Agent** window, select **Create the Controller or extend the existing Controller for this host** and click **Next**.
4. In the **Enter Controller Directory Location** window, specify the location and click **Next**.
5. In the **Enter Controller Connection Data** window, complete the fields as described in [Table 3-18](#).

Table 3-18 Controller Connection Configuration

In this field . . .	Enter the following . . .
Controller Host	Fully-qualified host name of the Controller machine; example: <i>adminbox.east.example.com</i>
Console port	HTTP port for the WLOC Administration Console; default: 9001

Table 3-18 Controller Connection Configuration

In this field . . .	Enter the following . . .
Console secure port	HTTPS port for the WLOC Administration Console; default: 9002
Console mode	<p>Select one of the following to specify how clients may connect to the Administration Console:</p> <p>Secure—HTTPS only</p> <p>Unsecure—HTTP only</p> <p>Both—Either HTTP or HTTPS</p>
Internal port	Port used by agents for unsecure internal communication with the Controller; default: 9003
Internal Secure Port	Port used by agents for secure internal communication with the Controller; default: 9004
Security mode	<p>Select one of the following to specify the security level to be used for internal communications between WLOC components.</p> <p>Unsecure—use HTTP without SSL and guarantee of message confidentiality and integrity. This is sufficient for development systems.</p> <p>Secure—use HTTPS providing message confidentiality and integrity. This should be used with production systems.</p> <p>NOTES:</p> <ul style="list-style-type: none"> • All Agents must use the same Security mode established on the Controller with which they communicate. • For instructions about configuring production level security, see “Secure Communications” on page 7-4.

6. In the **Configure Controller Logging** window, complete the fields as described in [Table 3-19](#).

Table 3-19 Configure Controller Logging

In this field . . .	Enter the following . . .
Logfile severity	Severity of events to log, default: INFO In the order of severity from least severe to most severe, the log levels are: TRACE, DEBUG, INFO, NOTICE, WARNING, ERROR, CRITICAL, ALERT, EMERGENCY Severity levels are inclusive. When set to INFO, the log will include NOTICE, WARNING, ERROR, CRITICAL, ALERT, and EMERGENCY events.
Stdout severity	Specify the severity level of events that should be written to stdout; default: INFO. Uses same event levels as the log file.
Logfile location	Accept the default location or enter a different directory and log file name. Default: <i>BEA_HOME\user_projects\controller\logs\Controller.log</i>

7. In the first **Configure Controller Notifications (1 of 6)** window, select the **Enable SMTP Notifications** check box if you want to enable SMTP notifications. Then complete the fields as described in [Table 3-20](#).

If you do not want to enable SMTP notifications, click **Next** without completing this window.

Table 3-20 Configure Controller Notifications (1 of 6)

In this field . . .	Enter the following . . .
To email address	E-mail address to which notifications should be sent.
From email address	E-mail address from which notifications should be sent.
SMTP Server	SMTP mail server through which to send notifications.

8. In the **Configure Controller Notifications (2 of 6)** window, select the **Enable JMX Notification** and/or **SNMP** check box if you want to enable one or both of those notification types. If you select SNMP notification, complete the fields as described in [Table 3-21](#).

If you do not want to enable either notification type, click **Next** without completing this window.

Table 3-21 Configure Controller Notifications (2 of 6)

In this field . . .	Enter the following . . .
Agent Host	Host name of the SNMP agent.
Agent Port	Port number of the SNMP agent.
Trap Host	DNS name or IP address of the computer on which the SNMP manager is running; default: localhost
Trap Port	Listening port of the SNMP manager; default: 1642
Trap Type	Select SNMPv1 or SNMPv2; default: SNMPv2

9. In the **Configure Controller Notifications (3 of 6)** window, select the **Enable JMS Notification** check box if you want to enable JMS notification. Then complete the fields as described in [Table 3-22](#).

If you do not want to enable JMS notification, click **Next** without completing this window.

Table 3-22 Configure Controller Notifications (3 of 6)

In this field . . .	Enter the following . . .
Destination JNDI Name	Fully-qualified package and class of the JMS notifier; default: <code>com.bea.adaptive.loc.notification.JMSNotifier</code>
Connection Factory JNDI Name	JNDI connection factory; default: <code>com.bea.adaptive.loc.notification.JMSNotifierQueueConnectionFactory</code>
Initial Factory Class	Fully-qualified package and class of the initial factory; default: <code>org.mom4j.jndi.InitialCtxFactory</code>
Provider URL	JNDI provider URL.
Security Principal	JNDI user name.
Password	JNDI user's password.

10. In the **Configure Agents for this Controller** window, click **Add** and specify the Agents to be managed by this Controller and click **Next**.

To perform this step at another time, click **Next** without specifying any Agents.

Table 3-23 Configure Agents for this Controller

In this field . . .	Enter the following . . .
Name	Agent name.
Hostname	The fully-qualified host name or IP address of the machine hosting the Agent.
Port	HTTP port on which to access the Agent. This was specified when configuring the Agent. Default: 8001
Secure port	HTTPS port on which to access the Agent. This was specified when configuring the Agent. Default: 8002
State	One of Enabled, Connected, or Disconnected. Default: Enabled
Passphrase Confirm Passphrase	The Agent passphrase specified when the Agent was created. Note: Communication between the Controller and Agent will fail unless your entry matches the Agent's current passphrase.

11. In the **Use SSH for WLOC ESX Agents** window, select the **Enable SSH for LVM instances** check box to initialize the LVM instance that will be started by the ESX Agent on the ESX Server with the SSH public key. Then specify the location of the SSH Public Key File and click **Next**.

This allows a SSH client that uses the corresponding private key to be trusted by the LVM instance when the SSH client later connects to the LVM instance using SSH. The Controller will pass the SSH public key to the ESX Agent which in turn will provide the public key to the LVM instance when the LVM instance is initialized by the ESX Agent.

12. In the **Enter User Data** window, accept the default username and password for logging in to the WLOC Administrative Console or overwrite these values as desired and click **Next**.

Notes:

- The default password is `changeit`. This is acceptable for development environments, but not for production.
- If you specify a new password, the value is encrypted before being saved in the XML configuration file.

13. In the **Configure Controller Keystore Passwords** window, you are prompted for the keystore passwords used for internal WLOC communications. The keystores are used if Secure mode is selected for Controller to Agent communications and if Secure or Both are selected for Console communications.

In most cases, this step depends on whether you are using WLOC in a production or development environment, as described in [Table 3-24](#).

Table 3-24 Configure Controller Keystore Passwords

In this environment . . .	Do the following . . .
Development	Click Next to use the default keystore passwords.
Production	Enter the passwords that will be used to secure the keystores used for production-level communications between WLOC components. For more information about the WLOC Controller keystores, see Chapter 7, “Configuring Security.”

14. In the **Create Controller Configuration** window, click **Create**. Progress messages then appear.
15. When it becomes active, click the **Done** button.

Modifying Controller Configurations

You can change the Controller configuration as follows:

- Using the WLOC Administration Console.
- Directly editing the `loc-controller-config.xml` file.
- Re-running the Configuration Wizard. When you use the Configuration Wizard to change the configuration, you can only add or modify information about the Agents connecting to the Controller.

Modifications made using the Administration Console take immediate effect, while changes made by editing the XML file or re-running the Configuration Wizard do not take effect until the Controller is restarted. In addition, no validation or error checking is performed when you directly modify the configuration file.

For information about the elements of the `loc-controller-config.xml`, see the [Controller Configuration Schema Reference](#) or examine the schema file:

`BEA_HOME\WLOC_HOME\schemas\loc-controller.xsd.`

Adding Agents to a Controller

You can add an Agent to a Controller by running the WLOC Configuration Wizard or using the WLOC Administration Console.

To add an Agent to a Controller using the WLOC Configuration Wizard:

1. Start the WLOC Configuration Wizard.
2. Proceed to the **Choose Controller or Agent** window and select **Create the Controller or extend the existing Controller for this host** and click **Next**.
3. In the **Enter Controller Directory Location** window, specify the configuration directory for the existing Controller and click **Next**.
4. For each Agent you wish to add to the Controller's configuration, click **Add** and enter the information listed in [Table 3-23](#) for each Agent. When done, click **Next**.
5. In the **Update Existing Controller Configuration** window, click **Create**. This updates the Controller's `loc-controller-config.xml` file with the new Agent information.

Note: You need to restart the Controller after adding an Agent so that the Controller can find the Agent and connect to it.

Starting and Stopping the Controller and Agents

This section describes how to start and stop WLOC Controllers and Agents.

- [“Starting an Agent” on page 4-1](#)
- [“Starting the Controller” on page 4-2](#)
- [“Accessing the WLOC Administration Console” on page 4-2](#)
- [“Stopping an Agent” on page 4-2](#)
- [“Stopping the Controller” on page 4-3](#)
- [“Running the Controller and Agents as Windows Services” on page 4-3](#)
- [“Unexpected Shutdowns” on page 4-3](#)

Starting an Agent

To start an Agent:

1. Change to the directory where the Agent was created. The default directory is `BEA_HOME\user_projects\agent1`.
2. Open a command window and run `.\bin\startAgent.sh` (UNIX or Linux) or `.\bin\startAgent.cmd` (Windows).

Starting the Controller

To start a Controller:

1. Change to the directory where the Controller was created. The default directory is `BEA_HOME\user_projects\controller`.
2. Open a command window and run `.\bin\startController.sh` (UNIX or Linux) or `.\bin\startController.cmd` (Windows).

Accessing the WLOC Administration Console

The WLOC Administration Console starts when you start the Controller. You can access the console using a web browser.

- For HTTP connection, the URL is: `http://hostname:port/wloc-console`
- For HTTPS, the URL is: `https://hostname:port/wloc-console`

where

hostname is the host name or IP address

port is the port specified when the Controller was configured. The HTTP default is 9001; the HTTPS default is 9002.

The default user name and password for the WLOC Administration Console are **WLOCBootUser** and **changeit**.

Stopping an Agent

To stop an Agent using the Administration Console, do one of the following:

- Select the **Agents** tab at the top of any console page to display a list of Agents. Select the Agent's check box and click **Shutdown**.
- Right-click the name of the Agent in the Inventory pane and select **Shutdown Agent** from the context menu.

As an alternative to shutting down an Agent, you can disconnect it. The Controller cannot deploy services to a disconnected Agent.

Stopping the Controller

To stop the Controller using the Administration Console:

1. Select the **Controller** tab at the top of any console page.
2. In the Controller page, select the **Control** tab.
3. Click **Shutdown Controller**.

Stopping the Controller also shuts down the Administration Console application.

Running the Controller and Agents as Windows Services

WLOC provides scripts that allow you to set up the Controller and Agent(s) to run as Windows services.

Note: To run a WLOC Agent as a Windows service, it must be run on a local disk drive. Running the Agent as a Windows service from a mapped drive is not supported.

To set up a Controller or Agent to run as a Windows service, open a Command Prompt window and enter one of the following commands:

- For the Controller, enter:
`BEA_HOME/user_projects/controller/bin/install_ControllerService.cmd`
- For an Agent, enter:
`BEA_HOME/user_projects/agent/bin/install_AgentService.cmd`

When the script completes, the service will be configured to start automatically upon system boot and run using the Local System account.

You can revert to running the Controller or Agent from a command line by removing the Windows service. Use one of the following commands:

- For the Controller, enter:
`BEA_HOME/user_projects/controller/bin/remove_ControllerService.cmd`
- For an Agent, enter:
`BEA_HOME/user_projects/agent/bin/remove_AgentService.cmd`

Unexpected Shutdowns

The unexpected shutdown of a WLOC Agent does not impact the operation of the Controller. When the Agent is rebooted, it will attempt to reconnect to its managed resources. The Controller

Unexpected Shutdowns

will poll for disconnected Agents periodically (once for every `heartbeat-interval` configured in `loc-controller-config.xml`) and re-establish the connection with the Agent when it becomes available.

If the Controller shuts down unexpectedly, the Agents to which it connects will continue to collect and locally store information from its managed resources. Once the Controller restarts and re-establishes connections with the Agents, they will then send that information to the Controller.

Configuring Services and Process Groups

This section describes configuration tasks associated with WLOC services. It contains the following sections:

- [“Overview” on page 5-1](#)
- [“Configuring a Service” on page 5-3](#)
- [“Configuring a Process Group” on page 5-4](#)
- [“Resource Requirements” on page 5-7](#)
- [“Ready Metrics for Processes” on page 5-8](#)
- [“JVM Arguments for Processes” on page 5-10](#)
- [“Creating Multiple Java Processes Using Templates” on page 5-14](#)
- [“Creating Multiple Java Processes Using Templates” on page 5-14](#)
- [“Service Deployment Process” on page 5-19](#)

Overview

A WLOC service is a collection of one or more processes that WLOC manages as a unit. Each process in a service is a software stack starting from the Java Virtual machine (JVM) and including the classes that are running in the JVM. You organize processes (JVMs) that perform the same function and have the same runtime characteristics into *process groups* within the

service. For example, you can organize all of the Managed Servers in a cluster within a process group.

The configuration information for a service contains the following types of information:

- Constraints on the number of process instances. Initial process constraints specify how many processes of each type are created when the service is deployed. Once the service is deployed, WLOC uses process constraints to determine the boundary requirements when a policy leads to an action that starts or stops instances. For example, if a policy leads to an action that would otherwise start a new instance of a process group, a maximum process constraint will stop the creation of a new process instance if you have already reached the maximum number of allowed processes. Similarly if a policy leads to an action that would otherwise destroy an instance, a minimum process constraint will prevent process destruction if you are already at the minimum requirement for the process group.
- Information that WLOC needs to start the processes. For example, a process group can specify Java classes or other executables along with startup parameters such as the names of WebLogic Server instances.
- Requirements for the physical computing resources that are needed to start and run each process. These resource requirements are expressed as a range of CPU cycles, memory, and IP addresses.
- Requirements for access to the collection of software and systems that are used to run the applications.
- An optional set of service policies which define a service level agreement (SLA) and actions to take when the service is operating outside of the SLA constraints.

For example, you can create a service that specifies the following:

- Java commands for starting three Managed Servers (MS1, MS2, and MS3) and an Admin Server, each of which belongs to a WebLogic Server cluster.
- A set of resource requirements that indicate that the service needs a total of at least 400 MHz of CPU cycles and 400 MB of RAM, and that the service can scale up to using no more than 800 MHz of CPU and 800 MB of RAM.
- The list of software that is needed in order for all the processes in the service to run.
- A set of policies that start the service with only 2 Managed Servers running and start the third Managed Server if a specific servlet's response time drops below 2 seconds. The policies can also require that the third Managed Server shuts down if the servlet's response time is faster than one second.

Service Process Management

Some of the types of processes that WLOC manages provide their own set of management tools. For example, WebLogic Server provides an Administration Console that creates and configures servers and that deploys applications. While you can use these external management tools to configure processes that are running as WLOC services, note the following restrictions:

- WLOC can control only those processes that you configure and deploy as part of a service.
- WLOC can monitor only those processes and classes for which you have created service policies.

For example, you create a WLOC service with a process group consisting of 3 WebLogic Managed Servers that are in a cluster. This cluster of Managed Servers hosts a collection of Web Services. If you use WebLogic Server utilities to add a new Managed Server (MS4) to the cluster, MS4 can function as a cluster member, but WLOC cannot manage MS4 until you add it to the service configuration. Similarly, if you use WebLogic Server utilities to deploy a new Web Service to the cluster, users can access the Web Service and WLOC can measure the additional use of memory and other computing resources that result from adding the new Web Service, but WLOC cannot monitor specific activity within the new Web Service until you add one or more service policies to the WLOC service configuration.

Configuring a Service

The primary tool for defining services is the WLOC Administration Console. Services you define or modify using the console will take place in real time. For detailed instructions about using the Administration Console to create services, see the [WLOC Administration Console Online Help](#). You can access the Help by clicking the Help link in the upper right corner of the Administration Console.

