BEA WebLogic Operations Control

Configuration Guide

Version 1.0
Revised: April 4, 2008

Introduction

Overview

Configuring the Controller and Agents

Starting and Stopping the Controller and Agents

Configuring Services

Defining Policies

Configuring Security

Logging, Auditing, and Monitoring

Silent Mode Configurations

CHAPTER 1

Introduction

This document describes configuration tasks associated with using WebLogic Operations Control™ (WLOC) 1.0 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" 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

The WLOC documentation set includes the following:

- *Installation Guide*—Describes how to install and uninstall the WLOC components.
- *LiquidVM User Guide*—Describes how to use LiquidVM to create and deploy virtualized Java software appliances directly onto virtualized server resources.
- *WLOC Administration Console Help*—The online help for WLOC's graphical user interface. You can access the WLOC Administration Console Help either by clicking the Help link in the upper right corner of the Administration Console, or at `http://edocs.bea.com/wloc/docs10/ConsoleHelp`.
- *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.

CHAPTER 2

Overview

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

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**

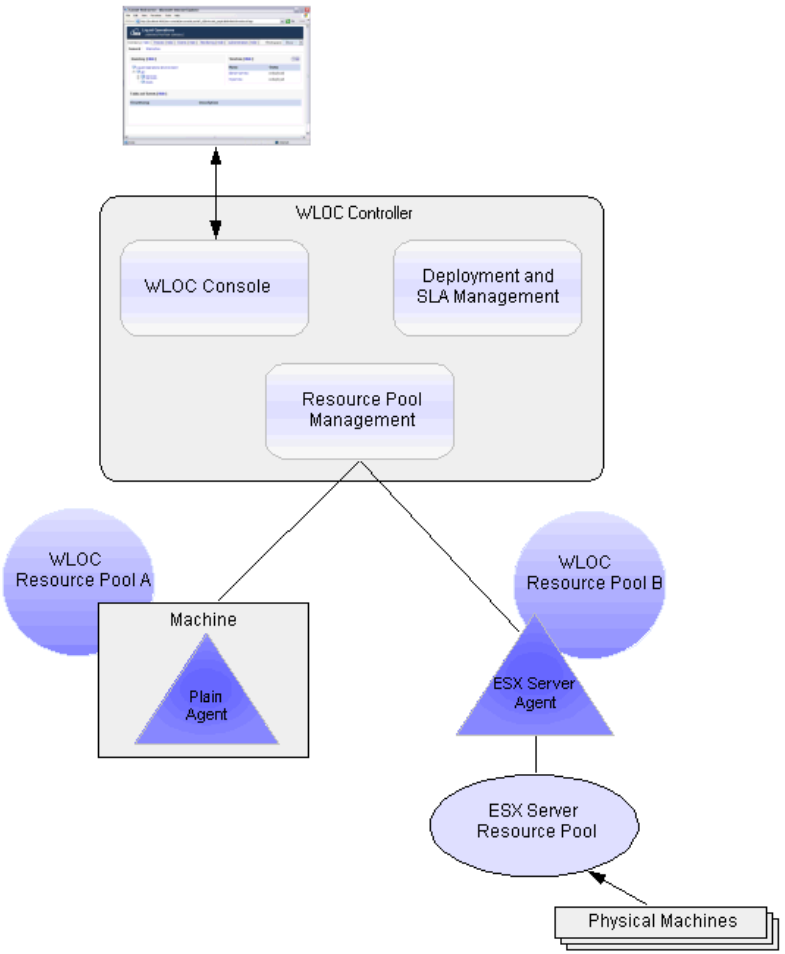
  WLOC Agents provide information about the environment to the Controller, start and stop processes, and invoke other actions at the Controller's request. A plain agent gathers data and manages processes on a single physical machine. An ESX Agent communicates with the VMware Virtual Center to gather data 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 3-14. In addition, instructions on performing silent mode Controller configurations are located in Appendix A.

- **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 3-7. In

addition, instructions on performing silent mode Agent configurations are located in Appendix A.

- **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 are described in "Configuring Services" 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.

CHAPTER 3

Configuring the Controller and Agents

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

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 is located in the following directory:

BEA_HOME`\`*WLOC_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 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 managed 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 ESX 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**WLOC_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 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 sections provides step-by-step instructions for running the configuration wizard in GUI mode. For performing silent-mode configurations, see Appendix A.

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 described in Table 3-1.

Table 3-1 Invoking the Agent Configuration Wizard

Platform	Command
Windows	*BEA_HOME**WLOC_HOME*\common\bin\config.cmd **Note:** You may also select **WebLogic Operations Control 1.0>WLOC Configuration Wizard** from the Windows Start Menu.
UNIX or Linux	*BEA_HOME*/*WLOC_HOME*/common/bin/config.sh

2. On the Choose Agent or Controller window, select **Create a new Agent...** and click **Next**.
3. On the Enter Agent Directory Location window, accept the default location or specify a different directory and click **Next**.
4. On the Configure Agent Connection Details window, complete the fields as described in Table 3-2.

Table 3-2 Configure Agent Connection Details

Field	Description
Agent Name	Specify a unique Agent name. **Note:** Managing multiple Agents through the Administration Console requires that each Agent has a unique name.

Table 3-2 Configure Agent Connection Details

Agent Host	Fully-qualified host name where the Agent resides; example: `agentbox.east.example.com`.
Agent Port	Agent's HTTP port number used when communicating with the Controller in unsecure mode; default: `8001`. **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.
Agent Secure Port	Agent's HTTPS port number used when communicating with the Controller in secure mode; default: `8002`. **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.
Agent Passphrase/ Confirm Agent Passphrase	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. If Security Mode is Unsecure, this setting will still encrypt the most sensitive data. **Note:** This passphrase must be entered when adding the Agent to the Controller or communication between the Controller and Agent will fail. For development environments, it is sufficient to accept the defaults. You do not need to know these passphrases.
Security Mode	Select a security mode for connections with the Controller. Unsecure — sufficient for development. Secure — should be used for production environments. Secure mode ensures confidentiality and integrity of the communication and requires setting up trust as an explicit step between the Controller and the Agent. **NOTE:** Both the Controller and Agent must be set to the same security mode.

5. On the Configure Agent Logging window, complete the fields as described in Table 3-3.

Table 3-3 Configure Agent Logging

Field	Description
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: *BEA_HOME*\user_projects\agent1\logs\Agent.log

6. On 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 thew window.

Table 3-4 Configure Agent Keystore Passwords

Environment	Description
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. On the Configure Agent Type window, select **Plain Agent** and click **Next**.

8. On 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)

Field	Description
Resource Pool Name	Enter 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	Specify a directory under *user_projects* for the JVM stdout output stream. Default: *BEA_HOME*\user_projects\agent1\stdout
Stderr directory	Specify a directory under user_projects for Stderr the output stream. Default: *BEA_HOME*\user_projects\agent1\stderr

9. On the Configure Plain Agent (2 of 2) window, specify each available software instance by clicking **Add** and completing each field as described in Table 3-6. When done, 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)

Field	Description
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 a Weblogic domain, specify the path to the domain.

10. On the Create Agent Configuration window, click **Create**. Progress messages then appear.
11. When it becomes active, click the **Done** button.

