BEA WebLogic Operations Control™

Introducing WebLogic Operations Control

Version 1.0
Document Revised: April 18, 2008

Contents

Introducing WebLogic Operations Control

Glossary

Introducing WebLogic Operations Control

This document provides an introduction to BEA WebLogic® Operations Control.

What is BEA WebLogic Operations Control?

While the introduction of virtualization technology into the operations center offers the promise of maximizing the use of your physical hardware, the technology also introduces complexities that can make it difficult to determine how closely your applications honor their service level agreements (SLAs) while making the most efficient use of computing resources.

BEA WebLogic Operations Control (WLOC) is a management framework for virtualized and non-virtualized enterprise Java applications that addresses the key challenges involved in application virtualization. To address these challenges, WLOC:

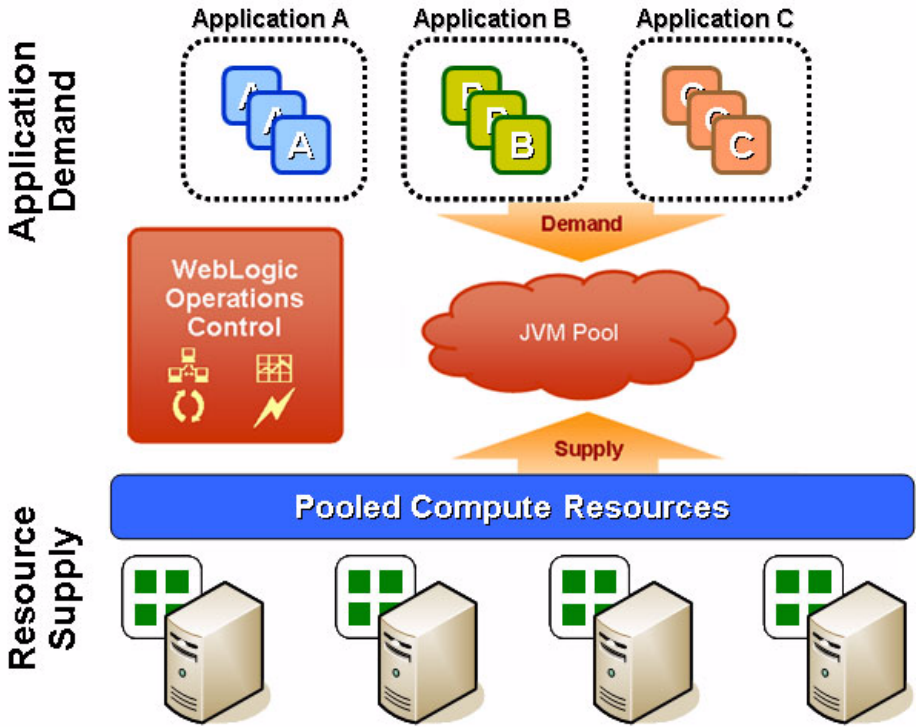
- Introduces a layer of abstraction over complex operations environments that enables your operations staff to think in terms of supply and demand.

- Offers a policy-based framework for creating and automatically enforcing service level agreements for Java applications.
- Monitors the use of resources across the operations center and distributes the deployment of Java applications in a manner that ensures the most efficient use of resources overall.

Managing Supply and Demand

WLOC introduces a layer of abstraction that enables your operations staff to think in terms of supply and demand instead of requiring them to understand the complexities of your computing environment and the unique requirement of each Java application that you support in your operations center.

Figure 1 Managing Supply and Demand With WLOC



On the demand side, you use WLOC to organize Java applications (processes) into WLOC services. You organize a group of related processes into a single service and manage the group as a unit; you can create one service for each process.

On the supply side, you use WLOC to organize the compute resources in your operations center into collections of resources, or resource pools. A WLOC resource pool can represent a single physical machine or a collection of virtualized resources that are made available through Hypervisor software.

WLOC effectively manages the *supply* of available resources in order to meet the *demands* of the deployed services.