When defining service processes in the console, you can manually enter the required information or use a number of helper functions the console provides for importing much of the information. These include:

- importing a running WebLogic domain
- importing a domain's configuration file
- importing from an existing service

It is also possible to define or modify a service by directly editing the metadata configuration file. The location and name of this file is:

`BEA_HOME\user_projects\controller\config\metadata-config.xml`

Note that direct modifications to the file do not take effect until the Controller is restarted. In addition, manual changes are not validated or error-checked until they are loaded by the next Controller boot.

There are number of resources to help you obtain more information about `metadata-config.xml` file. These include:

- The metadata schema file. The name and location of this file is `BEA_HOME\WLOC_HOME\schemas\loc-metadata.xsd`. This file may also be accessed at http://download.oracle.com/docs/cd/E13156_01/wloc/docs103/schemaref/metadata/loc/-metadata.xsd.
- The *Service Metadata Configuration Schema Reference*.
- The sample metadata configuration file provided when WLOC is installed. The name and location of this file is:
`BEA_HOME\user_projects\controller\config\metadata-config.xml.SAMPLE`.

Configuring a Process Group

Note: Process groups are referred to as process types in the service `metadata-config.XML` file.

When configuring a process group, you need to specify information required by WLOC to instantiate it, including:

- the number of processes that the service should run
- the class or JAR file that instantiates the process
- JVM startup arguments
- a Ready Metric to determine whether the process is available

When defining the process group, the console prompts for information about its processes. [Table 5-1](#) describes in more detail the information needed to define a process group.

Table 5-1 Process Group Configuration

Process Group Attribute	Description
Process Group	A string identifier for the process group.
Number of Processes	Number of processes that are configured for the process group when it is initially created.
Name	Name of the initial JVM instance. For each additional instance, a numeric suffix is added to the name.
Description	Description of the JVM.
Main Class or Main JAR	Class or JAR file that instantiates this process.
Host	Fully-qualified host name where the JVM resides. The host and port number are used to determine the address the Agent uses to collect JMX metric information from the endpoint.
Starting Port #	Starting port number for the machine where the JVMs reside. The first process instance uses this port number. For each additional instance, the port address is incremented by 1. The host and port number are used to determine the address the Agent uses to collect JMX metric information from the endpoint. For more information, see “Creating Multiple Java Processes Using Templates” on page 5-14 .
JMX Service URL	JMX service URL to use instead of specifying a WLS host and port. This URL is used to connect to a non-WLS endpoint, such as any MBean server.
Classpath	CLASSPATH for the class or JAR file that instantiates this process. For information about obtaining CLASSPATH information, see “Determining the CLASSPATH” on page 5-13 .
JVM Arguments	JVM arguments to use with the command that instantiates the process. These arguments are passed as Java options. Separate each Java argument using a space. For more detailed information, see “JVM Arguments for Processes” on page 5-10 .

Table 5-1 Process Group Configuration (Continued)

Process Group Attribute	Description
Java Arguments	Command line arguments needed for running the application in this JVM. These are passed to the Java main class as arguments.
Username and password	Security credentials required to make a JMX connection to the JVM. These will be the default values for all subsequently created processes.
Instance Directory	Working directory in which the process instance is started.
Native Lib Directory	Directory in which the JVM native libraries reside. The value will be passed to jvm args as <code>-Djava.library.path</code>
Use Native JMX	Flag that specifies whether or not to use JMX connection native JMX MBean server.
SSH Enabled	Flag that specifies whether or not SSH is enabled for the instance.
Protocol	Protocol used for JMX connections (<code>iiop</code> , <code>iiops</code> , <code>http</code> , and <code>https</code>).
Ready metric	<p>Information about the metric that WLOC obtains from the JVM instance and uses to determine whether or not a process is available.</p> <p>For more detailed information, see “Ready Metrics for Processes” on page 5-8.</p>
Process Requirements	The required minimum and maximum number of processes.
Resource Requirements	The required minimum and maximum amount of CPU and memory. For more information, see “Resource Requirements” on page 5-7 .
Software Requirements	The name and directory location of the software required by the process.
Manual Placement Addresses	Suggested IP addresses for placement of the process. Doing this supplements WLOC’s internal placement algorithm, which searches for the set of placement locations that would work given the constraints of memory, CPU capacity, software availability, and IP addresses.

Resource Requirements

When using the console to define a service's processes, the console solicits information about the service's resource requirements. This information constitutes the service's out-of-gate deployment policies and is saved in the form of constraints and actions.

Resource requirements specify the required amount of physical computing resources for each instance in a process group. For example, to create a process group that includes a WebLogic Server domain, specify resource requirements for running all WebLogic Server instances. You can specify that a WebLogic cluster needs a minimum of 400 MHz of CPU cycles and 400 MB of RAM and can scale up to a maximum of 800 MHz of CPU and 800 MB of RAM.

Note: For CPU measurement, WLOC uses normalized megahertz (MHz) across CPU architectures so that a megahertz of processing on an i386 processor is comparable to a megahertz on other types of processors.

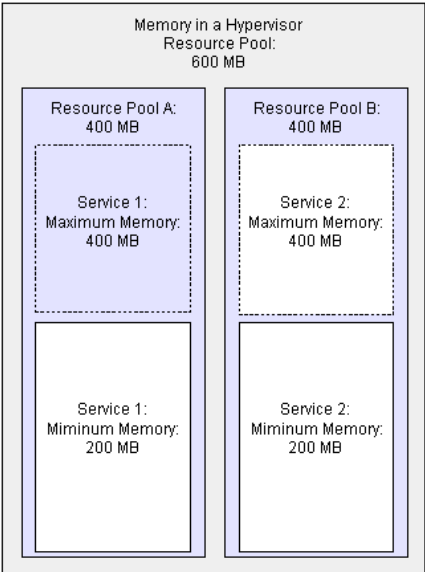
These requirements do not include resources needed to run software outside of the process group, but declared as a dependency. For example, if a process group specifies that it requires a RDBMS system, the resource requirements do not include computing resources for the RDBMS.

Minimums and Maximums

Resource requirements are specified in terms of minimum and maximum amounts. The minimum requirement specifies the smallest amount of resources that must be reserved for exclusive use by a process group. As the resources consumed by a process group approach the minimum, a WLOC policy can request additional resources up to the maximum requirement.

WLOC may not be able to actually obtain additional resources if all available resources are being consumed by other WLOC resource pools or non-WLOC applications. For example, [Figure 5-1](#) shows a hypervisor resource pool containing two WLOC resource pools. Although the pool provides 600 MB of memory, the WLOC Resource Pool has two services, each configured for a minimum of 200 MB and maximum of 400 MB. While both services can consume the minimum required memory (200 MB) at the same time, they cannot simultaneously consume the maximum. In [Figure 5-1](#), service 1 is consuming 400 MB, so the total amount of consumed memory is 600 MB. Service 2 cannot use additional memory until service 1 decreases its memory consumption.

Figure 5-1 Resource Minimums and Maximums



Ready Metrics for Processes

WLOC needs to know when a process has successfully started and is ready for work. This is made possible by specifying the process group’s ready metric. When defined, the action to start an instance is not considered complete until the ready metric is satisfied. If the ready metric isn't satisfied in a configurable amount of time, then the instance is considered failed and will be stopped.

Ready metric information is described in [Table 5-2](#).

Table 5-2 Ready Metric

Ready Metric Attribute	Description
Instance Name	The name of the MBean that has the attribute to test. Example: <code>com.bea:Name=AdminServer,Type=ServerRuntime</code>
Attribute	The name of the MBean attribute to test. Example: <code>state</code> . For more information about the format to use to specify MBean attributes, see WebLogic Server MBean Reference .

Table 5-2 Ready Metric

Ready Metric Attribute	Description
Value	A constant value to test the MBean attribute against. This field is valid only if the Operator is set to Value Equals. For example: RUNNING.
Operator	The operator (Value Equals or Value Exists).
Value Type	The value type (Integer, Long, Float, Double, Boolean, Date, String). The default is Integer.
Wait	(Optional) The number of milliseconds to wait after the ready metric constraint is satisfied before the process is considered ready.

[Table 5-3](#) specifies a ready metric that reports a JVM instance is in RUNNING state:

Table 5-3 WebLogic Server Instance Ready Metric Configuration

Ready Metric Attribute	Value
Ready metric instance	Name=com.bea:AdminServer,Type=ServerRuntime
Attribute	State
Value	RUNNING
Operator	ValueEquals
Value Type	String
Wait	30000

[Listing 5-1](#) shows how this ready metric as contained in metadata-config.xml.

Listing 5-1 WebLogic Server Instance Ready Metric Configuration

```
<ns2:ready-information>
  <ns2:check-type>ValueEquals</ns2:check-type>
  <ns2:max-wait-period>300000</ns2:max-wait-period>
  <ns2:instance>com.bea:Name=AdminServer,Type=ServerRuntime</ns2:instance>
```

```
<ns2:attribute>State</ns2:attribute>
<ns2:value>RUNNING</ns2:value>
<ns2:value-type>java.lang.String</ns2:value-type>
</ns2:ready-information>
```

JVM Arguments for Processes

For proper operation and tuning, managed applications must typically start with a specific set of JVM arguments. For each process group, you can specify the JVM arguments to use when the process starts. In the WLOC Administration Console, you specify JVM arguments on the **Process Properties** page when you define the instance.

Note: JVM arguments are set when the JVM instance is staged. If you update the JVM arguments after the JVM instance is staged, the settings will not be updated from the configuration unless the JVM is destroyed and restaged. To use the new settings, you have to unstage the service and restart it.

When configuring a JVM instance, you need to specify connection information to tell the Agent how to get the management information from the running instance. For more information, see [“Setting the JMX Port” on page 5-13](#). You also need to configure the JVM instance itself to open the port to which the Agent needs to connect.

Typically, the native configuration for the JVM instance will specify how it exposes its management information. In some cases, however, you may need to specify command line arguments for the instance. For example, with WebLogic Server, the listen port can be defined in the `config.xml` for the WebLogic domain or it can be overridden on the command line. For non-WebLogic endpoints, you may want to enable remote access to the JDK platform MBeanServer by setting command line arguments.

- For JVMs running on JRockit before version 150_11, use the following single argument:

```
-Xmanagement-Djrockit.managementserver.port=<port-number>
```

- For JVMs running on JRockit version 150_11 and higher, use the following two arguments:

```
-Djrockit.managementserver.port=<port-number>
```

```
-Xmanagement:ssl=false,authenticate=false
```

Note: The WLOC Plain Agent uses the JDK platform MBeans to obtain the current CPU and memory use of a service. For Plain Agents, you must be sure that the platform MBeans are available from the exposed port.

Setting Communication Protocols

Although all of the management information for a managed endpoint is obtained through JMX MBeans, the communication protocols over which that information is passed can vary depending on the application running in the VM. Here are some examples:

WebLogic

When the endpoint is WLS, the WebLogic server exposes JMX over its general listen port (default 7001). Note that in addition to WLS MBeans, the WLS MBeanServer also exposes the JDK platform MBeans.

Coherence

When the endpoint is Coherence, the port for exposing Coherence MBeans is configured by setting command line properties starting with `-Dtangosol.coherence`. Note that best practices for Coherence are to set up a small set of instances within the cluster to expose JMX data; these instances aggregate information about all cluster members. As mentioned above, the Plain Agent uses the JDK Platform MBeans to obtain CPU and Memory statistics. Therefore, certain charts in the WLOC Administration Console will not be able to obtain default CPU and memory statistics for each instance with this setup. For more information on exposing MBeans from Coherence, see the Coherence documentation.

Generic Application

If the application is not WebLogic or Coherence, you can expose management information from the platform MBeanServer via a port that is opened by the JVM itself. This is configured by setting command line properties starting with `-Dcom.sun.management`

Although all three of the above examples are providing JMX data, the underlying protocols are all different and the JMX client, in this case the Agent, needs to know how to read the data. JMX provides quite a bit of flexibility in how the JMX information is transported and serialized on the wire. In the case of WLS, various protocols may be used, for example `http`, `https`, `t3`, `t3s`, etc.

If you only set protocol, host and port, the WLOC agent will assume that it is interacting with a WebLogic endpoint and will use WebLogic RMI to transport the data over the protocol you specify.

If the endpoint is not WLS, then you should set `native-jmx` to `true` in the `metadata-config.xml` file. In the WLOC Administration Console, select the **Use Native JMX** check box on the **Process Properties** page when you define the instance. When you do so, the WLOC agent will use standard JMX protocols.

Coherence uses two ports to expose JMX. One port is for the JMX traffic itself and the other is for the JNDI information needed to lookup JMX. The basic protocol, host and port settings are

not sufficient to connect to this endpoint, so when interacting with the Coherence MBeanServer, you must specify the `jmx-service-url` in the `metadata-config.xml` file. In the WLOC Administration Console, you specify this value on the **Process Properties** page when you define the instance. When `jmx-service-url` is specified, the protocol, host and port settings will be ignored.

[Listing 5-2](#) shows the JVM arguments used to start a WLS Administration Server.

Listing 5-2 JVM Arguments Example

```
-Xmanagement -Dcom.sun.management.jmxremote.port=7091
-Xmx128m
-Xms64m
-Dweblogic.Name=examplesServer
-Dweblogic.management.username=weblogic
-Dweblogic.management.password={Salted-3DES}L9NHXoCmDmOXAobBz2ennw==</
-Djava.security.policy=C:/files/bea/weblogic92/server/lib/weblogic.policy
-Dwls.home=C:/files/bea/weblogic92/server
-Dweblogic.RootDirectory=C:\files\bea\weblogic92\samples\domains\wl_server
```

[Listing 5-3](#) provides an example of the JVM arguments as contained in `metadata-config.xml`.

Listing 5-3 JVM Arguments in Metadata-Config.xml for a WLS instance

```
<ns2:jvm-args>
  <ns2:arg>-Xmx128m</ns2:arg>
  <ns2:arg>-Xms64m</ns2:arg>
  <ns2:arg>-da</ns2:arg>
  <ns2:arg>-cp</ns2:arg>
  <ns2:arg>CLASSPATH=C:\bea\patch_weblogic921\profiles\default\
sys_manifest_classpath\weblogic_patch.jar;C:\bea\WEBLOG~1\server\
lib\weblogic_sp.jar;C:\bea\WEBLOG~1\server\lib\weblogic.jar;C:\bea\
WEBLOG~1\server\lib\webservices.jar;</ns2:arg>
  <ns2:arg>-Dwls.home=C:\bea\weblogic92\server</ns2:arg>
  <ns2:arg>-Dweblogic.management.discover=true</ns2:arg>
  <ns2:arg>-Dweblogic.Name=AdminServer</ns2:arg>
  <ns2:arg>-Dweblogic.management.username=weblogic</ns2:arg>
  <ns2:arg>-Dweblogic.management.password=weblogic</ns2:arg>
```

```

<ns2:arg>-Djava.security.policy=C:\bea\weblogic92\server\lib\
weblogic.policy</ns2:arg>
<ns2:arg>-Dweblogic.RootDirectory=C:\bea\user_projects\domains\
WLOCdomain</ns2:arg>
</ns2:jvm-args>

```

For additional information about JVM arguments to use when starting WebLogic Server instances, see the [weblogic.Server Command-Line Reference](#).

Setting the JMX Port

You set the JMX port of the managed endpoint using the **Starting Port #** field in the Administration Console, or the `<port>` field of the `<jvm-instance>` definition in the `metadata-config.xml` file. The host and port number are used to determine the address the Agent uses to collect JMX metric information from the endpoint.

The following rules apply to the JMX port setting:

- For WLS endpoints

The WLS listen port, by default, becomes the JMX port. However, if the WLS Administration Port is enabled, the Administration Port should be used as JMX port. For more information, see “[Configure the domain-wide administration port](#)” in the *WLS Administration Console Online Help*.

- For generic Java endpoints

Enable the JMX port using the JVM specific method for your platform as follows:

- JRockit JVM: Use `-Xmanagement` as described in “[-X Command-line Options](#)” in the *JRockit Command Line Reference*
- Sun JVM: Refer to the Sun JMX documentation at <http://java.sun.com/javase/6/docs/technotes/guides/management/agent.html>

You can also use `-Dcom.sun.management.jmxremote.port=<port_num>` to set the JMX port.