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 described in Table 3-7.

Table 3-7 Invoking the Configuration Wizard

Platform	Command
Windows	*BEA_HOME**WLOC_HOME*\common\bin\config.cmd **Note:** You may also select **WebLogic Operations Control 1.0>WLOC Configuration Wizard** from the Windows Start Menu.
UNIX or Linux	*BEA_HOME*/*WLOC_HOME*/common/bin/config.sh

2. On the Choose Agent or Controller window, select **Create a new Agent...** and click **Next**.
3. On the Enter Agent Directory Location window, accept the default location or specify a different directory under `user_projects` and click **Next**.
4. On the Configure Agent Connection Details window, complete the fields as described in Table 3-8.

Table 3-8 Configure Agent Connection Details

Field	Description
Agent Name	Agent name. The Administration Console displays this as the Agent name.
Agent Host	Fully-qualified host name where the Agent resides; example: `agentbox.east.example.com`.

Table 3-8 Configure Agent Connection Details

Field	Description
Agent Port	HTTP port number used when the Agent and Controller are connecting in unsecure mode; default: `8001`. **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.
Agent Secure Port	HTTPS port number used when the Agent and Controller are connecting in secure mode; default: `8002`. **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.
Agent Passphrase/ Confirm Agent Passphrase	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. Even if Security Mode is Unsecure, this setting will encrypt the most sensitive data. **Note:** This passphrase must be entered when adding the Agent to the Controller or communication between the Controller and Agent will fail. For development environments, it is sufficient to accept the defaults. You do not need to know these passphrases.
Security Mode	Select a security mode for connections with the Controller. `Unsecure` — sufficient for development. `Secure` — should be used for production environments. Secure mode ensures confidentiality and integrity of the communication and requires setting up trust as an explicit step between the Controller and the Agent. "Securing Controller to Agents Communication" on page 7-7. **NOTE:** Both the Controller and Agent must be set to the same security mode.

5. On the Configure Agent Logging window, complete the fields as described in Table 3-9.

Table 3-9 Configure Agent Logging

Field	Description
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: *BEA_HOME*`\user_projects\agent1\logs\Agent.log`

6. On 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

Environment	Description
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. On the Configure Agent Type window, select **ESX Agent** and click **Next**.

8. On 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)

Field	Description
Name	Accept the default name or enter a different one.
Description	Accept or modify the default.

9. On 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)

Field	Description
Virtual Center Host	The IP address or name of the VirtualCenter Server.
Username	The user name of a VirtualCenter administrator.
Password/Confirm Password	The administrator's password.
VMWare SSL Certificate	Select the **Connect to the Virtual Center to retrieve SSL certificate** checkbox. 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.
ESX Agent Configuration Type - Dynamic or Static	Select the **Configure the ESX Agent dynamically...** checkbox 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. Accept the default selection of the **Use secure connection...** checkbox if the Virtual Center has been configured to use secure connections. Clear the checkbox if the Virtual Center uses only insecure connections.

If select the **Configure the ESX Agent dynamically...** checkbox, 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. On 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)

Field	Description
Data Center Name	Name of the Datacenter that contains the resource pool managed by this Agent.
Compute Resource	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. On 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)

Field	Description
Name	The Virtual Machine Port Group to which the LiquidVM instance is assigned. 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. **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.
JVM IP Address	One or more IP addresses reserved for the LiquidVM instances. Specify multiple addresses on the same line separated using a comma (,). Example: `182.343.121.122,182.343.121.123,182.343.121.122` A LiquidVM instance will use only one of the IP addresses specified.
Description	An arbitrary description.
Gateway IP Address	The Gateway address used by the LiquidVM instance. The address 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 its IP address.
Netmask	The Netmask used by the LiquidVM instance. The Netmask 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.
DNS Server Address	The primary and alternate DNS Server used by the ESX Server host. Specify multiple addresses on the same line separated using a comma (,). Example: `10.344.22.86,10.170.43.81` The LiquidVM instance cannot use remote DNS lookup if this is not specified.
Domain Name	The domain name used by the LiquidVM instance.

12. On 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)

Field	Description
Name	The ISO name.
Description	An arbitrary description.

ISO Software Path	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.
	To specify it manually, the syntax is `[datastore] /path/filename`
	Examples:
	`[storage1 ] /wlsve922.iso`
	`[storage1 ] /wlsve/wlsve922.iso`

Version	The VMWare version (`1.0`, `1.1`, or `1.2`).

13. On the Configure ESX Agent (6 of 6) window, define each NFS share are 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)

Field	Description
Name	The NFS share name.

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

Field	Description
Description	An arbitrary description.

NFS Software Path	The NFS share path using the following syntax:
	`<ip_address>:<path>,uid=<uid_num>,gid=<gid_num>`
	where
	`<ip_address>` — IP Address of the NFS share host
	`<path>` — path to the share
	`<uid_num>` — userid number
	`<gid_num>` — group id number
	Example:
	`182.34.234.92:\WLOC\shares\domainw,uid=10947,gid=3498`

14. On 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 console's help system.

Controller Configurations

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 wizard using one of the commands shown in Table 3-17.

Table 3-17 Invoking the Controller Wizard

Platform	Command
Windows	*BEA_HOME**WLOC_HOME*\common\bin\config.cmd **Note:** You may also select **WebLogic Operations Control 1.0>WLOC Configuration Wizard** from the Windows Start Menu.
UNIX or Linux	*BEA_HOME*/*WLOC_HOME*/common/bin/config.sh

2. On the Welcome window, click **Next**.
3. On the Choose Controller or Agent window, select **Create the Controller...** and click **Next**.
4. On the Enter Controller Directory Location window, specify the location and click **Next**.
5. On the Enter Controller Connection Data window, complete the fields as described in Table 3-18.

Table 3-18 Controller Connection Configuration

Field	Description
Controller Host	Fully-qualified host name of the Controller machine; example: `adminbox.east.example.com`
Console port	HTTP port for the WLOC Administration Console; default: `9001`
Console secure port	HTTPS port for the WLOC Administration Console; default: `9002`
Console mode	Select one of the following to specify how clients may connect to the administration console: `Secure`—HTTPS only `Unsecure`—HTTP only `Both`—Either HTTP or HTTPS
Internal port	Port used by agents for unsecure internal communication with the Controller; default: `9003`

Table 3-18 Controller Connection Configuration

Field	Description
Internal Secure Port	Port used by agents for secure internal communication with the Controller; default: `9004`
Security mode	Select one of the following to specify the security level to be used for internal communications between WLOC components. `Unsecure`—use HTTP without SSL and guarantee of message confidentiality and integrity. This is sufficient for development systems. `Secure`—use HTTPS providing message confidentiality and integrity. This should be used with production systems. NOTES: • 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. On the Configure Controller Logging window, complete the fields as described in Table 3-19.