Meeting Quality of Service Goals

WLOC provides an environment for encapsulating service level agreements (SLAs) as a collection of requirements and policies. The operations team can define policies based on application-level SLAs that govern the allocation of hardware and software resources, ensuring that Quality of Service (QoS) goals are met across virtualized and non-virtualized platforms. When predefined conditions occur, WLOC triggers an action. For example, WLOC might dynamically allocate resources to a service.

Choosing Resources Intelligently

WLOC monitors the use of resources across the operations center and distributes the deployment of Java applications in a manner that ensures the most efficient use of resources overall.

When you deploy a service or when a WLOC action requests that an additional process be started, WLOC examines all resource pools to determine where to host the service or process. To choose a resource pool, WLOC first eliminates any resource pool that cannot satisfy such dependencies as IP addresses or access to software. For example, if a service requires access to WebLogic Server software, WLOC eliminates any resource pools that cannot provide access to WebLogic Server software.

After considering declared dependencies, WLOC considers the capacity of each remaining resource pool, the SLAs of any services that are currently deployed, and the relative priorities declared for each service. It then uses one of the following algorithms that you select:

- Most resources available.This algorithm chooses the resource pool that currently has the most excess capacity.

  For example, if resource pool A has 600 MHz of CPU and 600 MB of RAM that are currently unused and resource pool B has 400 MHz of CPU and 400 MB of RAM that are unused, WLOC chooses resource pool A.

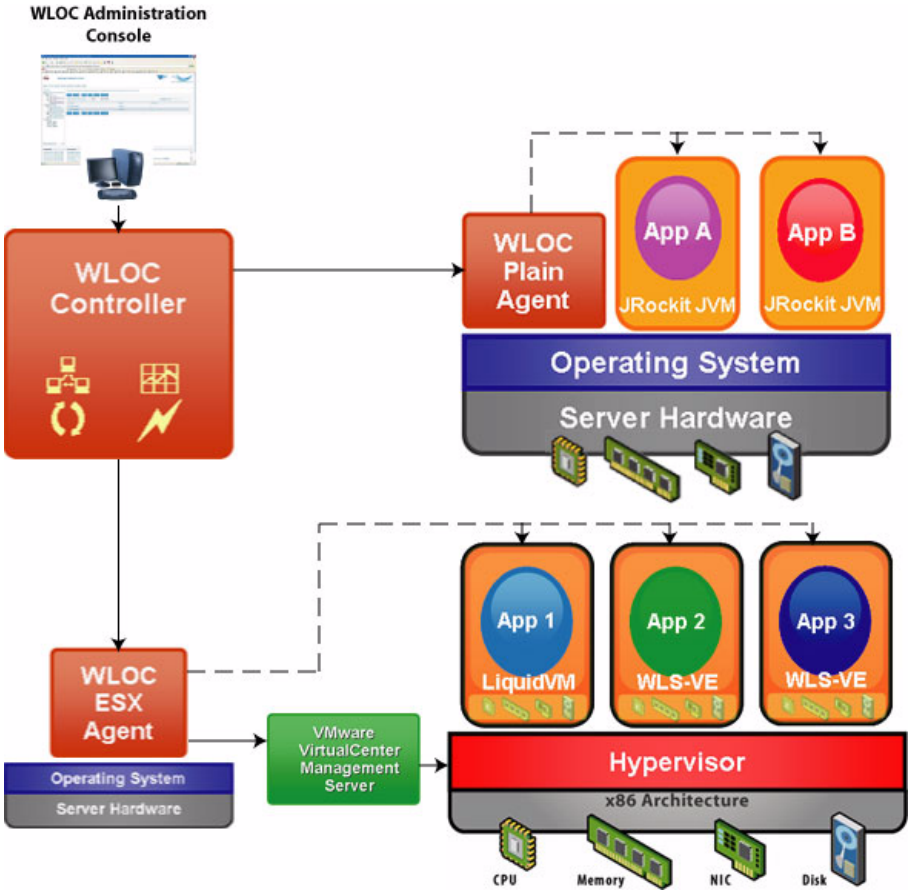
- Most efficient use of resources. This algorithm chooses the resource pool that has just enough unused resources to satisfy the minimum resource requirements of the deployment request. This algorithm ensures that WLOC is best positioned to handle services whose resource requirements are larger than the current request.