Determining the CLASSPATH

One of the required JVM arguments is the CLASSPATH needed to start the JVM instance. This section describes how to obtain the CLASSPATH in different environments.

JVMs on Plain Agents

- For WebLogic Server, execute the `setDomainEnv` command and obtain the WLS CLASSPATH by echoing CLASSPATH environment variable. Examples:

Windows — `\bin\setDomainEnv.cmd`
Solaris/Linux — `./bin/setDomainEnv.sh`

- For generic Java applications, set the classpath based on application installation on Plain Agent host. Examples:

Windows — `c:\apps\myapp\lib\myapp1.jar;c:\apps\myapp\lib\myapp2.jar`
Solaris/Linux — `/apps/myapp/lib/myapp1.jar;c:/apps/myapp/lib/myapp2.jar`

LiquidVMs

CLASSPATH elements can be sourced anywhere from the LiquidVM file system. If the element is in the ISO image, the path must begin with `/appliance` since the ISO is mounted on `/appliance`. If the element is in an NFS share, then the path must begin with the NFS mount point. If the element is in the local disk, then the path would begin with `/` (forward slash).

For WLS-VE 9.2v1.1, the standard classpath as set by the `commonStartVE` script is:

```
/bea/patch_weblogic922/profiles/default/sys_manifest_classpath/weblogic_patch.jar:  
/appliance/java/lib/tools.jar:/appliance/bea/weblogic92/server/lib/weblogic.jar:  
/appliance/bea/weblogic92/server/lib/webservices.jar:
```

Creating Multiple Java Processes Using Templates

To simplify the task of creating multiple copies of Java processes in a service, WLOC provides the capability to define the common characteristics of the process in a template, and to use variables to replace common parameters. You can then use the template to generate multiple copies of the process without having to physically configure duplicates in the metadata configuration. When you create the template, you define:

- The common parameters that apply to all instances of the process
- The number of copies of the process that you want to create
- A set of variables that will be used to replace references in the template

Note: The number of values that you specify in the variables must be equal to the number of copies to be created.

Defining the Common JVM Parameters

You define the common parameters of the Java process template in the same way in which you define an individual process instance. If you know the name of the variables that you will define, you can specify them as part of your arguments when you configure the template. Otherwise, you will need to update the template definition after the variables are created. For more information, see the following topics:

- [“Configuring a Process Group” on page 5-4](#)
- [“Define process groups for a service”](#) in the *WLOC Administration Console Online Help*

Specifying the Number of Processes

In addition to the variable definitions, you can specify the number of copies of the instance that you want to create by using the `max-copies` element:

```
<ns2:max-copies>5</ns2:max-copies>
```

The default is 1.

Note: In the Administration Console, you specify the number of copies using the **Max Copies** field on the **Process Group Template Properties** page.

For example, when you specify 5 as the value for the `max-copies` element, WLOC expands the definitions into five `JavaInstanceMetaData` instances.

When `max-copies` is greater than 1, the name of each Java instance will be appended with a dash followed by the number of the copy (0,1,2,...), such as `managed-0`, `managed-1`, and so on.

Defining Variables

To take advantage of the variable definitions, you can insert variable references into certain attributes of the `jvm-instance` definition. You can specify variables for the following attributes:

- Java arguments `<JavaArgs>`
- JVM arguments `<JvmArgs>`
- Instance directory `<InstanceDir>`
- Host `<Host>`

- Port <Port>
- JMX Service URL <JmxServiceUrl>

You can also define variables for the Instance and Value attributes of the Ready Metric definition.

Two types of variables are supported: scalar and arrays. If you use a scalar variable, then its value replaces the reference in the attribute. If you use an array variable, you can specify a set of values to use to replace the reference in the attribute.

For example, you can create a variable named `ManagedPort` that you can use to define a different port value for each instance as follows:

```
<ns2:port>${ManagedPort}</ns2:port>
```

To use a scalar variable, you simply specify the value to be substituted for the reference. For example, a scalar value in the metadata file is shown as:

```
<ns2:variable>
  <ns2:name>ManagedPort</ns2:name>
  <ns2:value>7010</ns2:value>
</ns2:variable>
```

You can specify array variables as follows:

- As a single value to specify that all instances use the same value, for example `${port}=7010`.
- In a comma-separated list, for example `${ManagedPort}=7010, 7011, 7012,7013`.
- As a range of values, for example `${ManagedPort}=@range{7010,7013}`.

An array using a list or range of values is shown in the metadata file as:

```
<ns2:variable>
  <ns2:name>ManagedPort</ns2:name>
  <ns2:values>
    <ns2:value>7010</ns2:value>
    <ns2:value>7011</ns2:value>
    <ns2:value>7012</ns2:value>
    <ns2:value>7013</ns2:value>
  </ns2:values>
</ns2:variable>
```

In this example, the first copy will be assigned port number 7010, the second 7011, and so on.

There is an implicit variable `${index}` which can be used to insert the array index somewhere in your declaration, for example

```
<ns2:arg>-Dweblogic.Name=managed-${index}</ns2:arg>
```

For instructions about creating variables in the Administration Console, see [“Defining variables for multiple Java processes”](#) in the *WLOC Administration Console Online Help*.

Sample Template Using Variables

[Listing 5-4](#) provides a sample JVM template using variables. The template specifies that 5 copies of the instance should be created, and uses variables for the name, host, and port fields. The variables and the variable references are shown in bold text.

Listing 5-4 Sample JVM Template Using Variables

```
<ns2:jvm-instance>
  <ns2:name>MyServerInstance</ns2:name>
  <ns2:description>This JVM instance defines a template to stamp out
five instances of MyServer</ns2:description>
  <ns2:max-copies>5</ns2:max-copies>
  <ns2:variables>
    <ns2:variable>
      <ns2:name>serverName</ns2:name>
      <ns2:values>
        <ns2:value>server-A</ns2:value>
        <ns2:value>server-B</ns2:value>
        <ns2:value>server-C</ns2:value>
        <ns2:value>server-D</ns2:value>
        <ns2:value>server-E</ns2:value>
      </ns2:values>
    </ns2:variable>
    <ns2:variable>
      <ns2:name>host</ns2:name>
      <ns2:values>
        <ns2:value>123.12.123.100</ns2:value>
        <ns2:value>123.12.123.107</ns2:value>
        <ns2:value>123.12.123.104</ns2:value>
        <ns2:value>123.12.123.109</ns2:value>
```

```
        <ns2:value>123.12.123.111</ns2:value>
    </ns2:values>
</ns2:variable>
<ns2:variable>
    <ns2:name>port</ns2:name>
    <ns2:value>@range{8000,8005}</ns2:value>
</ns2:variable>
</ns2:variables>
<ns2:main-class>com.mycompany.MyServer</ns2:main-class>
<ns2:jvm-args>
    <ns2:arg>-Xms64m</ns2:arg>
    <ns2:arg>-Xmx128m</ns2:arg>
    <ns2:arg>-cp</ns2:arg>
    <ns2:arg></ns2:arg>
    <ns2:arg>-Dcom.mycompany.servername=${serverName}</ns2:arg>
    <ns2:arg>-Dcom.mycompany.maxthreads=8</ns2:arg>
    <ns2:arg>-Dcom.mycompany.minthreads=3</ns2:arg>
</ns2:jvm-args>
<ns2:java-args/>
<ns2:native-lib-dir></ns2:native-lib-dir>
<ns2:instance-dir></ns2:instance-dir>
<ns2:native-jmx>false</ns2:native-jmx>
<ns2:protocol>http</ns2:protocol>
<ns2:host>${host}</ns2:host>
<ns2:port>${port}</ns2:port>
<ns2:username></ns2:username>
<ns2:ssh-enabled>false</ns2:ssh-enabled>
<ns2:wait-for-ssh>false</ns2:wait-for-ssh>
<ns2:priority>0</ns2:priority>
<ns2:copies-at-create/>
<ns2:copies-at-shutdown/>
</ns2:jvm-instance>
```

Service Deployment Process

To start a WLOC service, you deploy it using the WLOC Administration Console. Before you start the service, the state of the service is undeployed, which indicates that there are no instances running. When you start the service, the Controller performs the following steps:

1. Evaluates the process requirements of each process group in the service.
2. Compares the resource pools available for each process group in the service against the resource agreements and eliminates any resource pools that cannot host the service. For example:
 - If the service requires high availability, resource pools indicating that they do not support high availability are eliminated as candidates.
 - If the service requires a specific IP Address, resource pools that do not contain that IP address are eliminated as candidates.
 - If the service specifies software requirements (`iso-constraint` and `software-constraint`), resource pools that do not offer access to all of the required software are eliminated as candidates.
 - If the service consists of a single process, resource pools that offer fewer computing resources than the service's minimum resource requirements are eliminated.

For example, if a resource pool offers 150 MHz of CPU cycles and a service contains one process and requires at least 200 MHz (and up to 400 MHz), the resource pool is eliminated as a candidate. Note, however, that if a resource pool offers 200 MHz of CPU cycles, the resource pool is considered a candidate for deployment.

 - If the service consists of multiple processes, WLOC might choose multiple resource pools to run the service.
3. After the process of elimination, the Controller uses the placement algorithm specified when the service was defined to determine the resource pool on which to place the service:
 - **Prefer resource pools with the most resources:** WLOC selects the resource pool combination that provides the greatest amount of computing resources.
 - **Prefer resource pools with fewer resources:** WLOC selects the resource pool that most closely matches the minimum resource requirements of the service. This algorithm ensures the most efficient use of resources in your data center.

Note: The placement criteria is the same for both plain and ESX instances with the exception of the `iso-constraint`, which is not applicable to plain instances.

Service Deployment Process

4. Stages each instance individually.
5. Starts each instance individually. When the minimum number of processes from each group is started, the service is deployed.
6. Evaluates the runtime policy and initiates the specified actions when a constraint is violated.
7. Continually evaluates the runtime policy using information it obtains from the managed application.

Defining Policies

This section describes how to create policies for deploying and managing applications through WLOC. It includes the following topics:

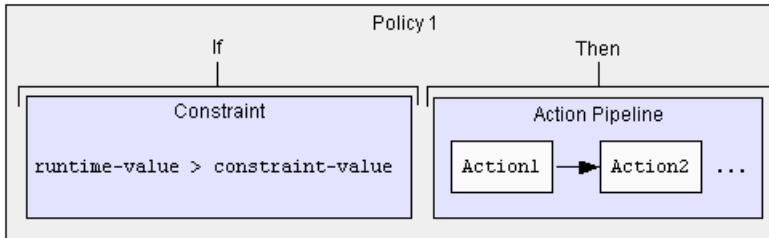
- [“Policy Components” on page 6-1](#)
- [“Policy Types” on page 6-4](#)
- [“Creating Policies” on page 6-3](#)
- [“Constraint Types” on page 6-6](#)
- [“Action Types” on page 6-11](#)
- [“Action Pipelines” on page 6-13](#)

Policy Components

Each WLOC policy consists of a constraint and an action or action pipeline. The constraint defines some deployment or runtime requirement of a service, while the action or action pipeline defines one or more actions to take if the service runtime-value is outside the constraint-value range.

For example, a policy can consist of a constraint that defines a runtime requirement of at least two running processes and an action that starts a process if only one is found to be running.

[Figure 6-1](#) illustrates the basic constraint/action relationship.

Figure 6-1 Policy Constraint and Actions

The assignment of a constraint and action to a policy is referred to as the ‘constraint/action binding’.

[Listing 6-1](#) shows a constraint/action binding as part of a service definition in the `metadata-config.xml` file. In the example, the ‘key’ names are references to the complete constraint and actions definitions located elsewhere in the file

Listing 6-1 Constraint/Action Binding in metadata-config.xml

```
<ns2:service>
  <ns2:name>MyService</ns2:name>
  <ns2:description>MyService</ns2:description>
  <ns2:state>deployed</ns2:state>
  <ns2:priority>0</ns2:priority>
  <ns2:constraint-bindings>
    <ns2:constraint-binding>
      <ns2:constraint-key>MyServicedeploy-key</ns2:constraint-key>
      <ns2:action-key>MyServicestartserviceaction</ns2:action-key>
    </ns2:constraint-binding>
  </ns2:constraint-bindings>
</ns2:service>
```

It is very important that the constraint/action binding be appropriate to the scope of a policy. For example, a constraint that specifies some service deployment state must be bound to an action that can stage, deploy, or undeploy a service.

Creating Policies

To create a policy, you:

1. Create a constraint that defines some service deployment state or runtime performance requirement.

Note: The WLOC Administration Console uses the term *Rules* to refer to constraints.

2. Create an action or action pipeline that specifies the action(s) to take if the application runtime does not satisfy the constraint definition. Doing this in the Administration Console effectively binds the action (or action pipeline) with the constraint.

The recommended method of defining a policy is to use the WLOC Administration Console, but you may also directly edit the metadata configuration file (`metadata-config.xml`). When doing the latter, you must provide three definitions: the constraint binding under the service or process type, the constraint, and the action.

[Listing 6-2](#) provides an XML snippet of a constraint binding. The constraint binding is contained within the service definition as shown in [Listing 6-1](#). Note that the constraint binding contains ‘keys’ to the actual constraint and action definitions elsewhere in the file.

Listing 6-2 Constraint/Action Binding

```
<ns2:constraint-binding>
  <ns2:constraint-key>MyServicedeploy-key</ns2:constraint-key>
  <ns2:action-key>MyServicestartserviceaction</ns2:action-key>
</ns2:constraint-binding>
</ns2:constraint-bindings>
```

[Listing 6-3](#) provides an XML snippet of the constraint definition referenced in the constraint binding shown above.

Listing 6-3 Constraint Definition

```
<ns2:deployment-state-constraint>
  <ns2:name>MyService-service-deploy</ns2:name>
  <ns2:key>MyServicedeploy-key</ns2:key>
```

```
<ns2:priority>0</ns2:priority>
<ns2:state>starting</ns2:state>
<ns2:evaluation-period>0</ns2:evaluation-period>
</ns2:deployment-state-constraint>
```

[Listing 6-4](#) provides an XML snippet of the action definition referenced in the constraint/action binding shown in [Listing 6-2](#).

Listing 6-4 Action Definition

```
<ns2:actions>
  <ns2:action>
    <ns2:name>MyServicestartserviceaction</ns2:name>
    <ns2:key>MyServicestartserviceaction</ns2:key>

    <ns2:impl-class>com.bea.adaptive.actions.internal.StartServiceAction</ns2:
impl-class>
    <ns2:adjudicate>false</ns2:adjudicate>
    <ns2:properties/>
  </ns2:action>
</ns2:actions>
```

For assistance in understanding how to edit the metadata configuration file, use the schema file (*BEA_HOME*\wloc_10.3\schemas\loc-metadata.xsd) and also refer to the [Service Metadata Configuration Schema Reference](#).

Policy Types

To begin identifying the policies needed to ensure that the managed application deploys and performs as required, it may be useful to categorize policies by purpose or functionality. From this standpoint, policies can be regarded as deployment, runtime, or administrative.

These policy types are discussed in the following sections:

- [“Deployment Policies” on page 6-5](#)

- [“Runtime Policies” on page 6-5](#)
- [“Administrative Policies” on page 6-6](#)

Deployment Policies

A deployment policy defines a desired deployment state (deployed, staged, undeployed) of a managed application and the action(s) to take to if the application does not currently satisfy the desired state. For example, a deployment policy may require a service deployment state of ‘deployed’. If the current service state is not deployed, WLOC will deploy the service.

[Table 6-1](#) indicates the types of constraints and actions that can be used to define deployment policies.

Table 6-1 Deployment Policy Constraints and Actions

Constraint Types	Action Types
Deployment	Service
Resource Agreement	Notification
For more information, see “Constraint Types” on page 6-6 .	For more information, see “Action Types” on page 6-11 .

Note: Resource Agreement constraints are not bound to an action.