Table 3-19 Configure Controller Logging

Field	Description
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: *`BEA_HOME`*`\user_projects\controller\logs\Controller.log`

7. On the first Configure Controller Notifications (1 of 6) window, select the **Enable SMTP Notifications** checkbox 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)

Field	Description
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. On the Configure Controller Notifications (2 of 6) window, select the **Enable JMX Notification** and/or **SNMP** checkbox 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)

Field	Description
Agent Host	Hostname 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. On the Configure Controller Notifications (3 of 6) window, select the **Enable JMS Notification** checkbox 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)

Field	Description
Destination JNDI Name	Fully-qualified package and class of the JMS notifier; default: `com.bea.adaptive.loc.notification.JMSNotifier`
Connection Factory JNDI Name	JNDI connection factory; default: `com.bea.adaptive.loc.notification.JMSNotifierQueueConnectionFactory`
Initial Factory Class	Fully-qualified package and class of the initial factory; default: `org.mom4j.jndi.InitialCtxFactory`
Provider URL	JNDI provider URL.
Security Principal	JNDI user name.
Password	JNDI user's password.

10. On 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

Field	Description
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`

Table 3-23 Configure Agents for this Controller

Field	Description
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. On the Use SSH for WLOC ESX Agents window, select the **Enable SSH for LVM instances** checkbox 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. On 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. On 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

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

Modifying Controller Configurations

Controller configurations can be changed using the WLOC administration console, by directly editing the `loc-controller-config.xml`, or by re-running the configuration wizard. When rerunning the configuration wizard, the only change that can be made is to 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 by running 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. On 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 in each Agent's row. When done, click **Next**.
5. On the Update Existing Controller Configuration window, click **Create**. This updates the Controller's `loc-controller-config.xml` file with the new Agent information.

CHAPTER 4

Starting and Stopping the Controller and Agents

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

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\startAgent.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:

1. Select the **Agents** tab at the top of any console page to display a list of Agents.
2. Select the Agent's checkbox and click **Shutdown**.

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. On 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 line 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. The Agent will attempt to restart automatically and reconnect to both its managed resources and the Controller.

If the Controller shuts down unexpectedly, the Agents to which it connects will continue to collect and locally store information from its managed resources. They will then send that information to the Controller after it restarts.

CHAPTER 5

Configuring Services

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

Overview

A service is a set of processes managed by WLOC. A process is typically a stack that contains the JVM, the application container, and application logic. For example, a service can be used to manage one or more WebLogic Server instances a domain.

A service's configuration includes such information as:

- The physical computing resources required to run all of its processes. These requirements are expressed as a range of CPU cycles, memory, and disk space.
- Service metadata that defines the Java classes or other executables that make up the service's processes.

- Policies that define service level agreements (SLA) and actions to take when the service is operating outside of the SLA constraints.
- A service priority that is used to resolve conflicts when more than one service fails to meet its SLA at the same time.

Services and Process Groups

A typical service manages a group of processes that are logically related and organized in *process groups*. 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	The number of processes required for the process group.
Name	Name of the initial JVM instance. For each additional instance, a numeric suffix is added to the name.
Description	A description of the JVM.
Main Class or Main JAR	The class or JAR file that instantiates this process.
Host	The fully-qualified host name where the JVM resides.
Starting Port	The 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.

Table 5-1 Process Group Configuration

Process Group Attribute	Description
Classpath	The classpath for the class or JAR file that instantiates this process. For information about obtaining classpath information, see "Classpath" on page 5-10.
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" on page 5-8.
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 making a JMX connection to the JVM. These will be the default values for all subsequently created processes.
Instance Directory	The directory where the JVM instance resides.
Native Lib Directory	The directory where the JVM native libraries reside. The value will be passed to jvm args as `-Djava.library.path`
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 (`iiop`, `iiops`, `http`, and `https`).
Ready metric	Information about the metric that WLOC obtains from the JVM instance and uses to determine whether or not a process is available. For more detailed information, see "Ready Metrics" on page 5-6.
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 5-5.

Table 5-1 Process Group Configuration

Process Group Attribute	Description
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.

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. See the WLOC Administration Console help system for step-by-step instructions.

When defining the service's 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\metada-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. This includes:

- The metadata schema file. The name and location of this file is *BEA_HOME**WLOC_HOME*\schemas\loc-metadata.xsd. This file may may also be accessed at http://edocs.bea.com/wloc/docs10/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.

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 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.

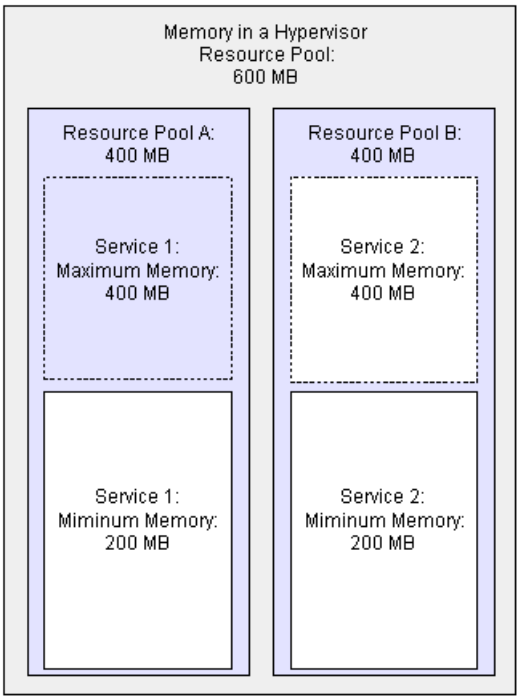
Note: For CPU measurement, WLOC uses normalized megahertz (MHz) across CPU architectures so that a megahertz of processing on an i386 processor is comparable to a megahertz on processors types.

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.

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

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: `com.bea:Name=AdminServer,Type=ServerRuntime`
Attribute	The name of the MBean attribute to test. Example: `state`. For more information about the format to use to specify MBean attributes, see WebLogic Server MBean Reference.
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 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.

The WLOC plain agent uses a management server to obtain the current CPU and memory use of a service. For plain agents, you must be sure to turn on the management server. To do so, include the following JVM arguments in the process group specification. Note that the port number must be a unique on each machine. The default port is 7091.

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 above, use the following two arguments:

```
-Djrockit.managementserver.port=<port-number>
-Xmanagement:ssl=false,authenticate=false
```

These arguments are not needed for process groups managed by a Hypervisor Agent.

Listing 5-2 shows the JVM arguments used to start the 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

```
<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*.

Classpath

Among the JVM required 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 LiquidVM file system. If the element is in the iso, the path must begin with '`/appliance`' since the iso is mounted on '`/appliance`'. If the element is in a 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 '`/`'.

- For WLS-VE 9.2v1.0, the standard classpath as set by the `commonStartVE` script is:

  ```
  /bea/patch_weblogic921/profiles/default/sys_manifest_classpath/weblogic
  _patch.jar:/appliance/java/lib/tools.jar:/appliance/bea/weblogic92/serv
  er/lib/weblogic.jar:/appliance/bea/weblogic92/server/lib/webservices.ja
  r
  ```

- WLS-VE 9.2v1.1 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/serv
  er/lib/weblogic.jar:/appliance/bea/weblogic92/server/lib/webservices.ja
  r:
  ```

- For generic Java application on LVM, an example classpath is as follows:

  ```
  /mynfs/patches/myapp_patch1.jar:/patches/myapp_patch2.jar:/appliance/ap
  plications/myapp/lib/myapp1.jar:/appliance/applications/myapp/lib/myapp
  2.jar
  ```

Service Deployment

The state of a service is undeployed until it is started. When the service is started, the following occurs:

1. The Controller evaluates the process requirements of each process group in the service.
2. The Controller 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.
3. To determine the resource pool on which to place the service, the Controller uses the placement algorithm specified when the service was defined.
4. The Controller stages each instance individually. When the minimum number of processes from each group is started, the service is deployed.
5. The Controller evaluates the runtime policy and initiates the specified actions when a constraint is violated.
6. The Controller continually evaluates the runtime policy using information if obtains from the managed application.