For example, if resource pool A has 600 MHz of CPU and 600 MB of RAM that are currently unused and resource pool B has 400 MHz of CPU and 400 MB of RAM that are unused, **and** if you deploy a service with a minimal requirement of 200 MHz of CPU and 200 MB of RAM, WLOC chooses resource pool B.

Overview of the WLOC Deployment Architecture

A typical WLOC deployment contains a single WLOC **Controller**—the centralized component that gathers data about the operating environment—and multiple **Agents** that manage and monitor resources and communicate that information back to the Controller. The Controller hosts the WLOC Administration Console that enables you to visually configure, manage, and monitor the WLOC environment.

Figure 2 Components of WLOC



Note: All of the components use HTTPS or HTTP to communicate.

Each WLOC component shown in the previous figure is described in the following table.

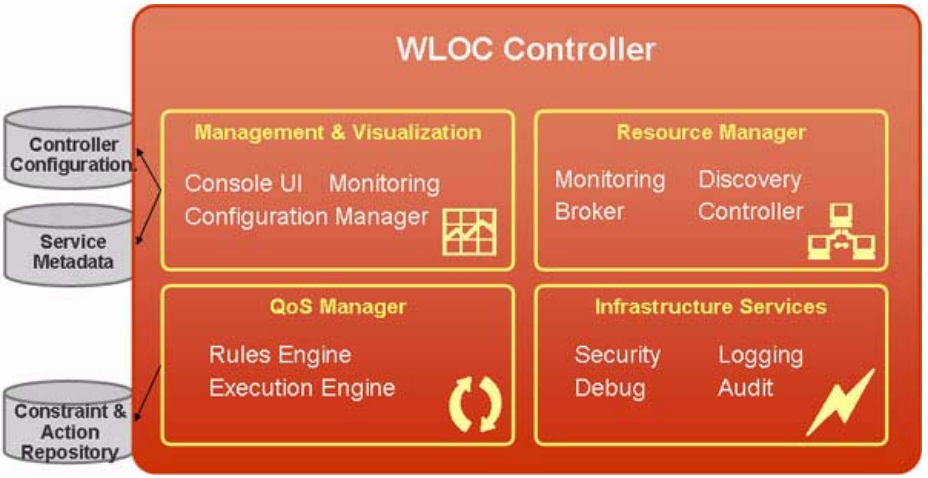
Table 1 Components of WLOC

Component	Description
Controller	Centralized component that gathers data from Agents about the operating environment and deployed services to deliver adaptive management. The Controller uses the data gathered to enforce policies and to deploy new services in a way that best honors the Service Level Agreements (SLAs) of all deployed services. The Controller hosts the WLOC Administration Console. For more information, see "Controller" on page 6.
Agents	Provide information about the environment to the Controller, start and stop processes, and invoke other actions at the request of the Controller. Each Agent operates within a narrow scope: a plain Agent gathers data and manages processes on a single physical machine and an ESX Agent gathers data and manages processes on a VMware resource pool.The Hypervisor software makes available to WLOC a collection of virtualized resources, which can be supported by one or more physical hosts. For more information, see "Agents" on page 8.
WLOC Administration Console	Web browser-based, graphical user interface that you use to configure, manage, and monitor services in your operations center. It is hosted by the Controller. For more information, see "Defining Services to Organize Processes" on page 12.
Managed Java processes	A plain Agent can manage any type of Java process; an ESX Agent can manage only virtualized Java applications that run on LiquidVM (LVM), such as WebLogic Server Virtual Edition (WLS-VE).
VMware Virtual Center	An ESX Agent communicates with VMware Virtual Center to gather data about the VMware resource pool that is available for use by WLOC and to manage instances of LVM, such as WLS-VE. After an LVM instance starts, the Agent communicates with that LVM instance to gather monitoring data and invoke management actions. In this way, the VMware Virtual Center manages the supply of resources, whereas WLOC manages the demand on the resources. The VMware infrastructure must be configured using VMware tools. You cannot use WLOC to configure the virtualized environment, only to leverage the resources that are available.