Runtime Policies

Runtime policies are used to adjust the ongoing performance of a managed application and make adaptive runtime adjustments as necessary. Examples:

- A policy may define the minimum number of running JVM processes. If a JVM instance fails for some reason and the number of running instances falls below the minimum, WLOC will start an additional JVM instance.
- A policy may specify the minimum heap size required and instantiate additional JVM instances if the heap size falls below the minimum required.
- A policy may start additional JVM instances if the workload on the application increases, and then stop instances when the workload decreases.
- A policy may start additional JVM instances based on known times of increased demand.

[Table 6-2](#) indicates the constraints and actions that can be used to define a runtime policy.

Table 6-2 Runtime Policy Constraints and Actions

Constraints Types	Action Types
Runtime	Process
Resource Agreement	Resource
For more information, see “Constraint Types” on page 6-6 .	For more information, see “Action Types” on page 6-11 .

Administrative Policies

An administrative policy defines an administrative action to perform when a constraint is violated. For example, an administrative action can generate an email notification or an Administration Console event message if some constraint is violated.

[Table 6-3](#) indicates the types of constraints and actions that can be used to define an administrative policy.

Table 6-3 Administrative Policy Constraints and Actions

Constraints Types	Action Types
Deployment	Notification
Runtime	Configuration
For more information, see “Constraint Types” on page 6-6 .	For more information, see “Action Types” on page 6-11 .

Constraint Types

WLOC provides out-of-box constraint types for defining requirements using a range of common and important measurements of health and performance. [Table 6-4](#) describes the out-of-box constraint types.

Note: For detailed information about these constraint types, see [“Define Constraints”](#) in the *WLOC Administration Console Online Help*.

Table 6-4 Constraint Types

Type	Description
Deployment	<p>These define the desired deployment state (Deployed, Staged, Undeployed) of a service.</p> <p>In the Administration Console, these are selectable as:</p> <ul style="list-style-type: none"> Based on deploying a service at a date/time Based on undeploying a service at a date/time Based on a service deployment state
Runtime	<p>These are JVM-based constraints and can be used in various ways, such as specifying the required number of running instances, setting the minimum amount of CPUs to allocate, or generating an event message based on some metric.</p> <p>Runtime constraints based on MBean attributes can be defined using directly observable metrics (heap size) or on functions applied to these metrics (total heap size of all JVMs). For more information, see “Constraints Based on MBean Attributes” on page 6-8.</p> <p>In the Administration Console, these are selectable as:</p> <ul style="list-style-type: none"> Based on firing a service event at a date/time Based on firing a process group event at a date/time Based on firing a process group event based on the outcome of an action Based on firing a process group event based on the outcome of another event Based on a named attribute Based on the value of an attribute or function

Table 6-4 Constraint Types (Continued)

Type	Description
Resource Agreement	<p>These constraints are used for service placement and JVM creation. When defining the constraint in the Administration Console, it is not bound to an action. In <code>metadata-config.xml</code>, the constraint must be bound to an action whose value is INTERNAL.</p> <p>In the Administration Console, these are selectable as:</p> <ul style="list-style-type: none">Based on a maximum amount of CPUBased on a share of CPUBased on a minimum amount of memoryBased on a maximum amount of memoryBased on a share of memoryBased on minimum number of processesBased on maximum number of processesBased on software availabilityBased on IP AddressBased on ISO software availabilityBased on amount of Local Disk Size required

Constraints Based on MBean Attributes

Runtime constraints based on MBean attributes are evaluated against runtime JMX attributes collected from the managed JVM(s). They can be based on directly-observed metric values of attributes or on custom metrics obtained by applying some function on these values for one JVM instance (or some aggregation of JVM instances).

Named Attributes

The WLOC Administration Console provides a set of named constraints, referred to as SmartPack constraints, that are based on named MBean attributes. When you select one of these constraints, the console will automatically provide the MBean instance name, type, and the attribute name. [Table 6-5](#) describes the Smart Pack constraints.

Table 6-5 Named Attribute Constraints

Constraints	Description
MinFreeHeapSize MaxFreeHeapSize	Minimum and maximum free heap size for any one JVM in the process type.
MinAverageFreeHeapSize MaxAverageFreeHeapSize	Minimum and maximum mean free heap size of all JVMs in the process type.
MinFreePhysicalMemory MaxFreePhysicalMemory	Minimum and maximum free physical memory size for any one JVM in the process type.
MinAverageFreePhysicalMemory MaxAverageFreePhysicalMemory	Minimum and maximum mean free physical memory size of all JVMs in the process type.
MinUsedPhysicalMemory MaxUsedPhysicalMemory	Minimum and maximum used physical memory size for any one JVM in the process type.
MinAverageUsedPhysicalMemory MaxAverageUsedPhysicalMemory	Minimum and maximum mean used physical memory size of all JVMs in the process type.
MinJvmProcessorLoad MaxJvmProcessorLoad	Minimum and maximum percent of CPU usage in a range of 0.0 to 1.0. A value of 0.5 equates to 50% of CPU.
MinAverageJvmProcessorLoad MaxAverageJvmProcessorLoad	Minimum and maximum mean percent of CPU usage of all VMs in the process type.

For the purposes of directly editing `metadata-config.xml`, [Listing 6-5](#) shows an XML snippet containing a named attribute constraint.

Listing 6-5 MBean Named Attribute Constraint

```

<ns2:instance-name>com.bea:Name=myserver,Type=ServerRuntime</ns2:instance-
name>
<ns2:instance-type>weblogic.management.runtime.ServerRuntimeMBean</ns2:ins
tance-type>
<ns2:attribute-name>RestartsTotalCount</ns2:attribute-name>
<ns2:constraint-type>max</ns2:constraint-type>
<ns2:value>10</ns2:value>

```

Custom Metrics

A constraint can be based on simple numerical comparison or more complex functions applied to an MBean attribute. Constraint functions can include metric parameters using the following syntax:

```
[<type>|<instance name>|<attribute name>]
```

For example:

```
Sum([weblogic.management.runtime.ServerRuntimeMBean|. *|RestartsTotalCount],  
[weblogic.management.runtime.ServerRuntimeMBean|. *|Port])
```

where

The instance name specification allows regular expression syntax so you are not required to use names that may be context-specific. In the following examples, "." is used as the instance name, but it could be any regular expression.

Sum — Function

weblogic.management.runtime.ServerRuntimeMBean — Instance type

. * — Instance name

RestartsTotalCount — Attribute name

Port — Second attribute name

A metric parameter can also contain other functions. For example:

```
Sum(Sum([weblogic.management.runtime.ServerRuntimeMBean|. *|RestartsTotalCo  
unt],[weblogic.management.runtime.ServerRuntimeMBean|. *|RestartsTotalCount  
]),[weblogic.management.runtime.ServerRuntimeMBean|. *|RestartsTotalCount])
```

Scalars in function definitions are constant numeric values. Scalars cannot be replaced by functions. For example:

```
SumWithScalar([weblogic.management.runtime.ServerRuntimeMBean|. *|RestartsT  
otalCount],37)
```

For a complete list of WLOC-supplied functions, see [“Define custom metrics”](#) in the *WLOC Administration Console Online Help*.

[Listing 6-6](#) show an XML snippet defining a constraint that uses the `MeanofCollection` function. Since a function returns a *double*, it is encapsulated in a `double-constraint` element. The `instance-type` element of these constraints is always `FUNCTION` and the function definition itself is specified in the `attribute-name` element.

Note that the value in the `instance-name` element does not have to match the actual function name.

Listing 6-6 Constraint with a Function in Metadata-config.xml

```

<ns2:double-constraint>
  <ns2:name>MyConstraint</ns2:name>
  <ns2:key>MyConstraintKey</ns2:key>
  <ns2:priority>0</ns2:priority>
  <ns2:state>deployed</ns2:state>
  <ns2:instance-name>MeanOfCollection</ns2:instance-name>
  <ns2:instance-type>FUNCTION</ns2:instance-type>
  <ns2:attribute-name>MeanOfCollection([weblogic.management.runtime.
    ServerRuntimeMBean|. * |ServerStartupTime])</ns2:attribute-name>
  <ns2:constraint-type>max</ns2:constraint-type>
  <ns2:value>10000</ns2:value>
</ns2:double-constraint>

```

Action Types

[Table 6-6](#) describes out-of-box actions provided with WLOC. When creating these actions in the WLOC Administration Console, the console prompts for the required parameter values.

Note: For detailed information about these action types, see [“Define Actions”](#) in the *WLOC Administration Console Online Help*.

Table 6-6 Action Types

Action Type	Description
Service	Actions that start, stop, stage, test, or destroy a service. StageServiceAction StartServiceAction StopServiceAction DestroyServiceAction TestStartServiceAction

Table 6-6 Action Types

Process	<p>Actions that start, stop, destroy, stage, suspend, change the configuration, or gracefully stop an instance of a Java process.</p> <p>StartJavaInstanceAction StopJavaInstanceAction DestroyJavaInstanceAction StageJavaInstanceAction ResumeJavaInstanceAction RemoteAction SuspendJavaInstanceAction JavaInstanceConfigAction</p>
Notification	<p>These actions send different types of notifications.</p> <p>Console EmailNotificationAction JMXNotificationAction JMSNotificationAction SNMPNotificationAction</p>
Resource	<p>Actions that change the maximum, minimum or share of CPU or memory that is available.</p> <p>ChangeMaxCPUAction ChangeMinCPUAction ChangeShareCPUAction ChangeMaxMemoryAction ChangeMinMemoryAction ChangeShareMemoryAction</p>
Configuration	<p>Actions that perform JMX operations on specified managed objects. You can define actions that:</p> <ul style="list-style-type: none"> • Creates an instance of the MBean. • Destroys an instance of the MBean. No additional parameters are needed. • Sets an MBean attribute to a specified value • Invokes an operation on the MBean.

Note: Most actions can also be executed directly from the Administration Console without being bound to a constraint and included in a policy.

Action Pipelines

Actions can be combined into action pipelines so that a sequence of actions takes place when a constraint is violated.

Note: If an action fails at any point in the pipeline, the pipeline terminates and no subsequent actions in the pipeline are executed.

Action Pipeline Example

To define the actions needed to start a WebLogic Server Admin Server instance, then start a cluster of Managed Server instances, and then send an email notification, you would do the following:

1. Define an action that starts the WLS Admin Server instance.
2. Define an action that starts the Managed Server instances.
3. Define an action that sends e-mail notification when the server instances start up.
4. Define an action pipeline that consists of all these actions.
5. Create a policy with an appropriate constraint and bind the pipeline to the constraint.

Action Pipelines

Configuring Security

This section provides information about securing WebLogic Operations Control (WLOC). It includes the following topics:

- [“WLOC Users, Groups, and Security Roles”](#) on page 7-1
- [“Secure Communications”](#) on page 7-4
- [“Keystores”](#) on page 7-10
- [“Password Encryption”](#) on page 7-16
- [“File System Security”](#) on page 7-16

WLOC Users, Groups, and Security Roles

To secure access to WLOC, WLOC uses role-based access control, which enables you to assign different levels of privileges to different users or groups. WLOC provides the security roles and determines their access privileges. To facilitate the administration of large numbers of users, WLOC also provides a set of groups that you can configure to be in one or more WLOC roles. You use the WLOC Administration Console to create users and assign them to groups or directly to security roles. For more information about using the WLOC Administration Console to manage users, groups, and security roles, see [“Manage users and groups”](#) in the *WLOC Administration Console Online Help*.

Anonymous users cannot access the WLOC Administration Console.

WLOC Boot User

When running the WLOC Configuration Wizard to create the Controller, the wizard prompts for the username and password of the boot user. This account is used to log into the WLOC Administration Console. The default username and password are *WLOCBootUser* and *changeit* respectively.

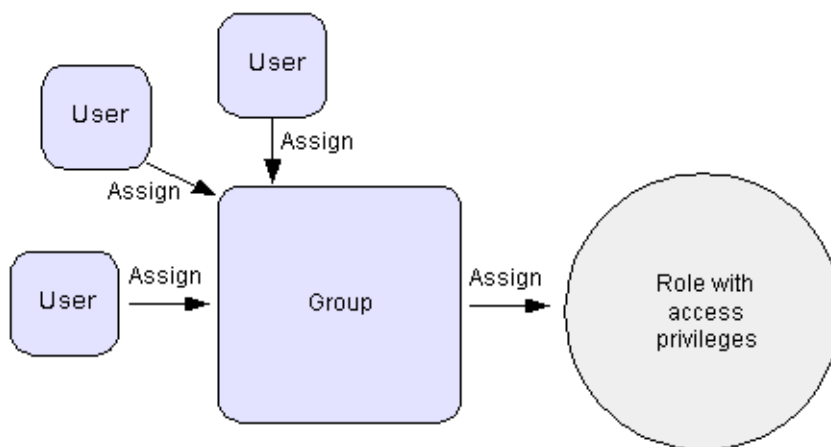
Note: To protect the security of your system, Oracle recommends that you change this password, especially in a production environment. To do so, see [“Modify Users”](#) in the *WLOC Administration Console Online Help*.

WARNING: Do not delete the boot user. Otherwise, you will no longer be able to log into the WLOC Administration Console without recreating the Controller.

Users and Groups

A user is an entity that can be authenticated. WLOC uses password authentication only. A group is a collection of users who usually have something in common, such as working in the same department in a company. For efficient security management, Oracle recommends that instead of adding users directly to security roles, you add users to groups and then you assign groups to security roles.

Figure 7-1 Assign Users to Groups and Groups to Roles



[Table 7-1](#) describes WLOC security groups and the out-of-box security role assignment for the group. You may create additional groups, but may not create additional security roles. You can indirectly create a scoped service admin role by creating a service; when you create a new service, a security role is created to administer that service. In addition, for each service that you create, WLOC creates a group named `ServiceAdministrators_Service-Name`, where *Service-Name* is the name of the service. To allow a user to manage all services, you should add the user to the `ServiceAdministrators` group. To allow a user to manage only a single *Service-Name* service, you should add the user to the `ServiceAdministrators_Service-Name` group. The `ServiceAdministrators_Service-Name` group is deleted when the service is deleted.

Table 7-1 WLOC Security Groups

Group	Role Assignment
Administrators	Admin
ServiceAdministrators	ServiceAdmin
Monitors	Monitor
ServiceAdministratorsScoped	ServiceAdminScoped

WLOC Security Roles

WLOC provides the following security roles:

- Admin
- ServiceAdmin
- Monitor

In addition, for each service that you create, WLOC provides a role named `ServiceAdmin_Service-Name`, where *Service-Name* is the name of the service.

[Table 7-2](#) describes the access privileges for the WLOC security roles. You cannot create additional security roles or change the access privileges for any role. You should not add users to the `ServiceAdmin_Service-Name` role; you should add them to the `ServiceAdministrators_Service-Name` group instead.