Deploying Services in Virtualized Environments

When WLOC deploys a service, it first searches for the set of placement organizations that match the given constraints for memory, CPU, and such factors as software and IP address. A service's placement-algorithm attribute affects placement organization and specifies whether you want VMs placed into pools so that there is as little memory resource left over as possible (`PreferSmaller`) or as much left over as possible (`PreferLarger`). The default is `PreferLarger`.

CHAPTER 6

Defining Policies

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

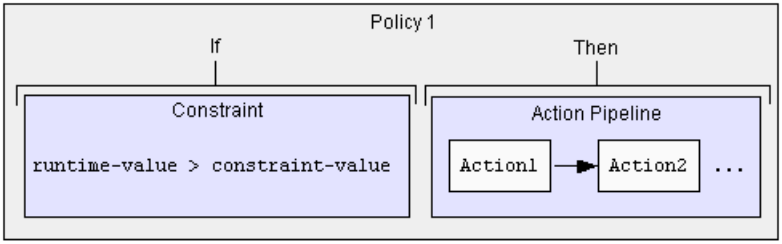
Policy Components

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 more actions to take if the service runtime-value is outside the constraint-value range.

For example, a policy could 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’.

Figure 6-2 shows a constraint/action binding as it appears in the `metadata-config.xml` file. In the example, the ‘key’ names are references to the complete constraint and actions definitions located further down in the file

Figure 6-2 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>
```

It is very important 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 itself, and the action itself.

Figure 6-3 provides an XML snippet of a constraint binding within a service definition. Note that the constraint binding contains ‘keys’ to the actual constraint and action definitions elsewhere in the file.

Figure 6-3 Constraint/Action Binding

```
 <ns2:name>MyService</ns2:name>
 <ns2:description>MyService</ns2:description>
 <ns2:state>undeployed</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>
```

Figure 6-4 provides an XML snippet of the constraint definition referenced in the constraint binding shown above.

Figure 6-4 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>
```

For assistance in understanding how to edit the metadata configuration file, use the schema file (*BEA_HOME**WLOC_HOME*\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 6-4
- "Runtime Policies" on page 6-5
- "Administrative Policies" on page 6-5

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 Resource Agreement	Service Notification
For more information, see Table 6-4, "Constraint Types," on page 6-7.	**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 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 Resource Agreement	Process Resource
For more information, see Table 6-4, "Constraint Types," on page 6-7.	For more information, see Table 6-6, "Action Types," on page 6-12.

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 Runtime	Notification Configuration
For more information, see Table 6-4, "Constraint Types," on page 6-7.	For more information, see Table 6-6, "Action Types," on page 6-12.

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 the *Define Constraints* help topic in the WLOC Administration Console help system.

Table 6-4 Constraint Types

Type	Description
Deployment	These define the desired deployment state (Deployed, Staged, Undeployed) of a service. In the Administration Console, these are selectable as: Based on deploying a service at a date/time Based on undeploying a service at a date/time Based on a service deployment state

Table 6-4 Constraint Types

Type	Description
Runtime	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. 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. In the Administration Console, these are selectable as: 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
Resource Agreement	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 `metadata-config.xml`, the constraint must be bound to an action whose value is **INTERNAL**. In the Administration Console, these are selectable as: Based on a maximum amount of CPU Based on a share of CPU Based on a minimum amount of memory Based on a maximum amount of memory Based on a share of memory Based on minimum number of processes Based on maximum number of processes Based on software availability Based on IP Address Based on ISO software availability Based 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. 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 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 MinAverageUsedPhysicalMemory	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.

Figure 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 the WLOC Administration Console help system.

Figure 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.

Figure 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 administration console, the console prompts for the required parameter values.

Note: For detailed information about these action types, see the *Define Actions* help topic in the WLOC Administration Console help system.

Table 6-6 Action Types

Action Type	Description
Service	Actions that start, stop, stage, test, or destroy a service. StageServiceAction StartServiceAction StopServiceAction DestroyServiceAction TestStartServiceAction
Process	Actions that start, stop, destroy, stage, suspend, change the configuration, or gracefully stop a instance of a Java process. StartJavaInstanceAction StopJavaInstanceAction DestroyJavaInstanceAction StageJavaInstanceAction ResumeJavaInstanceAction RemoteAction SuspendJavaInstanceAction JavaInstanceConfigAction
Notification	These actions send different types of notifications. Console EmailNotificationAction JMXNotificationAction JMSNotificationAction SNMPNotificationAction

Table 6-6 Action Types

Resource	Actions that change the maximum, minimum or share of CPU or memory that is available. ChangeMaxCPUAction ChangeMinCPUAction ChangeShareCPUAction ChangeMaxMemoryAction ChangeMinMemoryAction ChangeShareMemoryAction
Configuration	Actions that performs JMX operations on specified managed objects. You can define actions that: • 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.

CHAPTER 7

Configuring Security

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

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 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. To secure the console, this should be changed.

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, BEA 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

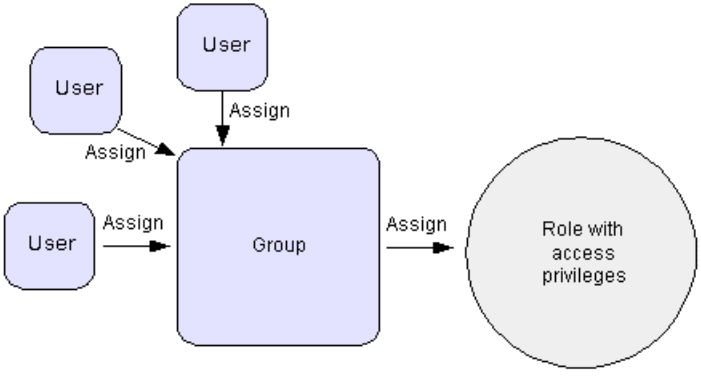


Table 7-1 describes WLOC security groups and the out-of-box security role assignment for the group. You may create additional groups, but may not create additional security roles. You can indirectly create a scoped service admin role by creating a service; when you create a new service, a security role is created to administer that service.

Table 7-1 WLOC Security Groups

Group	Role Assignment
Administrators	Admin
ServiceAdministrators	ServiceAdmin
Monitors	Monitor

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.

