



BEA WebLogic Operations Control

Use Case Example

Contents

Introduction

Main Steps	1-2
Related Documents	1-2

Configure the WLOC Resource Environment

Step 1: Install and Create the Plain Agent	2-1
Install the Plain Agent	2-2
Create the Plain Agent	2-2
Agent Directory Structure	2-9
Agent Configuration File	2-10
Step 2: Install and Create the Controller	2-13
Install the Controller	2-13
Create the Controller	2-14
Controller Directory Structure	2-20
Controller Configuration File	2-21

Establish the WLOC Runtime Environment

Step 1: Start the Plain Agent	3-1
Step 2: Start the Controller	3-2
Step 3: Start the WLOC Administration Console	3-3

Define the Service Under Management

Step 1: Create the Service and Process Groups	4-3
---	-----

Define the Administration Server Process Group	4-5
Define the Managed Servers Process Group	4-11
Define Resource Requirements for the Service	4-17
View Deployment Policy	4-19
Create Services Using Helper Methods	4-20
CreditCheckService Service Metadata Configuration	4-20
AdminServer Process Group Metadata Configuration	4-23
Step 2: Define the Adaptive Runtime Policies	4-25
Runtime Policy Metadata Configuration	4-32

Deploy the Service Against Available Resources

Deployment Scenario	5-2
Deploy the Service	5-3

Monitor WLOC Services and Resources

Create a View	6-2
Browse the Resources Pane	6-2

Introduction

BEA WebLogic Operations Control (WLOC) is a management environment that can increase the efficiency of your operations center by hiding the complexity of the underlying operational environment and presenting the resources and Java applications in a simple supply and demand mode.

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

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

This document provides a basic use-case example for WLOC. In this use case we will describe the steps to:

- Establish a resource environment by installing and creating a Plain Agent and Controller
- Establish the runtime environment by starting the Agent, Controller, and the WLOC Administration Console.
- Use the Administration Console to define a service to be managed.
- Deploy the service against the available resources.
- Monitor the WLOC service and the resource environment with the WLOC Administration Console.

Main Steps

The following table summarizes the main steps demonstrated in this example.

Table 1-1 Use Case Example Main Steps

To complete this task . . .	We demonstrate how to perform the following steps . . .
Establish the WLOC resource environment	<ul style="list-style-type: none"> • “Step 1: Install and Create the Plain Agent” on page 2-1 • “Step 2: Install and Create the Controller” on page 2-13
Establish the WLOC runtime environment	<ul style="list-style-type: none"> • “Step 1: Start the Plain Agent” on page 3-1 • “Step 2: Start the Controller” on page 3-2 • “Step 3: Start the WLOC Administration Console” on page 3-3
Define services under management	<ul style="list-style-type: none"> • “Step 1: Create the Service and Process Groups” on page 4-3 • “Step 2: Define the Adaptive Runtime Policies” on page 4-25
Deploy services against available resources	<ul style="list-style-type: none"> • “Deployment Scenario” on page 5-2 • “Deploy the Service” on page 5-3
Monitor WLOC services and resource environment with the WLOC Administration Console	<ul style="list-style-type: none"> • “Create a View” on page 6-2 • “Browse the Resources Pane” on page 6-2

Related Documents

The WLOC documentation set includes the following:

- [Installation Guide](#)—Describes how to install and uninstall the WLOC components.
- [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 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.

Introduction

Configure the WLOC Resource Environment

In a WLOC environment, resource pools provide 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.

To establish a WLOC resource environment, you need to configure a controller and one or more agents. You can do so using the WLOC Configuration Wizard. When you configure an Agent, you configure its resource pool. When you configure the Controller, you bind it to the Agents so that it can get information about the resources and deploy services accordingly.

The Controller also hosts the WLOC Administration Console which provides a graphical interface into the WLOC environment.

In this example, we will install and create a Plain Agent and a Controller.

The main steps in this topic include:

- [Step 1: Install and Create the Plain Agent](#)
- [Step 2: Install and Create the Controller](#)

Step 1: Install and Create the Plain Agent

A Plain Agent manages the computing resources for the physical machine on which the Agent is installed. You can configure a Plain Agent to allocate all or a subset of the available machine resources to WLOC services.

After you install the agent, you create it using the WLOC Configuration Wizard.

Use the following steps to install and create the Plain Agent.

Install the Plain Agent

The Plain Agent is installed as part of a complete WLOC installation, or can be selected individually using the Custom installation option. For details about installing WLOC, see the [WLOC Installation Guide](#).