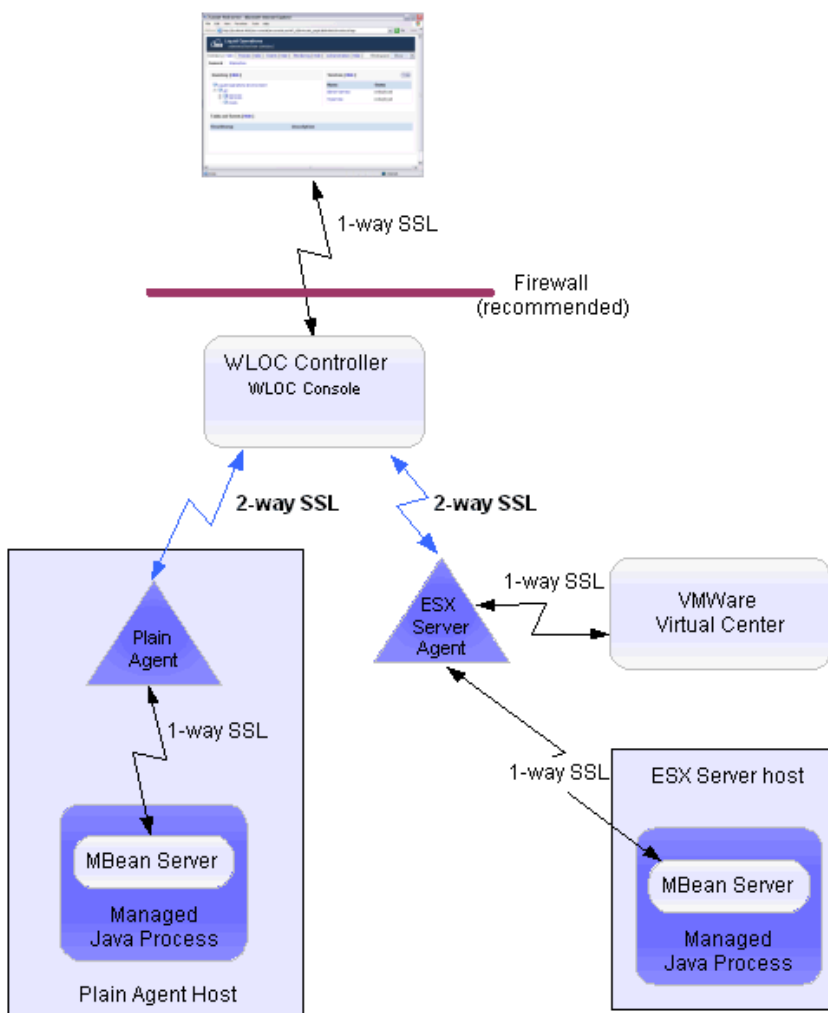
Table 7-2 WLOC Security Roles

	Admin	Service Admin	Monitor	ServiceAdminService-Name
Configure the Controller and Agents	X			
View monitoring data in the console	X	X	X	Only for Service-Name
Create and modify users and groups	X			
Create and modify resource pools	X			
Create and modify services	X	X		Only for Service-Name
Deploy or undeploy services	X	X		Only for Service-Name
Create and modify policies	X	X		Only for Service-Name
Approve adjudications	X	X		Only for Service-Name

Secure Communications

To secure communications between WLOC, managed endpoints, and the infrastructure that hosts the endpoints, WLOC uses 1-way or 2-way SSL. It assumes that you have configured perimeter security measures (e.g., firewalls) and restricted access to the host and its filesystem to trusted users.

Figure 7-2 summarizes security for WLOC communication.

Figure 7-2 Security for WLOC Communication

Configuring Firewalls

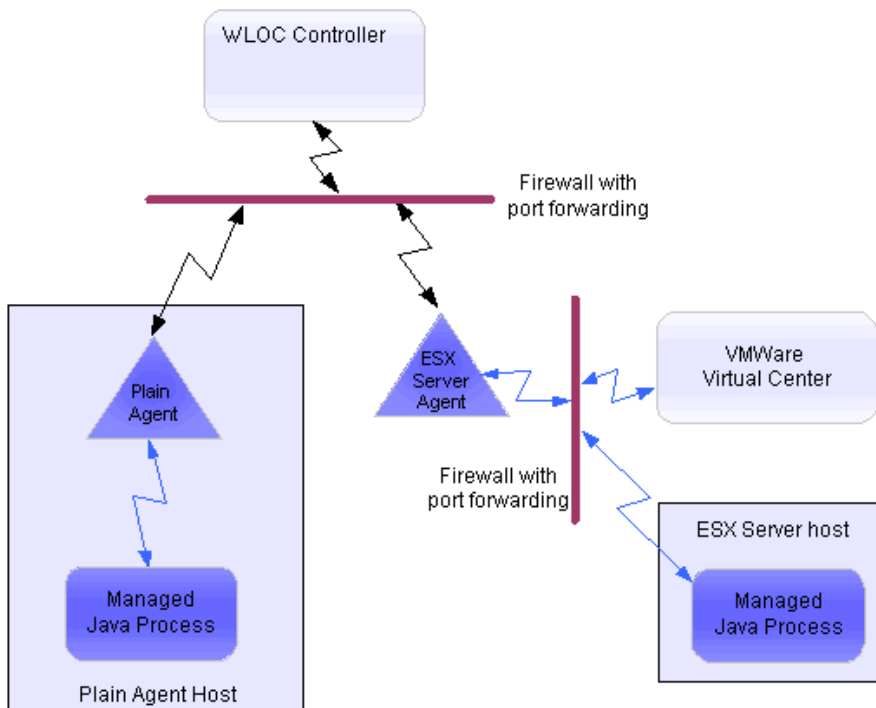
Oracle recommends that you place the Controller and all Agents behind the same firewall whenever possible (see [Figure 7-2](#)). Controllers and Agents communicate only through HTTPS if secure communications is enabled. In this communication, the Controller always initiates

communication by sending a request and the Agent returns a response. Once the Controller has initiated a connection to the Agent and the Agent responds, either the Agent or the Controller can initiate an HTTPS connection.

If you place Controllers and Agents behind separate firewalls, you must configure the firewall to open at least one HTTPS communication channel. In addition, you must configure port forwarding for each Agent behind the firewall. In such an environment, the Controller sends requests to the firewall and the firewall forwards each request to the appropriate Agent. See [Figure 7-3](#).

Similarly, Agents use one-way protocols (HTTPS or IIOPS) to communicate with hypervisor software and with managed processes. If you use a firewall to separate Agents from hypervisor software and managed processes, you must configure the firewall to forward requests from the Agent to the hypervisor software and managed processes.

Figure 7-3 Firewalls with Port Forwarding



Securing WLOC Administration Console to Controller Communication

Communication between the Web browser and the WLOC Administration Console is secured using 1-way SSL. In this communication:

- The browser is in the role of SSL client and the WLOC Controller is in the role of SSL server.
- To provide its identity, the WLOC Controller sends a certificate from its JKS identity keystore. This identity JKS keystore is created when WLOC is installed. After running the WLOC Configuration Wizard, a self-signed demonstration certificate (stored under the alias of `controller`) is included in the keystore. See [“Keystores” on page 7-10](#).

To enable secure communication between the Administration Console and the Controller, set the Controller’s **Console Security Mode** to `SECURE` or `BOTH`. This can be set when running the configuration wizard. In addition, it can be changed using Administration Console on the **Networking** tab or by modifying the `<console-mode>` element in *loc-controller-config.xml*.

Securing Controller to Agents Communication

The Controller uses a separate SSL connection to communicate with each Agent using 2-way SSL over HTTPS. In this communication the Controller is the SSL client and each Agent is an SSL server.

To enable secure connections between a Controller-Agent connections:

- Each Agent’s internal trust keystore must contain a certificate for the Controller with which it communicates. For instructions, see [“Importing the Controller Identity into an Agent Trust Keystore” on page 7-8](#).
- The Controller’s internal trust keystore must contain a certificate for each Agent with which it communicates. For instructions, see [“Importing an Agent Identity into the Controller Trust Keystore” on page 7-8](#).
- Both the Controller and all its Agents must be set to use secure connections. This is established by setting the **Security Mode** to *Secure* when running the Configuration Wizard to create the Controller and Agent instances. After creating the instances, this value can be changed using the Administration Console or by editing the `<use-secure-connections>` element in the configuration XML file.

Note: Both the Controller its Agents must be set to the same security mode.

While the use of 2-way SSL prevents potential attackers from snooping communications, an Agent's Web Services interface does not require authorization. An Agent assumes that once an SSL connection is established, the caller is trustworthy. To prevent unauthorized use of WLOC Agents, ensure that all Agents are behind a firewall and that secure communications between the Controller and Agent is enabled.

Unsecure connections can be used in testing or other environments that do not require encryption. Be aware, however, that this would allow an attacker to directly access all Agent and Controller functions without any constraints.

Importing the Controller Identity into an Agent Trust Keystore

Importing the Controller's identity certificate into an Agent's internal trust keystore is performed using the Keytool utility located in the JDK `bin` directory, for example `BEA_HOME\jrockit_xxx_xx\bin`. To perform the import, follow these steps.

1. To import the Controller's identity certificate, open a Command Prompt window or shell and enter the following command:

```
keytool -import -noprompt -alias controllerinternal -file  
BEA_HOME/user_projects/controller/ssl/internal/controllerinternalidenti  
ty.pem -keystore  
BEA_HOME/user_projects/agent1/ssl/internal/internaltrust.jks -storepass  
jks_password
```

where

controllerinternal is an alias to create.

jks_password is the current password for the Agent's internal trust keystore. (The default value is *changeit*.)

2. To verify the import, enter the following command:

```
keytool -list -v -keystore  
BEA_HOME/user_projects/agent1/ssl/internal/internaltrust.jks -storepass  
jks_password
```

Importing an Agent Identity into the Controller Trust Keystore

Importing the Agent's identity certificate into the Controller's internal trust keystore is performed using the Keytool utility located in the JDK `bin` directory, for example

`BEA_HOME\jrockit_xxx_xx\bin`. To perform the import, follow these steps.

1. Open a Command Prompt window or shell and enter the following command:


```
keytool -import -noprompt -alias agentinternal -file
BEA_HOME/user_projects/agent1/ssl/internal/agentinternalidentity.pem
-keystore
BEA_HOME/user_projects/controller/ssl/internal/internaltrust.jks
-storepass jks_password
```

where

agentinternal is an alias to create. This must be different for each imported Agent identity.

jks_password is the Controller's internal trust keystore password. (The default value is *changeit*.)

2. To verify the import, enter the following command:

```
keytool -list -v -keystore
BEA_HOME/user_projects/controller/ssl/internal/internaltrust.jks
-storepass jks_password
```

Securing Agent to VMware Virtual Center Communication

All communications between Agents and Virtual Center are secured using 1-way SSL. In this communication the Agent is the SSL client and Virtual Center is the SSL server. To enable SSL, the Virtual Center's certificate must be imported into the Agent's internal trust keystore.

Importing the Virtual Center Identity into an Agent Trust Keystore

To configure secure connections between an Agent and the Virtual Center, the Virtual Center's identity certificate must be imported into the Agent's trust keystore. This will be automatically performed if you select the **Connect to the Virtual Center to retrieve SSL certificate** option when creating the Agent instance with the Configuration Wizard.

The Virtual Center identity can also be imported by using the GrabCert utility to connect to the Virtual Center to get the certificate and put it in the truststore. You can then use the Keytool utility to verify that the import was successful. The GrabCert utility is located in the *WLOC_HOME\common\bin* directory. The Keytool utility is located in the JDK bin directory, for example *BEA_HOME\jrockit_xxx_xx\bin*.

To do this, follow these steps.

1. Use the GrabCert utility to copy the certificate from the Virtual Center to

```
BEA_HOME/user_projects/ESXAgent/ssl/internal.
```

```
java -jar ./wloc_10.3/common/bin/GrabCert.jar host[:port]
[-alias=alias] [truststorepath [truststorepassword]]
```

where:

host:port is the host and port to connect to using SSL in order to request the remote site's certificate

-alias is the certificate alias to use when saving the remote certificate in the local truststore

truststorepath is the location of the local truststore in which to save the remote certificate. The truststore will be created if it doesn't already exist.

truststorepassword is the password for the local truststore

The defaults are as follows:

```
port: 443
alias: hostname
truststorepath: internaltrust.jks
truststorepassword: changeit
```

2. To verify the import, enter the following command:

```
keytool -list -v -keystore
BEA_HOME/user_projects/ESXAgent/ssl/internal/internaltrust.jks
-storepass jks_password
```

where

jks_password is the keystore password (default is *changeit*).

Securing Agent to MBean Server Communication

An Agent communicates with MBean Servers using IIOPS over a 1-way SSL connection. In this communication, the Agent is the SSL client and the MBean server is the SSL server. The managed process provides a certificate which you import into the managing Agent's internal trust keystore.

In addition, MBean Servers are usually configured to require authentication using a user name and password as credentials. (WebLogic Server MBean Servers always require authentication and authorization.)

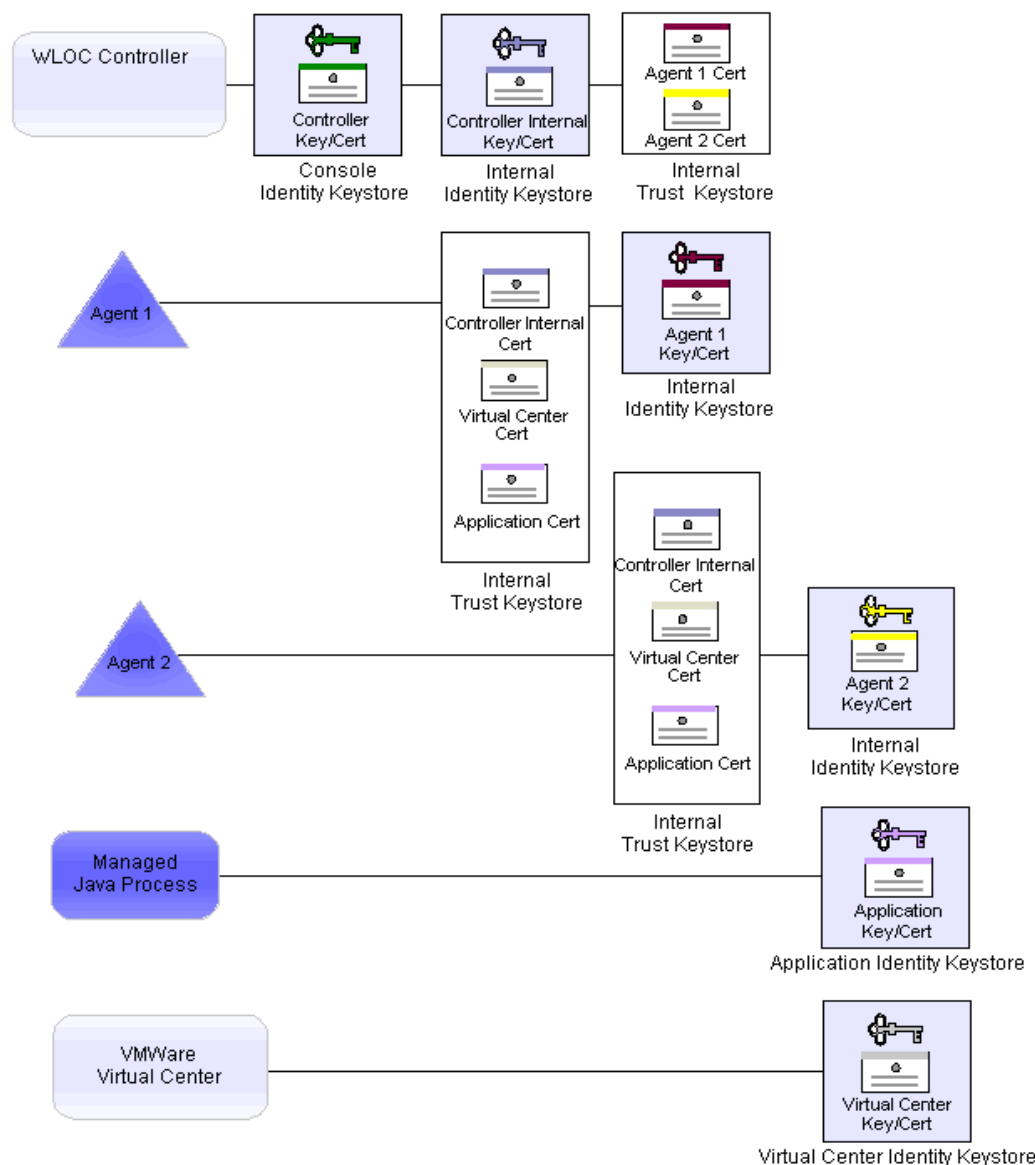
Keystores

For secure communications, WLOC uses Java keystores that are distributed throughout the environment. The WLOC Configuration Wizard creates all WLOC keystores and populates the identity keystores with private keys and self-signed certificates. The self-signed certificates are

also available as *pem* files outside of the identity keystores, so they can be imported into the corresponding trust keystores.

You populate the trust keystores. Managed processes and hypervisor software provide their own identity keystores. WLOC does not create keystores for managed processes and hypervisor software. See [Figure 7-4](#).

Figure 7-4 Distribution and Contents of WLOC Keystores



Controller Keystores

To support 1-way SSL between Web browsers and the WLOC Administration Console, the Controller uses the [Console Identity Keystore](#). To support 2-way SSL between the Controller and each Agent, the Controller uses the [Controller Internal Identity Keystore](#) and [Controller Internal Trust Keystore](#).