Table 7-2 WLOC Security Roles

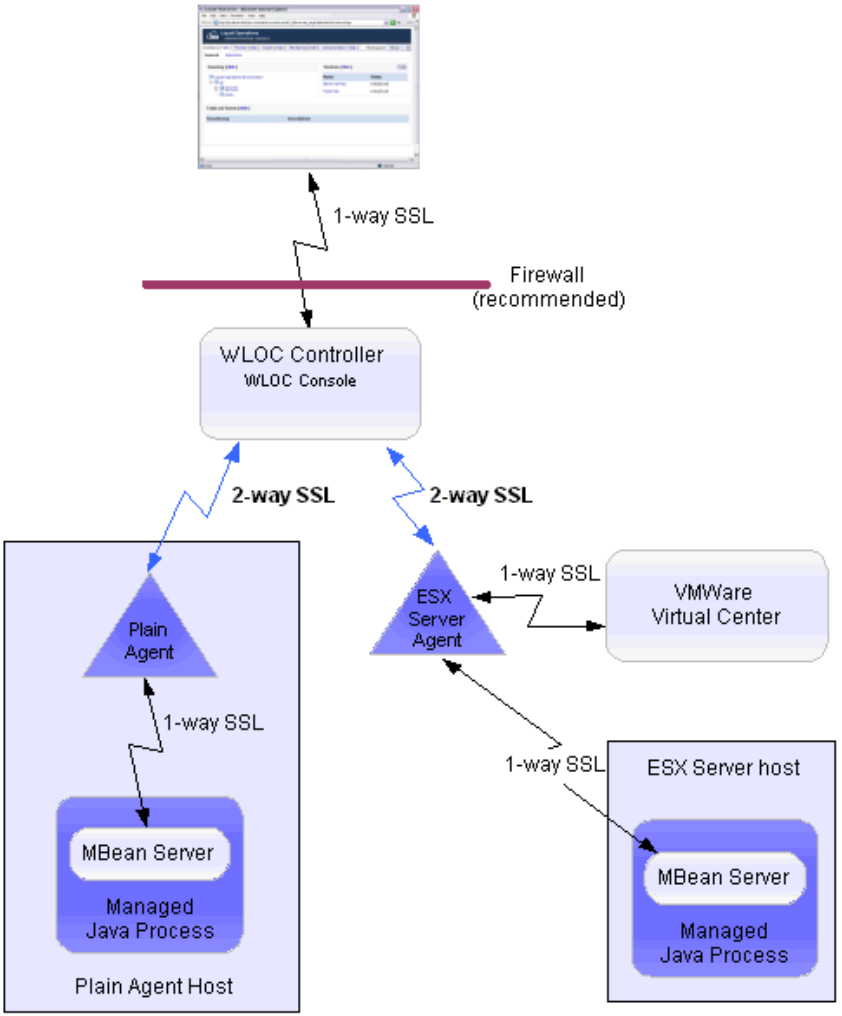
	Admin	Service Admin	Monitor	Service Admin*Service-Name*
Configure the Controller and Agents	X			
View monitoring data in the console	X	X	X	Only for *Service-Name*
Create and modify users and groups	X			
Create and modify resource pools	X			
Create and modify services	X	X		Only for *Service-Name*
Deploy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

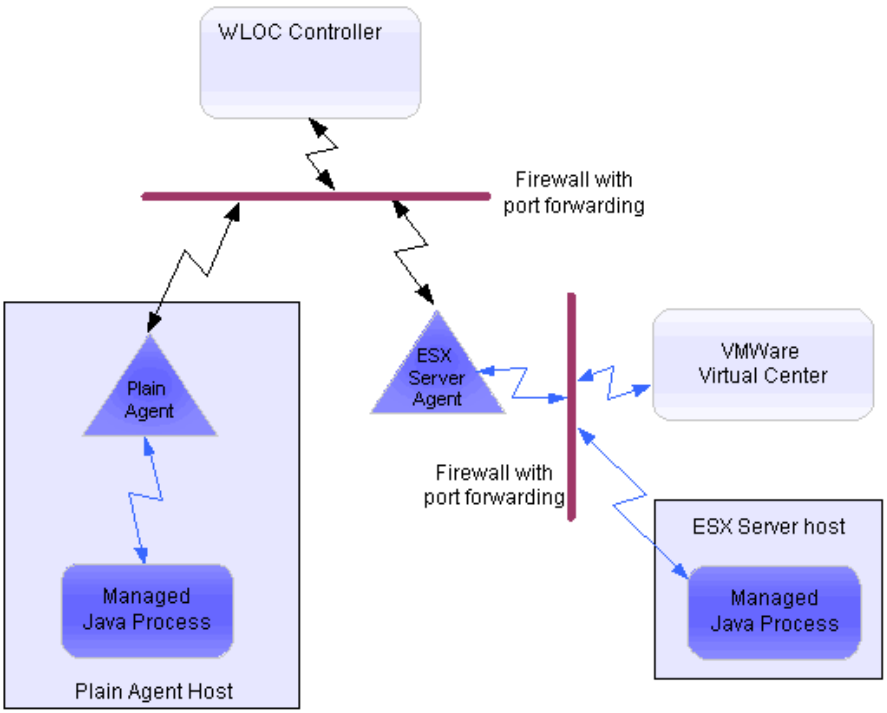
BEA recommends that you place the Controller and all Agents behind the same firewall whenever possible (see Figure 7-2). Controllers and Agents communicate only through HTTPS if secure communications is enabled. In this communication, the Controller always initiates

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 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 a 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 Agent 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.

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 *BEA_HOME*\jrockit_150_12\bin. To perform the import, follow these steps.

1. To import the Controller's identity certificate, open a command line 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 *BEA_HOME*\jrockit_150_12\bin. To perform the import, follow these steps.

1. Open a command line 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 copy the certificate from the Virtual Center and then using Keytool to perform the import. The Keytool utility is located in *BEA_HOME*\jrockit_150_12\bin.

To do this, follow these steps.

1. Use the GrabCert utility to copy the certificate from the Virtual Center to BEA_HOME/WLOC_HOME/user_projects/ESXAgent/ssl/internal.

2. Open a command line and enter the following command:

```
keytool -import -noprompt -alias vcinternal -trustcacerts -keystore
BEA_HOME/WLOC_HOME/user_projects/ESXAgent/ssl/internal/internaltrust.jk
s -storepass jks_password -file
BEA_HOME/WLOC_HOME/user_projects/ESXAgent/ssl/internal/vcidentity.pem
```

where

vcinternal is an alias to use for the Virtual Center identity.
jks_password is the Agent's internal trust keystore password (default is *changeit*).

3. 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.