Controller

The WLOC Controller is the centralized component that gathers data from Agents about the operating environment and deployed services to deliver adaptive management. The Controller uses the data gathered to intelligently deploy new services and to evaluate and enforce policies to honor the Service Level Agreements (SLAs) for all services in the environment.

Figure 3 WLOC Controller



Each WLOC environment includes a single Controller that is responsible for the following:

- Hosting the WLOC Administration Console that enables you to visually configure, manage, and monitor the WLOC environment.
- Managing resource pools.

  During the configuration of the Controller, you bind the Controller to the available Agents. To manage resource pools, the Controller communicates with the Agents to which it is bound to determine the computing resources that each Agent is capable of allocating and selects appropriate resource pools for deploying services.
- Managing the creation of services and service level agreements (SLAs).

  A WLOC *service* is a collection of one or more processes that WLOC manages as a unit. You define the service details and the associated metadata required to run each of the processes. To create SLAs, you define service policies that consist of deployment and runtime constraints and the actions to take if those constraints are not met.
- Managing service deployment and quality of service.

  To adapt the WLOC environment to best meet the SLAs of all deployed services, the Controller communicates with Agents to gather metrics. It compares policy constraints against the metrics and invokes actions when services operate outside the constraints.
- Providing logging and audit trails.

Agents

A WLOC Agent is a standalone Java process that renders the CPU cycles and memory of a machine or a collection of virtual resources as a resource pool for use by WLOC services.

Figure 4 WLOC Agent

Typically, a WLOC environment has multiple Agents that are responsible for the following:

- Managing and storing its configuration information.
- Providing ongoing visibility into the amount of resources that WLOC is using for a given resource pool.
- Controlling the life cycle of JVMs in a resource pool upon request from the WLOC Controller.
- Gathering metrics and monitoring data of its instantiated JVMs and making this data available to the Controller.
- Providing logging and audit trails.

As shown in Figure 2, WLOC provides two types of Agents:

- **Plain Agent** for managing any type of Java process. A Plain Agent renders the resources from the machine on which it resides as a WLOC resource pool.

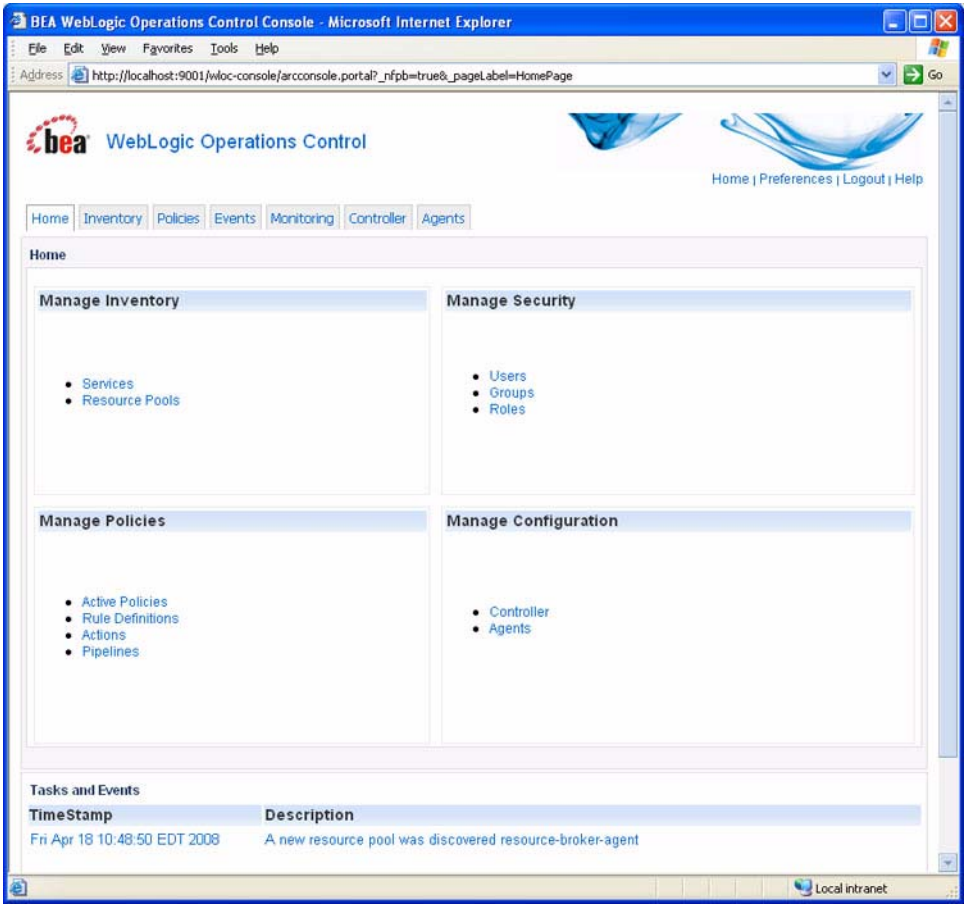
- **ESX Agent** for managing virtualized Java applications that run on LiquidVM (LVM), such as WebLogic Server Virtual Edition (WLS-VE). An ESX Agent renders a VMware resource pool as a WLOC resource pool; it can reside on any machine in your operations center.