Console Identity Keystore

The console identity keystore contains the private key and self-signed certificate that supports 1-way SSL communication between a Web browser and the WLOC Administration Console application. When a user initiates a session with the WLOC Administration Console application, the Web browser prompts the user to trust the Controller's certificate.

Keystore characteristics:

- The WLOC Configuration Wizard creates this keystore when you configure the Controller. It also creates the private key and self-signed certificate.
- The keystore is named `controlleridentity.jks` and is located in `CONTROLLER_DIR\ssl`, where `CONTROLLER_DIR` is the directory in which you install the Controller, for example `BEA_HOME\user_projects\controller\ssl\controlleridentity.jks`.
- The certificate in this keystore is saved under the alias `controller` and contains the name of the host on which the Controller is installed. It has a key size of 1024 and a private key type of RSA with a signature algorithm of MD5withRSA. It expires in 10958 days (30 years).
- You can replace the default private key and certificate with one that you generate. You can use the keystore tools that are provided in the JDK for this type of modification. When you replace the default private key and certificate, make sure that the replacement has the alias `controller`.
- Access to the keystore via the JDK keystore tools is protected by a password. You specify the password when running Configuration Wizard (the default value is *changeit*). The password is stored as an encrypted string in the Controller's configuration XML document.

Controller Internal Identity Keystore

- The Controller's internal identity keystore contains the private key and self-signed certificate that supports 2-way SSL communication between the Controller and all Agents.

The WLOC Configuration Wizard creates this keystore as well as the private key and self-signed certificate.

The keystore is named `internalidentity.jks` and is located in

`CONTROLLER_DIR\ssl\internal` where `CONTROLLER_DIR` is the directory in which you install the Controller, for example

`BEA_HOME\user_projects\controller\ssl\internal\internalidentity.jks`.

The certificate in this keystore is saved under the alias `controllerinternal` and contains the name of the host on which the Controller is installed. It has a key size of 1024 and a private key type of RSA with a signature algorithm of MD5withRSA. It expires in 10958 days (30 years).

Access to the keystore via the JDK keystore tools is protected by a password. You specify the password when running the Configuration Wizard. The password is stored as an encrypted string in the Controller's configuration document. While this keystore is not intended to be modified, you can use the JDK keystore tools to read the certificate values.

Controller Internal Trust Keystore

The Controller's internal trust keystore contains a copy of each Agent's self-signed certificate. (Note that WLOC does not support the use of CA certificates in its internal trust keystores.) These certificates are required to support 2-way SSL communication between the Controller and all Agents.

The WLOC Configuration Wizard creates this keystore, but does not populate it with Agent identities. This is accomplished as described in [“Importing an Agent Identity into the Controller Trust Keystore” on page 7-8](#).

The keystore is named `internaltrust.jks` and is located in

`CONTROLLER_DIR\ssl\internal` where `CONTROLLER_DIR` is the directory in which you install the Controller, for example

`BEA_HOME\user_projects\controller\ssl\internal\internaltrust.jks`.

Agent Keystores

To support 2-way SSL between an Agent and the Controller, as well as 1-way SSL between an Agent and managed processes and between an Agent and VMware Virtual Center, an Agent uses its [Agent Identity Keystore](#) and [Agent Internal Trust Keystore](#).

Agent Identity Keystore

An Agent's identity keystore contains the private key and self-signed certificate that supports 2-way SSL communication between the Agent and the Controller.

The WLOC Configuration Wizard creates this keystore as well as the private key and self-signed certificate. You cannot replace or modify the key or certificate in this keystore.

The keystore is named `internalidentity.jks` and is located in `AGENT_DIR\ssl\internal` where `AGENT_DIR` is the directory in which you install an Agent, for example `BEA_HOME\user_projects\agent1\ssl\internal\internalidentity.jks`.

The certificate in this keystore is saved under the alias `agentinternal` and contains the name of the host on which the Agent is installed. It has a key size of 1024 and a private key type of RSA with a signature algorithm of MD5withRSA. It expires in 10958 days (30 years).

Access to the keystore via the JDK keystore tools is protected by a password. You specify the password when you configure the Agent. The password is stored as an encrypted string in the Agent's configuration XML document.

While this keystore is not intended to be modified, you can use the JDK keystore tools to read the certificate values.

Agent Internal Trust Keystore

An Agent's internal trust keystore contains a copy of the Controller's self-signed certificate, as well as a copy of the CA certificates of the processes that it manages. A Hypervisor Agent also contains a copy of the VMware Virtual Center CA certificate. These certificates are required to support 2-way SSL communication between the Agent and Controller, as well as 1-way SSL between an Agent and managed processes and between an Agent and VMware Virtual Center.

The WLOC Configuration Wizard creates this keystore when you configure an Agent. It is named `internaltrust.jks` and is located in `AGENT_DIR\ssl\internal` where `AGENT_DIR` is the directory in which you install an Agent, for example `BEA_HOME\user_projects\agent1\ssl\internal\internaltrust.jks`.

To import the Controller's identity into the keystore, see [“Importing the Controller Identity into an Agent Trust Keystore” on page 7-8](#). To import the Virtual Center's identity into an ESX Agent's keystore, see [“Importing the Virtual Center Identity into an Agent Trust Keystore” on page 7-9](#).

Password Encryption

Any passwords you enter when installing WLOC, running the Configuration Wizard, or using the WLOC Administration Console will be encrypted before being stored in any of the XML configuration files.

WARNING: If you enter or modify a password by directly editing the configuration XML file, the password will remain in clear text until a new configuration file is generated (usually as a result of making a change to the configuration using the Administration Console), at which time any passwords in clear text will be encrypted. Note that simply stopping and re-starting the Controller or an Agent will not encrypt any clear text passwords.

File System Security

The file system security in your environment should completely restrict access to WLOC files and directories by non-WLOC users. This includes development or testing environments where passwords may be stored in clear text.

For more information about securing your WLOC installation, see [Securing a Production Environment](#).

Logging, Auditing, and Monitoring

This section describes how to monitor, log, and audit WLOC services and resources.

- [“Logging” on page 8-1](#)
- [“Auditing WLOC Actions” on page 8-7](#)
- [“Monitoring” on page 8-10](#)
- [“Viewing Events” on page 8-10](#)

Logging

The WLOC Controller and each Agent generate log messages that provide information about events such as service deployments, action failures, and other events. These messages are saved in log files that, by default, are located in the `log` sub-directory where the Agent or Controller was installed.

In addition, the Controller and Agents output messages to standard out. You can filter this output by message severity. For example, you can configure that only messages of `WARNING` severity or higher are output to standard out.

Configuring Logging

You can configure the following aspects of the logging feature:

Table 8-1 Logging Configuration

Field	Description
Severity	The severity of messages to write to the log file. See “Message Severity” on page 8-6 .
Log File Name	<p>Path name of the log file. The path can be absolute or relative to the Controller or Agent installation directory.</p> <p>To include a time and date stamp in the file name when the log file is rotated, add <code>java.text.SimpleDateFormat</code> variables to the file name. Surround each variable with percentage (%) characters.</p> <p>For example, if the file name is defined to be <code>myserver_%yyyy%_%MM%_%dd%_%hh%_%mm%.log</code>, the log file will be named <code>myserver_yyyy_mm_dd_hh_mm.log</code>.</p> <p>When the log file is rotated, the rotated file name contains the date stamp. For example, if the log file is rotated on 2 April, 2008 at 10:05 AM, the log file that contains the old messages will be named <code>myserver_2008_04_02_10_05.log</code>.</p> <p>If you do not include a time and date stamp, the rotated log files are numbered in order of creation. For example, <code>myserver.log00007</code>.</p>
Rotation Type	Rotation style to use for rotating log files. Valid options include By Size and By Time. For more information about log file rotation, see “Rotating Log Files” on page 8-6 .
Rotation Size	<p>Maximum log file size (in KBs) before the current log file is rotated and a new log file is created. This value is valid only if the Rotation type was set to By Size.</p> <p>When the file size is reached, the file is renamed by incrementing the file extension (e.g., <code>controller.log0001</code>, <code>controller.log0002</code>, etc.) and saved in the rotation directory. The server then writes messages to a new file named <i>filename.log</i>.</p> <p>WLOC sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.</p>
Rotation Directory	Directory in which log files are stored.

Table 8-1 Logging Configuration (Continued)

Field	Description
Number of Files Limited	<p>Flag that specifies whether to limit the number of stored log files. (Requires that you specify a file rotation type of SIZE or TIME.) After WLOC reaches this limit, it deletes the oldest log file and creates a new log file with the latest suffix.</p> <p>If you do not enable this option, the server creates new files indefinitely and you must clean up these files as necessary.</p>
Rotation File Count	<p>Maximum number of log files to create before overwriting them. When the maximum number of log files is reached, the oldest file is overwritten. This value is valid only if the Number of files limited flag is enabled.</p> <p>This number does not include the file that the server uses to store current messages.</p>
Rotation on Startup	Flag that specifies whether log files should be rotated automatically at startup.
Rotation Time	<p>Time of day that log files are rotated initially. Specify using the format: hh:mm.</p> <p>This value is used only if the Rotation Type is set to By Time.</p> <p>If the time you specify has already elapsed, WLOC will perform the day's rotation immediately.</p>
Rotation Time Span	<p>Frequency at which log files are rotated with Rotation Time Span Factor. Specify a number using the format: hh:mm. The default is 00:00 and equates to midnight.</p> <p>The actual rotation frequency is calculated as follows: $(\text{Rotation Time Span}) * (\text{Rotation Time Span Factor})$.</p> <p>This value defaults to 24 and is valid only if the Rotation type is set to By Time.</p>
Rotation Time Span Factor	<p>Frequency at which log files are rotated with Rotation Time Span. The rotation frequency is calculated as follows: $(\text{Rotation Time Span}) * (\text{Rotation Time Span Factor})$.</p> <p>This value is specified in milliseconds and defaults to 3600000. This value is valid only if the Rotation type is set to By Time.</p>
Standard Out Severity	The severity of log file messages to write to standard out. See “Output to Standard Out and Standard Error” on page 8-4 and “Message Severity” on page 8-6 .

Viewing Log Messages

Log messages can be viewed as follows:

- All current Controller log files can be viewed in the Administration Console by selecting the Controller's **Logs** tab. All rotated log files can be examined only by opening the log files individually.

The default location of Controller logs is the `logs` subdirectory where the Controller was installed.

- All Agent log files must be examined by accessing the log files directly on the file system. Currently, they cannot be viewed in the WLOC Administration Console.

The default location of Agent logs is the `logs` subdirectory where the Agent was installed.

Note: Do not modify log files by editing them manually. Modifying a file changes the timestamp and can confuse log file rotation. In addition, editing a file might lock it and prevent updates.

Log Message Format

When a Controller or Agent writes a message to its log file, the first line of each message begins with `####` followed by the message attributes. Each attribute is contained between angle brackets.

The following is an example of a message in the Controller's log file:

```
####<Oct 25, 2007 5:32:09 PM EDT> <Info> <LOCExecuteEngine> <> <>
<[ACTIVE] ExecuteThread: '1' for queue: 'weblogic.kernel.Default
(self-tuning) ' > <> <> <> <1193347929031> <BEA-2010502> <Action
Succeeded:EmailNotificationAction([WLS-Cluster,ID=7111863992976504417],
WLS-Cluster,ID=-4438336769125385457) for event
com.bea.adaptive.execute.internal.SyntheticServiceEvent@63bb71.>
```

In this example, the message attributes are: Locale-formatted Timestamp, Severity, Subsystem, Machine Name, Server Name, Thread ID, User ID, Transaction ID, Diagnostic Context ID, Raw Time Value, Message ID, and Message Text. For information about how these attributes are used, see [“Log Message Attributes” on page 8-5](#).

If the message includes a stack trace, the stack trace is included in the message text.

WLOC uses the host computer's default character encoding for the messages it writes.

Output to Standard Out and Standard Error

In addition to writing messages to log files, a Controller or Agent can print a subset of its messages to standard out. Usually, standard out is the shell (command prompt) in which you are running the Controller or Agent. However, some operating systems enable you to redirect standard out to some other location.

You can filter this output by message severity. For example, you can configure that only messages of `WARNING` severity or higher are output to standard out. You can also configure whether WLOC prints stack traces to standard out.

When a Controller or Agent writes a message to standard out, the output does not include the `####` prefix.

Log Message Attributes

The log messages contain a consistent set of attributes as described in [Table 8-2](#). In addition, if your application uses WebLogic logging services to generate messages, its messages will contain these attributes.

Table 8-2 WLOC Log Message Attributes

Attribute	Description
Locale-formatted Timestamp	Time and date when the message originated, in a format that is specific to the locale. The JVM that runs the Controller or each Agent refers to the host computer operating system for information about the local time zone and format.
Severity	Indicates the degree of impact or seriousness of the event reported by the message. See “Message Severity” on page 8-6 .
Subsystem	Indicates the subsystem of WLOC that was the source of the message.
Thread ID	Identifies the origins of the message. The <code>Thread ID</code> is the ID that the JVM assigns to the thread in which the message originated.
User ID	The user ID under which the associated event was executed. To execute some pieces of internal code, WLOC authenticates the ID of the user who initiates the execution and then runs the code under a special Kernel Identity user ID.
Raw Time Value	The timestamp in milliseconds.
Message ID	A unique seven-digit identifier. All message IDs that WLOC generates start with <code>BEA-</code> and fall within a numerical range of 2010000 to 2019999.
Message Text	A description of the event or condition.

Message Severity

The severity attribute of a WLOC log message indicates the potential impact of the event or condition that the message reports. [Table 8-3](#) lists the severity levels of log messages by severity, from lowest to highest.

Table 8-3 Message Severity

Severity	Meaning
TRACE	A trace level event used for user-level debugging.
DEBUG	A debug message used for internal debugging.
INFO	A low-level informational message; used for reporting normal operations.
NOTICE	An informational message with a higher level of importance.
WARNING	A suspicious operation or configuration has occurred but it might not affect normal operation.
ERROR	A user error has occurred. The system or application can handle the error with no interruption and limited degradation of service.
CRITICAL	A system error has occurred. The system can recover but there might be a momentary loss or permanent degradation of service.
ALERT	A particular system is in an unusable state while other parts of the system continue to function. Automatic recovery is not possible; the immediate attention of the administrator is needed to resolve the problem.
EMERGENCY	The Controller or Agent is in an unusable state. This severity indicates a severe system failure or panic.

Rotating Log Files

By default, the Controller and each Agent renames (rotates) its log file when the file grows to a size of 500 kilobytes. Each time the log file reaches this size, the WLOC renames the log file and creates a new *file-name.log* to store new messages. By default, the rotated log files are numbered in order of creation *file-namennnnn*. You can configure WLOC to include a time and date stamp in the file name of rotated log files; for example,

file-name-%yyyy%-%mm%-%dd%-%hh%-%mm%.log.

You can rotation file size, interval, and other properties based on the information in [Table 8-1](#) and [Table 8-4](#).

Some file systems place a lock on files that are open for reading. On such file systems, if your application is tailing the log file, or if you are using a command such as the DOS `tail -f` command in a command prompt, the tail operation stops after the server has rotated the log file. The `tail -f` command prints messages to standard out as lines are added to a file. For more information, enter `help tail` in a DOS prompt.

Debug Log Messages

To help you and Oracle Customer Support diagnose problems with a WLOC environment, WLOC can output debug log messages that provide a detailed description of events in the runtime environment.

You can configure WLOC to output debug messages from all WLOC components or from specific components or scopes. WLOC writes debug messages to its log files and you can configure WLOC to print them to standard out.

Auditing WLOC Actions

By default, the WLOC Audit Service is enabled and writes audit events to an audit log file. The default name of the file is `audit.log` located in the `logs` directory where the Controller or Agent was installed. The current Controller audit log is exposed in the Administration Console. The Controller's rotated audit logs and all Agent logs must be examined by directly accessing the file.