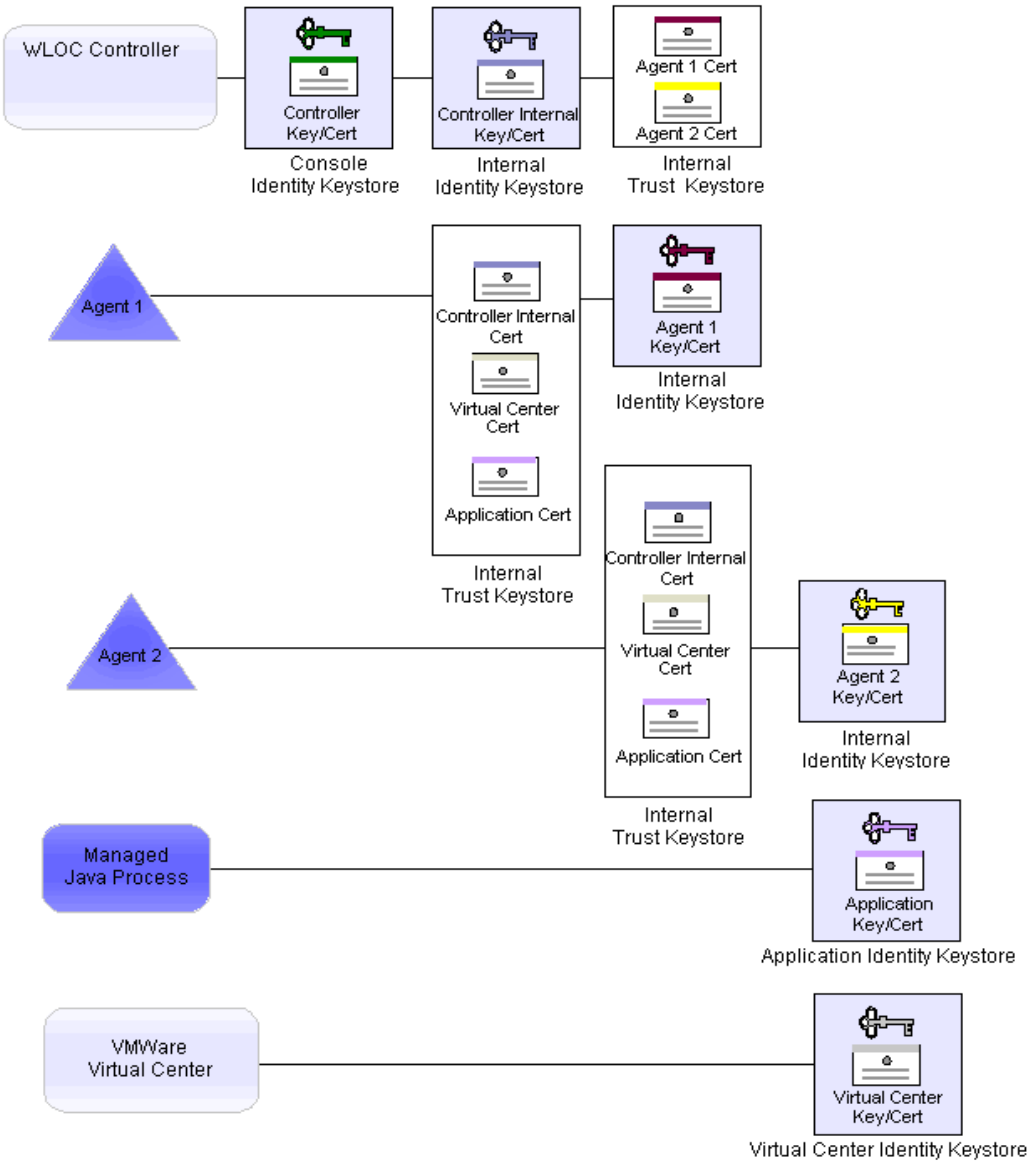
In addition, MBean Servers are usually configured to require authentication using a user name and password as credentials. (WebLogic Server MBean Servers always require authentication and authorization.)

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_HOME`*`/ssl` where *`CONTROLLER_HOME`* is the directory in which you install the Controller.
- 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_HOME`*`\ssl\internal` where *`CONTROLLER_HOME`* is the directory in which you install the Controller.

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.

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.

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_HOME`*`\ssl\internal` where *`AGENT_HOME`* is the directory in which you install an Agent.

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.

CHAPTER 8

Logging, Auditing, and Monitoring

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

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 save 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 8-5.
Log File Name	The name of the file that stores current log messages. To include a time and date stamp in the file name when the log file is rotated, add `java.text.SimpleDateFormat` variables to the file name. Surround each variable with percentage (%) characters. For example, if the file name is defined to be `myserver_%yyyy%_%MM%_%dd%_%hh%_%mm%.log`, the log file will be named `myserver_yyyy_mm_dd_hh_mm.log`. 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 `myserver_2008_04_02_10_05.log`. If you do not include a time and date stamp, the rotated log files are numbered in order of creation. For example, `myserver.log00007`.
Rotation Type	Should the log file be rotated by size or by time? For more information about log file rotation, see "Rotating Log Files" on page 8-6.
Rotation Size	If the log file is rotated by size, this is the size that triggers the rotation. When the file size is reached, the file is renamed by incrementing the file extension (e.g., controller.log0001, controller.log0002, etc.) and saved in the rotation directory. The server then writes messages to a new file named *filename*.log. **Note:** WLOC sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.
Rotation Directory	The directory where log files are saved upon rotation.
Number of Files Limited	Indicates 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. If you do not enable this option, the server creates new files indefinitely and you must clean up these files as necessary.

Table 8-1 Logging Configuration

Field	Description
Rotation File Count	If you limit the number of stored files, this attribute specifies the maximum number of log files to be stored. This number does not include the file that the server uses to store current messages.
Rotation on Startup	Select this checkbox if the log file should be rotated when the Controller or Agent is started.
Rotation Time	Time of day that log files are rotated. Specify using the format: `hh:mm`. This value is used only if the Rotation Type is set to *By Time*. If the time you specify has already elapsed, WLOC will perform the day's rotation immediately.
Rotation Time Span	The hour/minute interval in which log files are rotated. Specify a number in format: `hh:mm`. The default is *00:00* and equates to midnight. This value is used only if the Rotation Type is set to *By Time*. The actual rotation time is determined by the following formula: `(Rotation Time Span) * (Rotation Time Span Factor)`
Rotation Time Span Factor	The interval in milliseconds in which log files are rotated. This default value is *3600000*. This value is used only if the Rotation Type is set to *By Time*. The actual rotation time is determined by the following formula: `(Rotation Time Span) * (Rotation Time Span Factor)`
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 8-5.

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 examined by accessing the log files. Currently, they cannot be viewed in the Administration Console.