  The VMware infrastructure must be configured using VMware tools. You cannot use WLOC to configure the virtualized environment, only to leverage the resources that are available.

WLOC Administration Console

The WLOC Administration Console is the Web browser-based, graphical user interface that you use to configure, manage, and monitor services in your operations center. It is hosted by the WLOC Controller, which communicates with Agents to gather monitoring data and to invoke management actions.

Figure 5 WLOC Administration Console



The following table summarizes the tasks that you can perform using the WLOC Administration Console.

Table 5-1 WLOC Administration Console Tasks

Task	Description
Configuration	• Configure network communications for Controllers and Agents. • Organize your Java applications into services. • Create policies to enforce service level agreements (SLAs) for your services automatically. • Configure the logging and auditing features. • Create users and assign them to groups and roles.
Management	• Deploy and activate services. • Invoke actions to manually affect services. • Adjudicate actions that are to be initiated as a result of SLA violations.
Monitoring	• Monitor the performance of services. • Monitor the use of computing resources on machines that host resource pools. • View Controller log files and security auditing files.

Only authenticated WLOC users can access the WLOC Administration Console.

Note: Except for security data, WLOC stores its configuration in XML files. If you prefer, you can configure the WLOC components by editing the XML directly using a validating text editor instead of using the WLOC Administration Console. In this case, you will need to restart the Controller or Agent in order for the changes to take effect.

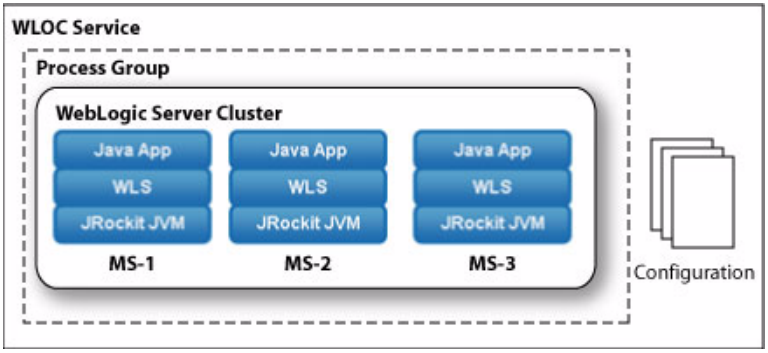
You **cannot** use the WLOC Administration Console to:

- Add servers to a BEA WebLogic Server domain. You can use the WebLogic Server Administration Console or the WebLogic Scripting Tool to add servers to a domain.
- Modify the configuration of a VMware ESX Server. Use the VMware Virtual Center to modify the ESX Server configuration.