You can configure the Audit Service in the Administration Console. For the Controller, this is performed on the Controller's **Audit** tab. For an Agent, it is performed on the **Audit** tab of the individual Agent. You specify name and rotation settings for Audit logs using the same settings as those used for Controller logs. For a description of these fields, see [Table 8-1](#). In addition, you can specify the audit types and whether or not to enable auditing as described in [Table 8-4](#).

Table 8-4 Audit Service Configuration

Field	Description
Enabled	Flag that specifies whether the audit log service is enabled.
Audit Types	<p>The category of events to audit.</p> <p>For an Agent, select ALL or AGENT_ACTION.</p> <p>For the Controller, select ALL or one or more of the following:</p> <p>CONTROLLER_CONFIGURATION SERVICE_CONFIGURATION RULES CONTROLLER_ACTION ADJUDICATION AGENT_CONFIGURATION</p> <p>For more information, see “Audit Event Types” on page 8-8.</p>

Audit Event Types

The WLOC Audit Service generates and records audit events as described in [Table 8-5](#).

Table 8-5 Audit Event Types

Field	Description
ALL	Include all audit types.
CONTROLLER_CONFIGURATION	(Controller Only) Any changes to the configuration of the Controller.
AGENT_CONFIGURATION	(Controller Only) Any changes to the configuration of an Agent.
SERVICE_CONFIGURATION	(Controller Only) Any changes to the configuration of a service.
RULES	(Controller Only) Each time a rule evaluates to true, WLOC generates a Rule Audit Event containing the relevant context of the Rule execution.
CONTROLLER_ACTION	(Controller Only) An action initiated by a WLOC policy.

Table 8-5 Audit Event Types

Field	Description
ADJUDICATION	(Controller Only) An adjudication decision (approves/deny).
AGENT_ACTION	(Agent Only) Any JVM lifecycle actions (such as stage/start/stop/destroy) initiated by the Agent.

Audit Format

The WLOC Audit Service uses message identifiers numbered 2014100 through 2014199 for all audit entries. Audit records begin with the ##### marker and each field uses the < prefix and > suffix. [Table 8-6](#) describes audit record fields.

Table 8-6 Audit Record Fields

Field	Description
Date	Message time and date in a format that is specific to the locale. The JVM that runs the Controller or each Agent refers to the host computer operating system for information about the local time zone and format. For example, Aug 20, 2007 2:11:24 PM EDT
Severity	The severity levels are INFO, NOTICE, WARNING, ERROR, CRITICAL, ALERT, EMERGENCY, DEBUG.
Subsystem	LOCAudit
Server Name	The Controller or Agent's machine name.
Thread ID	ID that the JVM assigns to the thread in which the message originated.
Timestamp	A millisecond timestamp.
Audit Type	Always <Audit>. Also see “Audit Event Types” on page 8-8 .
Message ID	A unique message identifier to enable easy retrieval and sorting for Audit events of a given type. The message ID range for WLOC is 2014100 through 2014199.
Message Body	Message text.

Listing 8-1 Example Audit Record

```
####<Aug 20, 2007 2:11:24 PM EDT> <Info> <LOCAudit> <seacoast1> <Thread-19>  
<1187633484747> <ControllerAction> <BEA-2013411> <Configured Pipeline  
configName with Pipeline Identifier PipeID successfully completed execution  
at time Aug 20, 2007 2:11:24 PM EDT. Execution elapsed time was 10,000  
milliseconds.>
```

Monitoring

For all active resources, the WLOC Administration Console provides flexible tools for designing and displaying charts and graphs. For active services, resource pools, JVMs, and MBean servers, you can specify a chart that displays:

- The amount of resources the service is using from a resource pool relative to the amount of resources available.
- Runtime statistics from each JVM within the service. The amount of information that is available depends on what the JVM provides.

Monitoring Service Performance

The Administration Console can display CPU and memory charts that indicate the performance of a service. For services managed by Plain Agents, you need to start up the JVM's management server when you start the service's processes. For information about how to do this, see [“JVM Arguments for Processes” on page 5-10](#).

Note: The service must be using a JRockit JVM.

For more information, see [“Monitor Resources”](#) in the *WLOC Administration Console Online Help*.

Viewing Events

By default, events are displayed at the bottom of the Administration Console page in the **Task and Events** viewer. This viewer displays up to the last 20 events that have occurred and is active no matter which tab is currently selected in the console. This viewer provides the same ability to filter messages that appear in it as provided on the Events page. You can also create custom viewing tabs with unique filters.

You can also view events using the **Events** tab in the Console.

In addition, you may define Administrative policies that will cause the Console to generate console messages based on some constraint. For additional information, see [“Administrative Policies” on page 6-6](#).

Silent Mode Configurations

This appendix provides information about silent-mode configurations.

- [“Running the Configuration Wizard in Silent-Mode” on page A-1](#)
- [“Plain Agent Template XML File” on page A-2](#)
- [“ESX Agent Template XML File” on page A-5](#)
- [“Controller Template XML File” on page A-10](#)

Running the Configuration Wizard in Silent-Mode

You may create an XML-formatted template that contains your configuration settings and then run the WLOC Configuration Wizard in silent mode so that it uses the template values without requiring you to complete the GUI windows.

To run the WLOC Configuration Wizard in silent-mode, follow these steps:

1. Create a XML file containing the configuration values for the WLOC component. The name of this file is arbitrary, but the examples shown below use the name `silent_config_input.xml`.

The XML files are described in the following sections:

- [“Plain Agent Template XML File” on page A-2](#)
- [“ESX Agent Template XML File” on page A-5](#)
- [“Controller Template XML File” on page A-10](#)

2. Launch the Configuration Wizard by entering one of the following commands:

On Windows:

```
BEA_HOME\wloc_10.3\common\bin\config.cmd -mode=silent  
-silent_xml=path\silent_config_input.xml -log=silent_config.log
```

On UNIX or Linux:

```
BEA_HOME/wloc_10.3/common/bin/config.sh -mode=silent  
-silent_xml=path/silent_config_input.xml -log=silent_config.log
```

where *BEA_HOME* is the BEA home directory in which you installed the software and *path* is the fully-qualified path to the *silent_config_input.xml* file.

Plain Agent Template XML File

The following XML document can be copied and used to create a silent-mode configuration template for a plain Agent. After copying the document, you must replace the italicized values with the actual values in your environment, as described in [“Plain Agent Configuration Values” on page A-3](#).

```
<?xml version="1.0" encoding="UTF-8"?>  
<domain-template-descriptor>  
<input-fields>  
  <data-value name="AGENT_DIR" value="c:/bea/user_projects/agent1" />  
  <data-value name="Agent.name" value="agent1" />  
  <data-value name="Agent.host" value="agent1.abc.com" />  
  <data-value name="Agent.port" value="8001" />  
  <data-value name="Agent.securePort" value="8002" />  
  <data-value name="Agent.encryption.password" value="changeit" />  
  <data-value name="Agent.useSecureConnections" value="false" />  
  <data-value name="Logging.fileSeverity" value="Info" />  
  <data-value name="Logging.stdoutSeverity" value="Info" />  
  <data-value name="Logging.baseFileName" value="./logs/Agent.log" />  
  <data-value name="Logging.fileRotationDir" value="./logs/logrotDir" />  
  <data-value name="Agent.internalidentity.keystorePassword" value="changeit"/>  
  <data-value name="Agent.internaltrust.keystorePassword" value="changeit" />  
  <data-value name="Agent.type" value="plainAgent" />  
  <data-value name="PlainAgent.name" value="agent1 pool" />  
  <data-value name="PlainAgent.description" value="agent1 pool" />  
  <data-value name="PlainAgent.cpuCapacity" value="2000" />  
  <data-value name="PlainAgent.diskCapacity" value="1024" />  
  <data-value name="PlainAgent.stdoutDir" value="./managed-stdout-stderr" />  
  <data-value name="PlainAgent.stderrDir" value="./managed-stdout-stderr" />  
</input-fields>  
</domain-template-descriptor>
```

Plain Agent Configuration Values

The following table describes the configuration values needed in the Plain Agent template XML file.

Table A-1 Plain Agent Configuration Values

Name	Description
AGENT_DIR	Complete path to the Agent directory. <pre><data-value name="AGENT_DIR" value="c:/bea/user_projects/agent1" /></pre>
Agent.name	The name of the Agent. <pre><data-value name="Agent.name" value="agent1" /></pre>
Agent.host	The fully-qualified host name where the Agent resides. <pre><data-value name="Agent.host" value="agent1.abc.com" /></pre>
Agent.port	The Agent's HTTP port number used when communicating with the Controller in unsecure mode. <pre><data-value name="Agent.port" value="8001" /></pre> <p>In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>
Agent.securePort	The Agent's HTTPS port number used when communicating with the Controller in secure mode. <pre><data-value name="Agent.securePort" value="8002" /></pre> <p>In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.</p>
Agent.encryption.password	A passphrase used to apply encryption beyond the Security Mode setting to encrypt certain sensitive data passed between the Controller and Agent. The password must be a minimum of 8 characters. This value will be encrypted. <pre><data-value name="Agent.encrypted.password" value="changeit" /></pre> <p>If Security Mode is unsecure (<code>false</code>), this setting will still encrypt the most sensitive data.</p>

Table A-1 Plain Agent Configuration Values (Continued)

Name	Description
<code>Agent.useSecureConnections</code>	<p>The security mode to use for connections with the Controller.</p> <p>false — Unsecure mode. Use in development environments only. true — Secure mode. Should be used in production environments.</p> <pre><data-value name="Agent.useSecureConnections" value="false" /></pre> <p>Specifying secure mode (true) ensures confidentiality and integrity of the communication and requires setting up trust between the Controller and the Agent. For details, see “Secure Communications” on page 7-4.</p> <p>Note: The Controller and all Agents must be set to the same security mode.</p>
<code>Logging.fileSeverity</code>	<p>The severity level of events to log.</p> <pre><data-value name="Logging.fileSeverity" value="Info" /></pre> <p>In the order of severity from least severe to most severe, the log levels are: Trace, Debug, Info, Notice, Warning, Error, Critical, Alert, Emergency</p> <p>Severity levels are inclusive. When set to Info, the log will include Notice, Warning, Error, Critical, Alert, And Emergency events.</p>
<code>Logging.stdoutSeverity</code>	<p>The severity level of events that should be written to stdout.</p> <pre><data-value name="Logging.stdoutSeverity" value="Info" /></pre> <p>Uses same event levels as the log file.</p>
<code>Logging.baseFileName</code>	<p>The log file directory and name.</p> <pre><data-value name="Logging.baseFileName" value="./logs/Agent.log" /></pre>
<code>Logging.fileRotationDir</code>	<p>The log rotation directory.</p> <pre><data-value name="Logging.fileRotationDir" value="./logs/logrottdir" /></pre>
<code>Agent.internalidentity.keystorePassword</code>	<p>The passphrase for the Agent’s internal identity keystore. This value will be encrypted.</p> <pre><data-value name="Agent.internalidentity.keystorePassword" value="changeit" /></pre>

Table A-1 Plain Agent Configuration Values (Continued)

Name	Description
Agent.internaltrust.keystorePassword	The passphrase for the Agent's internal trust keystore. This value will be encrypted. <data-value name="Agent.internaltrust.keystorePassword" value="changeit" />
Agent.type	The type of Agent. Must be set to plainAgent. <data-value name="Agent.type" value="plainAgent" />
PlainAgent.name	Name of the resource pool managed by the Agent. <data-value name="PlainAgent.name" value="agent1 pool" />
PlainAgent.description	An arbitrary description of the resource pool. <data-value name="PlainAgent.description" value="agent1 pool" />
PlainAgent.cpuCapacity	CPU capacity (in normalized megahertz) available to the resource pool. <data-value name="PlainAgent.cpuCapacity" value="2000" />
PlainAgent.diskCapacity	The disk capacity available to the resource pool. <data-value name="PlainAgent.diskCapacity" value="1024" />
PlainAgent.stdoutDir	Directory for the JVM stdout output stream. <data-value name="PlainAgent.stdoutDir" value="./managed-stdout-stderr" />
PlainAgent.stderrDir	Directory for the JVM Stderr output stream. <data-value name="PlainAgent.stderrDir" value="./managed-stdout-stderr" />

ESX Agent Template XML File

The following XML document can be copied and used to create a silent-mode configuration template for an ESX Agent. After copying the document, you must replace the italicized values with the actual values in your environment, as described in [“ESX Agent Configuration Values”](#) on page A-7.

```
<?xml version="1.0" encoding="UTF-8"?>
<domain-template-descriptor>
<input-fields>
```

The following elements are the same as those used for Plain Agents. For descriptions, see [“Plain Agent Configuration Values” on page A-3](#).

```
<data-value name="AGENT_DIR" value="c:\bea\user_projects\agent1" />
<data-value name="Agent.name" value="agent1" />
<data-value name="Agent.host" value="agent1.abc.com" />
<data-value name="Agent.port" value="8001" />
<data-value name="Agent.securePort" value="8002" />
<data-value name="Agent.encryption.password" value="changeit" />
<data-value name="Agent.useSecureConnections" value="false" />
<data-value name="Logging.fileSeverity" value="Info" />
<data-value name="Logging.stdoutSeverity" value="Info" />
<data-value name="Logging.baseFileName" value="./logs/Agent.log" />
<data-value name="Logging.fileRotationDir" value="./logs/logrotodir" />
<data-value name="Agent.internalidentity.keystorePassword" value="changeit"/>
<data-value name="Agent.internaltrust.keystorePassword" value="changeit" />
```

The following elements are described in [Table A-2](#).

```
<data-value name="Agent.type" value="esxagent" />
<data-value name="EsxAgent.name" value="vmware-agent" />
<data-value name="EsxAgent.description" value="VMWare ESX Information" />
<data-value name="EsxAgent.vcHost" value="vchost.abc.com" />
<data-value name="EsxAgent.username" value="admin66" />
<data-value name="EsxAgent.password" value="d0rwssap" />
<data-value name="EsxAgent.vmwarePool.dataCenter" value="datacenter1" />
<data-value name="EsxAgent.vmwarePool.computeResource" value="esxHost.abc.com" />
<data-value name="EsxAgent.vmwarePool.resourcePool" value="WLOCpool1" />
<data-value name="EsxAgent.vmwarePool.description" value="WLOCpool1" />
<data-group name="vmware-networks">
<data-element name="networks">
  <data-value name="name" value="VM Network"/>
  <data-value name="ipAddresses" value="10.244.22.86,10.170.43.81"/>
  <data-value name="description" value="WLOC VM Network"/>
  <data-value name="gateway" value="192.18.128.1"/>
  <data-value name="netMask" value="255.255.248.0"/>
  <data-value name="dnsServers" value="10.40.0.86,10.40.0.87"/>
  <data-value name="domainName" value="abc.com"/>
</data-element>
</data-group>
<data-group name="available-software">
<data-element name="iso-software">
  <data-value name="name" value="WLSVE9.2.2-ISO"/>
  <data-value name="description" value="WLSVE9.2.2-ISO"/>
```

```

    <data-value name="path" value="[SAN-store] wlsve/wlsve922.iso" />
    <data-value name="version" value="1.1" />
  </data-element>
  <data-element name="nfs-software">
    <data-value name="name" value="bea_home" />
    <data-value name="description" value="bea_home on NFS" />
    <data-value name="path"
value="192.18.128.67:/LOC/bea/bea.home,uid=55004,gid=10000" />
    <data-value name="mode" value="EXCLUSIVE" />
  </data-element>
</data-group>
</input-fields>
</domain-template-descriptor>

```

ESX Agent Configuration Values

The following table describes the values needed in the ESX Agent template XML file.