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

Note: BEA Systems recommends that you 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:

```
sxxx####<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 8-5.
Subsystem	Indicates the subsystem of WLOC that was the source of the message.
Thread ID	Identifies the origins of the message. The `Thread ID` 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 `BEA-` 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	Logs Trace level events.
DEBUG	Logs Debug events.
INFO	Used for reporting normal operations; a low-level informational message.
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 BEA 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 can configure the following aspects of the Audit Service:

Table 8-4 Audit Service Configuration

Field	Description
Log File Name	The name of the audit log file. To include a time and date stamp in the file name when the log file is rotated, add `java.text.SimpleDateFormat` variables to the file name. Surround each variable with percentage (%) characters. For example, if the file name is defined to be `myserver_%yyyy%_%MM%_%dd%_%hh%_%mm%.log`, the log file will be named `myserver_yyyy_mm_dd_hh_mm.log`. 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 `myserver_2008_04_02_10_05.log`. If you do not include a time and date stamp, the rotated log files are numbered in order of creation. For example, `myserver.log00007`.
Rotation Type	Should the audit log file be rotated by size or by time? For more information about log file rotation, see "Rotating Log Files" on page 8-6.
Rotation Size	If the audit log file is rotated by size, this is the size that triggers the rotation. When the file size is reached, the file is renamed by incrementing the file extension (e.g., controller.log0001, controller.log0002, etc.) and saved in the rotation directory. The server then writes messages to a new audit log file. **Note:** WLOC sets a threshold size limit of 500 MB before it forces a hard rotation to prevent excessive log file growth.
Rotation Directory	The directory where log files are saved upon rotation.
Number of Files Limited	Select this option to limit the number of stored log files to the value specified in the **Rotation File Count** field. (Requires that you specify a file rotation type of SIZE or TIME.) When the limit is reached, WLOC, the oldest log file is deleted and a new one is created. If you do not enable this option, the server creates new files indefinitely and you must clean up these files as necessary.
Rotation File Count	If you limit the number of stored files, this attribute specifies the maximum number of log files to be stored. This number does not include the file that the server uses to store current messages.

Table 8-4 Audit Service Configuration

Field	Description
Rotation on Startup	Select this checkbox if the log file should be rotated when the Controller or Agent is started.
Rotation Time	Time of day that log files are rotated. Specify using the format: `hh:mm`. This value is used only if the Rotation Type is set to *By Time*. If the time you specify has already elapsed, WLOC will perform the day's rotation immediately.
Rotation Time Span	The hour/minute interval in which log files are rotated. Specify a number in format: `hh:mm`. The default is *00:00* and equates to midnight. This value is used only if the Rotation Type is set to *By Time*. The actual rotation time is determined by the following formula: `(Rotation Time Span) * (Rotation Time Span Factor)`
Rotation Time Span Factor	The interval in milliseconds in which log files are rotated. This default value is *3600000*. This value is used only if the Rotation Type is set to *By Time*. The actual rotation time is determined by the following formula: `(Rotation Time Span) * (Rotation Time Span Factor)`
Enabled	When selected, the audit log service is enabled.
Audit Types	The category of events to audit. For an Agent, select ALL or AGENT_ACTION. For the Controller, select ALL or one or more of the following: CONTROLLER_CONFIGURATION SERVICE_CONFIGURATION RULES CONTROLLER_ACTION ADJUDICATION AGENT_CONFIGURATION For more information, see "Audit Event Types" on page 8-9.

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.
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 8-9.
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" on page 5-8.

Note: The service must be using a JRocket JVM.

For more information, see Monitor Resources in the WLOC Administration Console help system.

Viewing Events

By default, events are displayed at the bottom of the Administration Console page. In addition, you can view events in the **Events** tab.

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 6-5.

APPENDIX A

Silent Mode Configurations

This appendix provides information about silent-mode configurations.

Running the Configuration Wizard in Silent-Mode

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

To run the 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-11

2. Launch the configuration wizard by entering one of the following commands:

On Windows:

```
BEA_Home\WLOC_HOME\common\bin\config.cmd -mode=silent
-silent_xml=<path>\silent_config_input.xml -log=silent_config.log
```

On UNIX or Linux:

```
BEA_Home/WLOC_HOME/common/bin/config.sh -mode=silent
-silent_xml=<path>/silent_config_input.xml -log=silent_config.log
-log_priority=debug
```

where *<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="agent.east.acme.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="Unsecure" />
  <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.name	Specify the name of the Agent. `<data-value name="Agent.name" value="agent1" />`
Agent.host	Specify the fully-qualified host name where the Agent resides. `<data-value name="Agent.host" value="agent.east.acme.com"/>`
Agent.port	Specify the Agent's HTTP port number used when communicating with the Controller in unsecure mode. `<data-value name="Agent.port" value="8001" />` In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.
Agent.securePort	Specify the Agent's HTTPS port number used when communicating with the Controller in secure mode. `<data-value name="Agent.SecurePort" value="8002" />` In the unlikely event you are configuring more than one Agent on the same host, be sure that each Agent uses different port numbers.
Agent.encryption.password	Specify 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. `<data-value name="Agent.encryption.password" value="changeit" />` If Security Mode is `Unsecure`, this setting will still encrypt the most sensitive data.

Table A-1 Plain Agent Configuration Values

Name	Description
Agent.useSecureConnections	Specify the security mode to use for connections with the Controller. `Unsecure` — sufficient for development. `Secure` — should be used for production environments. `<data-value name="Agent.useSecureConnections" value="Unsecure" />` Specifying Secure mode 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**. **NOTE:** The Controller and all Agents must be set to the same security mode.
Logging.fileSeverity	Specify the severity level of events to log. `<data-value name="Logging.fileSeverity" value="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.
Logging.stdoutSeverity	Specify the severity level of events that should be written to stdout. `<data-value name="Logging.stdoutSeverity" value="Info" />` Uses same event levels as the log file.
Logging.baseFileName	Specify the log file directory and name. `<data-value name="Logging.baseFileName" value="./logs/Agent.log" />`
Logging.fileRotationDir	Specify the log rotation directory. `<data-value name="Logging.fileRotationDir" value="./logs/logrotdir" />`
Agent.internalidentity.keystorePassword	Specify the passphrase for the Agent's internal identity keystore. This value will be encrypted. `<data-value name="Agent.internalidentity.keystorePassword" value="changeit"/>`

Table A-1 Plain Agent Configuration Values

Name	Description
Agent.internaltrust.keystorePassword	Specify the passphrase for the Agent's internal trust keystore. This value will be encrypted. `<data-value name="Agent.internaltrust.keystorePassword" value="changeit"/>`
Agent.type	Specify plainAgent as the agent type. `<data-value name="Agent.type" value="plainAgent" />`
PlainAgent.name	Name of the resource pool managed by the Agent. `<data-value name="PlainAgent.name" value="pool 1" />`
PlainAgent.description	An arbitrary description of the resource pool. `<data-value name="PlainAgent.description" value="pool 1" />`
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="agent.east.acme.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="Unsecure" />
  <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" />
```

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="acme.rdd.com" />
<data-value name="EsxAgent.username" value="admin66" />
<data-value name="EsxAgent.password" value="dorwssap" />
<data-value name="EsxAgent.vmwarePool.dataCenter" value="datacenter1" />
<data-value name="EsxAgent.vmwarePool.computeResource" value="esxHost.rdd.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-element name="VM Network">
<data-value name="ipAddresses" value="10.344.22.86,10.170.43.81"/>
<data-value name="description" value="WLOC VM Network"/>
<data-value name="gateway" value="172.18.128.1"/>
<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="acme.com"/>
</data-element>
</data-group>
<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="172.18.128.67:/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	Specify `esxagent` as the Agent type. `<data-value name="Agent.type" value="esxagent" />`
EsxAgent.name	Specify the ESX Agent name. `<data-value name="EsxAgent.name" value="vmware-agent" />`
EsxAgent.description	Specify an arbitrary description of the ESX Agent. `<data-value name="EsxAgent.description" value="VMWare ESX Information" />`
EsxAgent.vcHost	Specify the Virtual Center host name. `<data-value name="EsxAgent.vcHost" value="acme.rdd.com" />`
EsxAgent.username	Specify the Virtual Center administrator username. `<data-value name="EsxAgent.username" value="admin66" />`
EsxAgent.password	Specify the Virtual Center administrator's password. `<data-value name="EsxAgent.password" value="dorwssap" />`
EsxAgent.vmwarePool.dataCenter	Specify the name of the Datacenter containing the resource pool managed by this Agent. `<data-value name="EsxAgent.vmwarePool.dataCenter" value="datacenter1" />`
EsxAgent.vmwarePool.computeResource	Specify the ESX Server host or cluster name. `<data-value name="EsxAgent.vmwarePool.computeResource" value="esxHost.rdd.com" />`