Defining Services to Organize Processes

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

Figure 6 Defining a Service—Cluster of WebLogic Server Instances



For each process group you specify the following:

- The number of instances of the process groups to create initially and the minimum and maximum number of process instances allowed for the service.
- The minimum amount of resources that the process requires and an upper limit of resources that the process can use. WLOC reserves the minimal resources for exclusive use by the process and grants additional resources if they are available.
- A priority for the service, which WLOC uses to resolve conflicts when more than one service fails to meet its SLA at the same time.
- Information required by WLOC in order to deploy processes, including the main class, JVM startup arguments, and software dependencies.
- A ready metric, which WLOC uses to determine when a process has been started.
- Software dependencies including the name and location of the software that the processes require to run.

Defining Policies to Enforce SLAs

You can define one or more policies that specify the deployment or runtime requirements (constraints) for the service and the actions to take if the SLA constraint is not met. For example, a policy can expand or shrink a service's footprint in response to the runtime environment.

You can place constraints on a process, a group of processes, or all processes in a service. If your managed processes expose management data through Java Management Extensions (JMX), you can define a constraint based upon the value of an MBean attribute in your processes or based upon a calculated value derived form an MBean attribute in your process or group of processes.

When a constraint is violated, WLOC invokes a Java class (action) that you configure. WLOC provides actions that you can configure to do the following:

- Start or stop a process
- Invoke an MBean operation that a process exposes as part of its JMX interface
- Request additional computing resources from Hypervisor software
- Display alert messages in the WLOC Administration Console
- Send email, SNMP traps, JMS messages, or JMX notifications

In addition, you can define policies that trigger actions at a predefined time.

You can combine actions into an action pipeline that specifies a sequence of actions to invoke. You can invoke actions or action pipelines from service policies or manually from the WLOC Administration Console.

Example

For example, you create a WLOC service that specifies a process group for a collection of externally-facing Web services, all of which run on a single WebLogic Server cluster. You can configure the process group as follows:

- For the resource minimum, you reserve 400 CPU cycles and 600 MB of RAM. For the maximum, you allow the service to use up to 800 CPU cycles and 800 MB of RAM.
- For the resource priority, you specify the highest priority over all other services.
- For the initial deployment state, you configure WLOC to start the cluster's Administration Server and 2 Managed Servers.
- You create a policy that starts an additional cluster member during business hours.

- You create another policy that starts 2 additional members if servlet response time drops below 2 seconds and that stops the additional members if response time is faster than 0.1 second.

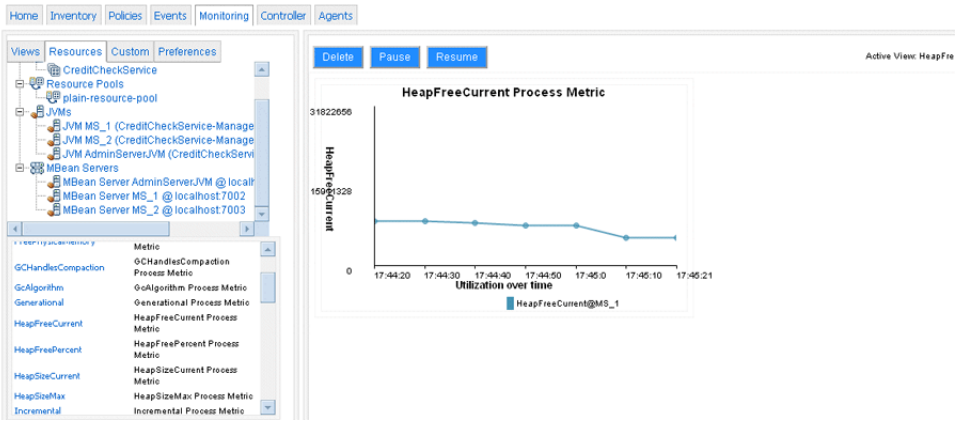
When you deploy the service, WLOC reserves 400 CPU cycles and 600 MB of RAM for exclusive use by the service. As WLOC adds processes to the service, it requests additional resources for use by the service up to the maximum. If the additional resources are currently being used by other processes, WLOC can remove resources from lower-priority processes, as long as each process retains its minimal reserve.