Table A-2 ESX Agent Configuration Values

Name	Description
Agent.type	The type of Agent. Must be set to esxagent. <data-value name="Agent.type" value="esxagent" />
EsxAgent.name	The ESX Agent name. <data-value name="EsxAgent.name" value="vmware-agent" />
EsxAgent.description	An arbitrary description of the ESX Agent. <data-value name="EsxAgent.description" value="VMWare ESX Information" />
EsxAgent.vcHost	The Virtual Center host name. <data-value name="EsxAgent.vcHost" value="vchost.abc.com" />
EsxAgent.username	The Virtual Center administrator username. <data-value name="EsxAgent.username" value="admin66" />
EsxAgent.password	The Virtual Center administrator's password. <data-value name="EsxAgent.password" value="dorwssap" />

Table A-2 ESX Agent Configuration Values (Continued)

Name	Description
EsxAgent.vmwarePool.dataCenter	<p>The name of the Datacenter containing the resource pool managed by this Agent.</p> <pre><data-value name="EsxAgent.vmwarePool.dataCenter" value="datacenter1" /></pre>
EsxAgent.vmwarePool.computeResource	<p>The ESX Server host or cluster name.</p> <pre><data-value name="EsxAgent.vmwarePool.computeResource" value="esxHost.abc.com" /></pre>
EsxAgent.vmwarePool.resourcePool	<p>The Resource Pool containing the LiquidVM instances to be managed by this Agent.</p> <pre><data-value name="EsxAgent.vmwarePool.resourcePool" value="WLOCPool1" /></pre>
EsxAgent.vmwarePool.description	<p>An arbitrary description of the resource pool.</p> <pre><data-value name="EsxAgent.vmwarePool.description" value="WLOCPool1" /></pre>
VMNetworkName	<p>The Virtual Machine Port Group to which the LiquidVM instance is assigned. For a cluster of ESX hosts, all hosts must have a Virtual Machine Port Group with the same name and the group must be mapped a physical adapter connected to the same physical network.</p> <pre><data-value name="name" value="VM Network" /></pre>
ipAddresses	<p>One or more IP addresses reserved for the LiquidVM instances. Specify multiple addresses on the same line separated using a comma (.).</p> <pre><data-value name="ipAddresses" value="10.244.22.86,10.170.43.81" /></pre>
description	<p>An arbitrary description.</p> <pre><data-value name="description" value="WLOC VM Network" /></pre>
gateway	<p>The Gateway address used by the LiquidVM instance. This can be determined from the physical network adapter to which the Virtual Machine Port Group is mapped. If not specified, the LiquidVM instance will use the default gateway based on its IP address.</p> <pre><data-value name="gateway" value="192.18.128.1" /></pre>

Table A-2 ESX Agent Configuration Values (Continued)

Name	Description
netMask	<p>The Netmask used by the LiquidVM instance. This can be determined from the physical network adapter to which the Virtual Machine Port Group is mapped. If not specified, the LiquidVM instance will use the default netmask.</p> <pre><data-value name="netMask" value="255.255.248.0" /></pre>
dnsServers	<p>The primary and alternate DNS Server used by the LiquidVM instance. Specify multiple addresses on the same line separated using a comma (.). The LiquidVM instance cannot use remote DNS lookup if this is not specified.</p> <pre><data-value name="dnsServers" value="10.40.0.86,10.40.0.87" /></pre>
domainName	<p>The domain name used by the LiquidVM instance.</p> <pre><data-value name="domainName" value="abc.com" /></pre>
iso-software	<p>Use one or more instances of the XML structure below to define each ISO that can be used by virtual machines created in the pool managed by this Agent. These definitions must be under the <code><data-group name="available-software"></code> element.</p> <p>Example:</p> <pre><data-element name="iso-software"> <data-value name="name" value="WLSVE9.2.2-ISO" /> <data-value name="description" value="My iso" /> <data-value name="path" value="[SAN-store] wlsve/wlsve922.iso"/> <data-value name="version" value="1.1"/> </data-element></pre> <p>Where:</p> <p><i>name</i>— ISO name</p> <p><i>description</i>— Arbitrary description</p> <p><i>path</i>— Location of the ISO software, including the datastore. Syntax: [datastore] /path/filename</p> <p><i>version</i>— VMWare version (1.0, 1.1, or 1.2)</p>

Table A-2 ESX Agent Configuration Values (Continued)

Name	Description
nfs-software	<p>Use one or more instances of the XML structure below to define each NFS mount point that can be accessed by virtual machines created in the pool managed by this Agent. These definitions must be under the <data-group name="available-software"> element.</p> <p>Example:</p> <pre><data-element name="nfs-software"> <data-value name="name" value="bea_home" /> <data-value name="description" value="bea home on nfs" /> <data-value name="path" value="192.18.128.67:/LOC/bea/bea.home,uid=55004,gid=1 0000" /> <data-value name="mode" value="EXCLUSIVE" /> </data-element></pre> <p>Where:</p> <p>name — NFS name</p> <p>description — Arbitrary description</p> <p>path — The NFS share path using the following syntax:</p> <pre><ip_address>:<path>,uid=<uid_num>,gid=<gid_num></pre> <p> <ip_address> — IP Address of the NFS share host</p> <p> <path> — path to the share</p> <p> <uid_num> — userid number</p> <p> <gid_num> — group id number</p> <p>mode — this value must be EXCLUSIVE</p>

Controller Template XML File

The following XML document can be copied and used to create a silent-mode configuration template for a Controller. After copying the document, you must replace the italicized values with the actual values in your environment, as described in [“Controller Configuration Values” on page A-12](#).

```
<?xml version="1.0" encoding="UTF-8"?>
<domain-template-descriptor>
<input-fields>
  <data-value name="CONTROLLER_DIR" value="c:/bea/user_projects/controller" />
  <data-value name="Controller.host" value="adminbox.east.example.com" />
  <data-value name="Controller.console.port" value="9001" />
```

```

<data-value name="Controller.console.securePort" value="9002" />
<data-value name="Controller.internal.port" value="9003" />
<data-value name="Controller.internal.securePort" value="9004" />
<data-value name="Controller.consoleMode" value="BOTH" />
<data-value name="Controller.useSecureConnections" value="false" />
<data-value name="Logging.fileSeverity" value="Info" />
<data-value name="Logging.stdoutSeverity" value="Info" />
<data-value name="Logging.baseFileName" value="./logs/controller.log" />
<data-value name="Logging.fileRotationDir" value="./logs/logrotmdir"/>
<data-value name="Notification.smtp.enabled" value="true"/>
<data-value name="Notification.smtp.toAddress" value="WLOC@abc.com"/>
<data-value name="Notification.smtp.fromAddress" value="WLOCadmin@abc.com"/>
<data-value name="Notification.smtp.smtpServer" value="smtpserver.abc.com"
/>
  <data-value name="Notification.jms.enabled" value="true"/>
  <data-value name="Notification.jms.destinationJndiName"
value="LOC_Queue_Notification" />
  <data-value name="Notification.jms.connectionFactoryJndiName"
value="LOC_QueueConnectionFactory" />
  <data-value name="Notification.jms.jndiProperties.initialFactory"
value="weblogic.jndi.WLInitialContextFactory" />
  <data-value name="Notification.jms.jndiProperties.providerUrl"
value="iiop://182.76.123.21:9911"/>
  <data-value name="Notification.jms.jndiProperties.securityPrincipal"
value="system"/>
  <data-value name="Notification.jms.jndiProperties.password" value="system"/>
  <data-value name="Notification.jmx.enabled" value="true"/>
  <data-value name="Notification.snmp.enabled" value="true"/>
  <data-value name="Notification.snmp.agent.host" value="abc.abc2.com"/>
  <data-value name="Notification.snmp.agent.port" value="2002"/>
  <data-value name="Notification.snmp.trapDestinations.destination.host"
value="abc.abc3.com"/>
  <data-value name="Notification.snmp.trapDestinations.destination.port"
value="1642"/>
  <data-value name="Notification.snmp.agent.trapVersion" value="SNMPv2"/>
  <data-group name="agents">
    <data-element name="agent">
      <data-value name="name" value="agent1"/>
      <data-value name="host" value="agent1.abc.com"/>
      <data-value name="port" value="8001"/>
      <data-value name="secure-port" value="8002"/>
      <data-value name="state" value="Enabled"/>
      <data-value name="password" value="changeit"/>
    </data-element>
  </data-group>
  <data-value name="LoginInfo.username" value="WLOCBootUser" />
  <data-value name="LoginInfo.password" value="changeit" />
  <data-value name="Controller.demoidentity.keystorePassword" value="changeit"
/>

```

```

    <data-value name="Controller.internalidentity.keystorePassword"
value="changeit" />
    <data-value name="Controller.internaltrust.keystorePassword"
value="changeit" />
    <data-value name="Controller.publicKeyFile" value=".keys/id_rsa.pub" />
  </input-fields>
</domain-template-descriptor>

```

Controller Configuration Values

The following table describes the values needed in the Controller template XML file.

Table A-3 Controller Configuration Values

Name	Description
CONTROLLER_DIR	Complete path to the Controller directory. <data-value name="CONTROLLER_DIR" value="c:/bea/user_projects/controller" />
Controller.host	Fully-qualified host name of the Controller machine. <data-value name="Controller.host" value="adminbox.east.example.com" />
Controller.console.port	HTTP port for the WLOC Administration Console <data-value name="Controller.console.port" value="9001" />
Controller.console.securePort	HTTPS port for the WLOC Administration Console. <data-value name="Controller.console.securePort" value="9002" />
Controller.internal.port	Port used by Agents for unsecure internal communication with the Controller. <data-value name="Controller.internal.port" value="9003" />
Controller.internal.securePort	Port used by Agents for secure internal communication with the Controller. <data-value name="Controller.internal.securePort" value="9004" />
Controller.consoleMode	Enter SECURE (HTTPS), UNSECURE (HTTP), or BOTH (both HTTP & HTTPS) to specify how clients may connect to the Administration Console: <data-value name="Controller.consoleMode" value="BOTH" />

Table A-3 Controller Configuration Values (Continued)

Name	Description
Controller.useSecureConnections	<p>Select one of the following to specify the security level to be used for internal communications between WLOC components.</p> <p>false — Unsecure mode. Uses HTTP without SSL and guarantee of confidentiality and integrity. Use in development environments only.</p> <p>true — Secure mode. Uses HTTPS providing message confidentiality and integrity. Should be used in production environments.</p> <p>NOTE: All Agents must use the same security mode as that used by the Controller with which they communicate.</p> <pre><data-value name="Controller.useSecureConnections" value="false" /></pre>
Logging.fileSeverity	<p>Level of events to log. In order of severity from least to most severe, log levels are: Trace, Debug, Info, Notice, Warning, Error, Critical, Alert, Emergency</p> <pre><data-value name="Logging.fileSeverity" value="Info" /></pre>
Logging.stdoutSeverity	<p>Level of events to write to stdout. Uses same event levels as the log file.</p> <pre><data-value name="Logging.stdoutSeverity" value="Info" /></pre>
Logging.baseFileName	<p>Log file and directory relative to the Controller directory.</p> <pre><data-value name="Logging.baseFileName" value="./logs/controller.log" /></pre>
Logging.fileRotationDir	<p>Rotation directory for controller logs.</p> <pre><data-value name="Logging.fileRotationDir" value="./logs/logrottdir" /></pre>
Notification.smtp.enabled	<p>Specify true to enable SMTP notification; otherwise, specify false.</p> <pre><data-value name="Notification.smtp.enabled" value="true" /></pre>
Notification.smtp.toAddress	<p>E-mail address to which notifications should be sent.</p> <pre><data-value name="Notification.smtp.toAddress" value="WLOC@abc.com" /></pre>

Table A-3 Controller Configuration Values (Continued)

Name	Description
Notification.smtp.fromAddress	E-mail address from which notifications should be sent. <data-value name="Notification.smtp.fromAddress" value="WLOCadmin@abc.com" />
Notification.smtp.smtpServer	SMTP mail server through which to send notifications. <data-value name="Notification.smtp.smtpServer" value="smtpserver.abc.com" />
Notification.jms.enabled	Specify <i>true</i> to enable JMS notification; otherwise, specify <i>false</i> . <data-value name="Notification.jms.enabled" value="true" />
Notification.jms.destinationJndiName	Destination JNDI name. <data-value name="Notification.jms.destinationJndiName" value="LOC_Queue_Notification" />
Notification.jms.connectionFactoryJndiName	Name of the JNDI connection factory. <data-value name="Notification.jms.connectionFactoryJndiName" value="LOC_QueueConnectionFactory" />
Notification.jms.jndiProperties.initialFactory	Fully-qualified package and class of the initial factory. <data-value name="Notification.jms.jndiProperties.initialFactory" value="weblogic.jndi.WLInitialContextFactory" />
Notification.jms.jndiProperties.providerUrl	JNDI provider URL. <data-value name="Notification.jms.jndiProperties.providerUrl" value="iiop://192.18.134.173:9901" />
Notification.jms.jndiProperties.securityPrincipal	JNDI user name. <data-value name="Notification.jms.jndiProperties.securityPrincipal" value="system" />

Table A-3 Controller Configuration Values (Continued)

Name	Description
Notification.jms.jndiProperties.password	JNDI user's password. <data-value name="Notification.jms.jndiProperties.password" value="system" />
Notification.jmx.enabled	Specify <i>true</i> to enable JMX notification; otherwise, specify <i>false</i> . <data-value name="Notification.jmx.enabled" value="true" />
Notification.snmp.enabled	Specify <i>true</i> to enable SNMP notification; otherwise, specify <i>false</i> . <data-value name="Notification.snmp.enabled" value="true" />
Notification.snmp.agent.host	Hostname of the SNMP agent. <data-value name="Notification.snmp.agent.host" value="abc.abc2.com" />
Notification.snmp.agent.port	Port number of the SNMP agent. <data-value name="Notification.snmp.agent.port" value="2002" />
Notification.snmp.trapDestinations.destination.host	DNS name or IP address of SNMP manager machine. <data-value name="Notification.snmp.trapDestinations.destination.host" value="abc.abc3.com" />
Notification.snmp.trapDestinations.destination.port	Listening port of the SNMP manager. <data-value name="Notification.snmp.trapDestinations.destination.port" value="1642" />
Notification.snmp.agent.trapVersion	SNMP version (SNMPv1 or SNMPv2) <data-value name="Notification.snmp.agent.trapVersion" value="SNMPv2" />

Table A-3 Controller Configuration Values (Continued)

Name	Description
Agent Information '<data-element name="agent">	<p>Use one or more instances of the XML structure below to define each Agent. This structure must be encapsulated under the <data-group name="agents"> element.</p> <p>Example:</p> <pre><data-group name="agents"> <data-element name="agent"> <data-value name="name" value="agent1"/> <data-value name="host" value="123.54.432.99"/> <data-value name="port" value="8001"/> <data-value name="secure-port" value="8002"/> <data-value name="state" value="Enabled"/> <data-value name="password" value="changeit"/> </data-element> </data-group></pre> <p>Where:</p> <p>name — Agent name.</p> <p>host — Fully-qualified host name or IP address of the Agent machine.</p> <p>port — HTTP port on which to access the Agent.</p> <p>secure-port — HTTPS port on which to access the Agent.</p> <p>state — One of Enabled, Connected, or Disconnected.</p> <p>password — Agent's current passphrase. Communication between the Controller and Agent will fail unless the entry matches the Agent's current passphrase.</p>
LoginInfo.username	<p>The username for logging into the Administration Console.</p> <pre><data-value name="LoginInfo.username" value="WLOCBootUser" /></pre>
LoginInfo.password	<p>The password for the above user.</p> <pre><data-value name="LoginInfo.password" value="changeit" /></pre>
Controller.demoidentity.keystorePassword	<p>The Controller identity keystore password used for connections to the console.</p> <pre><data-value name="Controller.demoidentity.keystorePassword" value="changeit" /></pre>

Table A-3 Controller Configuration Values (Continued)

Name	Description
Controller.internalidentity.keystorePassword	The Controller internal identity keystore password. <data-value name="Controller.internalidentity.keystorePassword" value="changeit" />
Controller.internaltrust.keystorePassword	The Controller identity trust keystore password. <data-value name="Controller.internaltrust.keystorePassword" value="changeit" />
Controller.publicKeyFile	The path and name of the SSH public key file to be passed to Agents that are creating LVM instances with SSH enabled. <data-value name="Controller.publicKeyFile" value="C:/sshKey/id_rsa.pub" />