Table A-2 ESX Agent Configuration Values

Name	Description
EsxAgent.vmwarePool.resourcePool	Specify the Resource Pool containing the LiquidVM instances to be managed by this Agent. `<data-value name="EsxAgent.vmwarePool.resourcePool" value="WLOCPool1" />`
EsxAgent.vmwarePool.description	Specify an arbitrary description of the resource pool. `<data-value name="EsxAgent.vmwarePool.description" value="WLOCPool1" />`
VMNetwork Name	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. `<data-element name="VM Network">`
ipAddresses	One or more IP addresses reserved for the LiquidVM instances. Specify multiple addresses on the same line separated using a comma (,). `<data-value name="ipAddresses" value="10.344.22.86,10.170.43.81"/>`
description	An arbitrary description. `<data-value name="description" value="WLOC VM Network"/>`
gateway	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. `<data-value name="gateway" value="172.18.128.1"/>`
netMask	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. `<data-value name="netMask" value="255.255.248.0"/>`
dnsServers	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. `<data-value name="dnsServers" value="10.40.0.86,10.40.0.87"/>`

Table A-2 ESX Agent Configuration Values

Name	Description
domainName	The domain name used by the LiquidVM instance. `<data-value name="domainName" value="acme.com"/>`

Table A-2 ESX Agent Configuration Values

Name	Description

iso-software	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 `<data-group name="available-software">` element.
	
	Where:
	
	name— ISO name
	description— Arbitrary description
	path — Location of the ISO software, including the datastore. Syntax: `[datastore] /path/filename`
	version — VMWare version (`1.0`, `1.1`, or `1.2`)
	
	Example:
	
	`<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>`

Table A-2 ESX Agent Configuration Values

Name	Description

nfs-software	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.
	
	Where:
	
	name— NFS name
	description— Arbitrary description
	path — The NFS share path using the following syntax:
	`<ip_address>:<path>,uid=<uid_num>,gid=<gid_num>`
	`<ip_address>` — IP Address of the NFS share host
	`<path>` — path to the share
	`<uid_num>` — userid number
	`<gid_num>` — group id number
	mode — this value must be EXCLUSIVE
	
	Example:
	
	`<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="172.18.128.67:/LOC/bea/bea.home,uid=55004,gid=10000"/>`
	`<data-value name="mode" value="EXCLUSIVE"/>`
	`</data-element>`

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-13.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Silent config wizard option: -mode=silent -silent_xml=/home/me/silent.xml
-->
<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="Secure" />
   <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/logrotdir"/>
   <data-value name="Notification.smtp.enabled" value="true"/>
   <data-value name="Notification.smtp.toAddress" value="WLOC@acme.com"/>
   <data-value name="Notification.smtp.fromAddress"
value="WLOCadmin@acme.com"/>
   <data-value name="Notification.smtp.smtpServer" value="smtpserver.acme.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="acme.acme2.com"/>
   <data-value name="Notification.snmp.agent.port" value="2002"/>
   <data-value name="Notification.snmp.trapDestinations.destination.host"
value="acme.acme3.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="agent.east.acme.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" />`

Table A-3 Controller Configuration Values

Name	Description
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" />`
Controller.useSecureConnections	Select one of the following to specify the security level to be used for internal communications between WLOC components. `Unsecure`—HTTP without SSL and guarantee of confidentiality and integrity. `Secure`— HTTPS providing message confidentiality and integrity. **NOTE:** All Agents must use the same Security mode established on the Controller with which they communicate. `<data-value name="Controller.useSecureConnections" value="Secure" />`
Logging.fileSeverity	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 `<data-value name="Logging.fileSeverity" value="Info" />`
Logging.stdoutSeverity	Level of events to write to stdout. Uses same event levels as the log file. `<data-value name="Logging.stdoutSeverity" value="Info" />`
Logging.baseFileName	Log file and directory relative to the Controller directory. `<data-value name="Logging.baseFileName" value="./logs/controller.log" />`
Logging.fileRotationDir	Rotation directory for controller logs. `<data-value name="Logging.fileRotationDir" value="./logs/logrotdir" />`
Notification.smtp.enabled	Specify *true* to enable SMTP notification; otherwise, specify *false*. `<data-value name="Notification.smtp.enabled" value="true"/>`
Notification.smtp.fromAddress	E-mail address from which notifications should be sent. `<data-value name="Notification.smtp.fromAddress" value="WLOCadmin@acme.com"/>`

Table A-3 Controller Configuration Values

Name	Description
Notification.smtp.toAddress	E-mail address to which notifications should be sent. `<data-value name="Notification.smtp.toAddress" value="WLOC@acme.com"/>`
Notification.smtp.smtpServer	SMTP mail server through which to send notifications. `<data-value name="Notification.smtp.smtpServer" value="smtpserver.acme.com" />`
Notification.jms.enabled	Specify *true* to enable JMS notification; otherwise, specify *false*. `<data-value name="Notification.jms.enabled" value="false"/>`
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://172.18.134.173:9901"/>`
Notification.jms.jndiProperties.securityPrincipal	JNDI user name. `<data-value name="Notification.jms.jndiProperties.securityPrincipal" value="system"/>`

Table A-3 Controller Configuration Values

Name	Description
Notification.jms.jndiProperties.password	JNDI user's password. `<data-value name="Notification.jms.jndiProperties.password" value="system"/>`
Notification.jmx.enabled	Specify *true* to enable JMX notification; otherwise, specify *false*. `<data-value name="Notification.jmx.enabled" value="true"/>`
Notification.snmp.enabled	Specify *true* to enable SNMP notification; otherwise, specify *false*. `<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="acme.acme2.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="acme.acme3.com"/>`
Notification.snmp.trapDestinations.destination.port	Listening port of the SNMP manager. `<data-value name="Notification.snmp.trapDestinations.destination.port" value="162"/>`
Notification.snmp.agent.trapVersion	SNMP version (`SNMPv1` or `SNMPv2`) `<data-value name="Notification.snmp.agent.trapVersion" value="SNMPv2"/>`

Table A-3 Controller Configuration Values

Name	Description
Agent Information ‘<data-element name="agent">	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. **Where**: *name* — Agent name. *host* — Fully-qualified host name or IP address of the Agent machine. *port* — HTTP port on which to access the Agent. *secure_port* — HTTPS port on which to access the Agent. *state* — One of Enabled, Connected, or Disconnected. *password* — Agent’s current passphrase. Communication between the Controller and Agent will fail unless the entry matches the Agent’s current passphrase. **Example**: <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>
LoginInfo.username	The username for logging into the Administration Console. <data-value name="LoginInfo.username" value="WLOCBootUser" />
LoginInfo.password	The password for the above user. <data-value name="LoginInfo.password" value="changeit" />
Controller.demoidentity.keystorePassword	The Controller identity keystore password used for connections to the console. <data-value name="Controller.demoidentity.keystorePassword" value="changeit" />

Table A-3 Controller Configuration Values

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" />`