Monitoring WLOC Resources

You can monitor the performance of WLOC resources in the following ways:

- For all active resources, access charts and graphs to describe the amount of resources the service is using from a resource pool relative to the amount of resources available and the runtime statistics from each JVM within the service.

Figure 7 Monitoring WLOC Resources—Current Free Heap Utilization



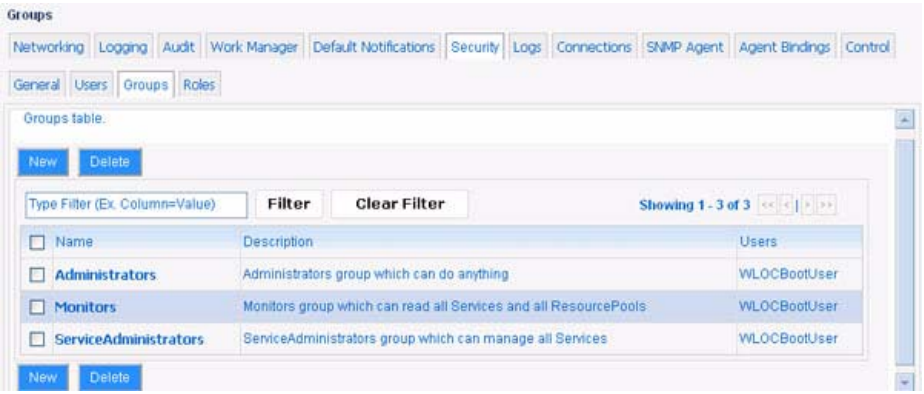
- Display information about events and actions that have occurred in your environment and the time that they occurred; and actions that are currently pending and that require approval.
- Review notifications that are triggered when a service operates outside of a policy constraint. You can configure WLOC to generate the following types of notifications: Java Message Service (JMS), Java Management Extensions (JMX), Simple Mail Transfer Protocol (SMTP), and Simple Network Management Protocol (SNMP).

- View log messages about events such as the deployment of services or the failure of one or more actions.
- View audit messages that capture changes to the Controller, Agents, or service configuration.

Managing WLOC Security

To secure access to the WLOC Administration Console, WLOC uses role-based access control which enables you to assign different levels of privileges to different users or groups. WLOC provides a set of security roles with pre-configured 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 security roles. You use the WLOC Administration Console to create users and assign them to groups or directly to security roles.

Figure 8 Managing WLOC Security



Related Information

The WLOC documentation set includes the following:

- *Installation Guide*—Describes how to install WLOC.
- *Use Case Example*— Provides a basic use-case example for WLOC.
- *Configuration Guide*—Describes how to configure and manage the WLOC Controller and Agents, configure services and policies to manage services, and configure security. It also describes how to use WLOC to monitor, log, and audit the operations of your services and resources.
- *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 Configuration 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.
- *Supported Configurations*—Platforms on which you can run WLOC Controllers and Agents.

Glossary

Action

A Java class that can display alert messages in the WLOC Administration Console, send notifications, or change the runtime state of a service. Actions can be adjudicated, requiring user input before WLOC invokes the action pipeline.

Actions can be invoked by the Controller as part of enforcing a policy, or manually from the WLOC Administration Console.

Action pipeline

A sequence of actions that WLOC invokes when a process operates beyond a constraint. Each policy can define at most one action pipeline. Action pipelines can be adjudicated, requiring user input before WLOC invokes the action pipeline.

Action pipelines can be invoked by the Controller as part of enforcing a policy, or manually from the WLOC Administration Console.

Agent

The WLOC component that interacts with the hosts of WLOC processes. Agents discover the computing resources that are available for allocation by WLOC, invoke commands on behalf of the Controller, and provide monitoring data to the Controller. WLOC includes the following types of Agents:

- **Plain Agent** for managing any type of Java process. A Plain Agent renders the resources from the machine on which it resides as a WLOC resource pool.
- **ESX Agent** for managing only virtualized Java applications that run on LiquidVM (LVM), such as WebLogic Server Virtual Edition (WLS-VE) running in a

virtualized environment. An ESX Agent renders VMware resource pools as WLOC resource pools; it can reside on any machine in your operations center.

Application
See “Service” on page 20 and “Process” on page 19.

Compute power
See “CPU cycles” on page 18.

Compute resource
See “Resource pool” on page 20.

CPU cycles
A measurement of the CPU resources that a resource pool can supply and that a service needs. The measurement is normalized across CPU architectures so that a megahertz of processing on an i386 processor is comparable to a megahertz on other types of processors.

Constraints
Runtime requirements that define the service level agreements (SLAs) for a service. When processes operate outside of constraints, policies can trigger actions or action pipelines.

WLOC provides preconfigured constraints and users can configure additional constraints. The preconfigured constraints enable users to place requirements on the most common and important measurements of health and performance.

Users can configure additional constraints based on time or the values of MBean attributes. For resource pools that are managed by ESX Agents, users can configure constraints based on the consumption of resources.

Controller
The centralized component that gathers data from Agents about the operating environment and deployed services to deliver adaptive management; evaluates constraints; and issues instructions to Agents to carry out actions. The Controller hosts the WLOC Administration Console. Each WLOC environment contains a single Controller.

Hypervisor
Virtualization software that allows multiple operating systems to run on a single physical computer at the same time.

JMS
Java Message Service.

JVM
Java Virtual Machine.

JMX
Java Management Extensions.

WLOC environment
The collection of all resource pools, services, processes, Agents and Controller in a single installation of WLOC.

Machine
Either a physical machine or a virtual machine.

Managed environment
See WLOC Environment.

Metric
A numeric runtime value that describes the performance of a process or process group. Some metrics are aggregations or calculations of raw (observed) data. Policies set constraints on metrics.

Physical host
The physical machine that is hosting an Agent or a virtual machine.

Policies
Runtime requirements for a service and actions to take when the service operates outside the requirements. Each policy consists of two parts: a single constraint and an action or pipeline of actions. You can create multiple policies for each service. Policies can apply to all processes in a service, to a group of processes (process type), or to a single process.

Process
A program that WLOC manages. For example, a single WebLogic Server Managed Server is a process. From the perspective of WLOC, a Java process consists of an entire JVM stack and includes any application server and applications being managed by the Controller.

Process group
A collection of processes in a service for which policies can be written. For example, you can create a process group that contains three WebLogic Server instances. You can write a policy that starts all three server instances when the service is deployed. In addition, you can write a policy that increases a specific resource for all three servers when a constraint is surpassed.

Process type

See “Process group” on page 19.

Resource pool

A virtual environment in which you can deploy WLOC services. Each resource pool provides access to physical computing resources (such as CPU cycles, memory, and disk space) and pre-installed software that a service needs to run. A resource pool also contains a description of the failover capabilities of the machines that host the computing and software resources.

Rules

See “Constraints” on page 18.

Service

A collection of one or more processes that WLOC manages as a unit. Each process in a service is a software stack starting from the Java Virtual machine (JVM) and including the classes that are running in the JVM.

For example, you can create a service for managing a single WebLogic Server instance on which you have deployed a single Java EE application, or you can create a service for managing all WebLogic Server instances in a clusters. A service specifies:

- Requirements for the physical computing resources that are needed to run all of its processes. These resource requirements are expressed as a range of CPU cycles, memory, and disk space.
- An optional set of policies that define a service level agreement and actions to take when the service is operating outside of the SLA.
- Metadata that defines the Java classes or other executables that make up the service's processes. For more information, see Services.