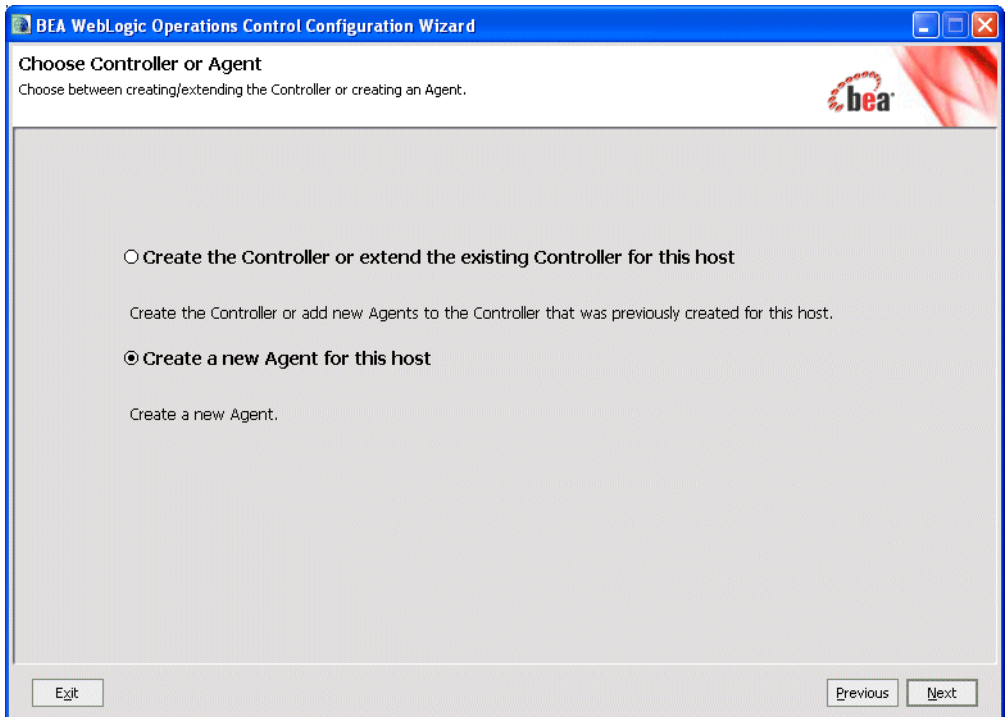
Create the Plain Agent

To create the Plain Agent, use the WLOC Configuration Wizard and complete the following steps:

1. From the Start Menu, select Start > WebLogic Operations Control 1.0 > WLOC Configuration Wizard.



2. In the **Welcome** window, click **Next**.
3. In the **Choose Controller or Agent** window, select **Create a new Agent for this host** and click **Next**.



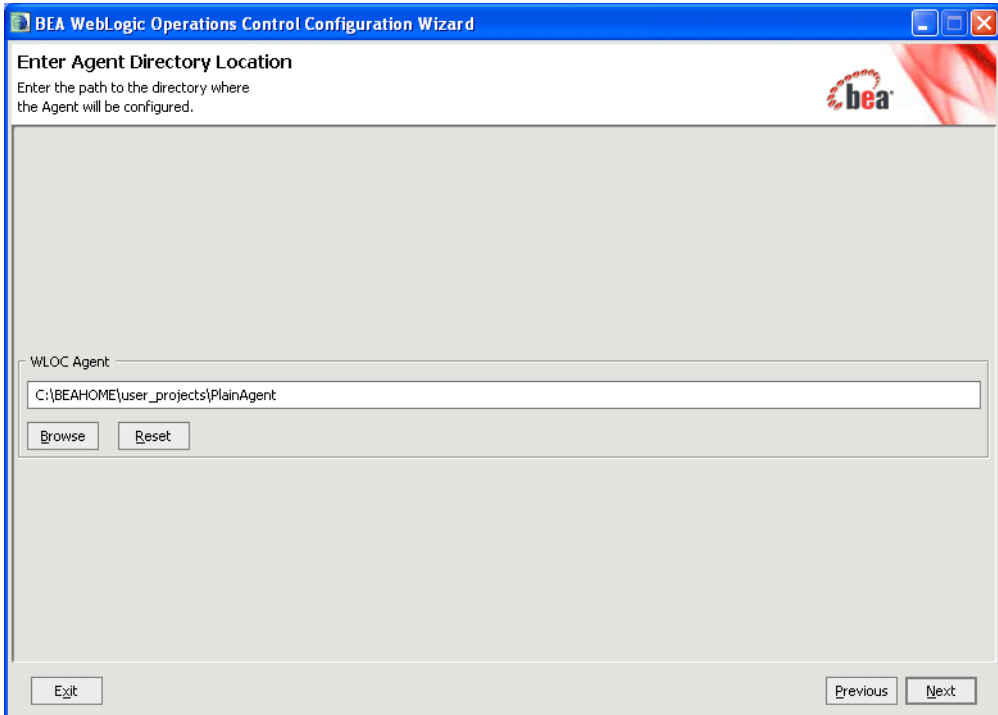
4. In the **Enter Agent Directory Location** window, specify the path and file name for the Agent and click **Next**.

By default, this directory is created in `BEA_HOME\user_projects\agent1`, but you can specify any name and directory location you choose.

Note that we used `C:\BEAHOME` as the `BEA_HOME` directory when we installed the WLOC software, therefore that `BEA_HOME` value is displayed as the default.

For this example, we accept the default `C:\BEAHOME\user_projects` location, and change the name of the directory to `PlainAgent`.

Configure the WLOC Resource Environment



5. In the **Configure Agent Connection Details** window, specify the following connection information for the Agent:

Table 2-1 Agent Connection Information

In this field . . .	Enter the following value . . .
Agent Name	PlainAgent
Agent Host	The URL for the host machine. In this example we use myhost.bea.com.
Agent Port	8001 (the default)
Agent Secure Port	8002 (the default)
Transfer Encryption Passphrase	Default

Table 2-1 Agent Connection Information

In this field . . .	Enter the following value . . .
Confirm Transfer Encryption Passphrase	Default
Security Mode	Unsecure (default)

BEA WebLogic Operations Control Configuration Wizard

Configure Agent Connection Details
Please specify the connection information for this Agent.

Agent Name*

Agent Host*

Agent Port*

Agent Secure Port*

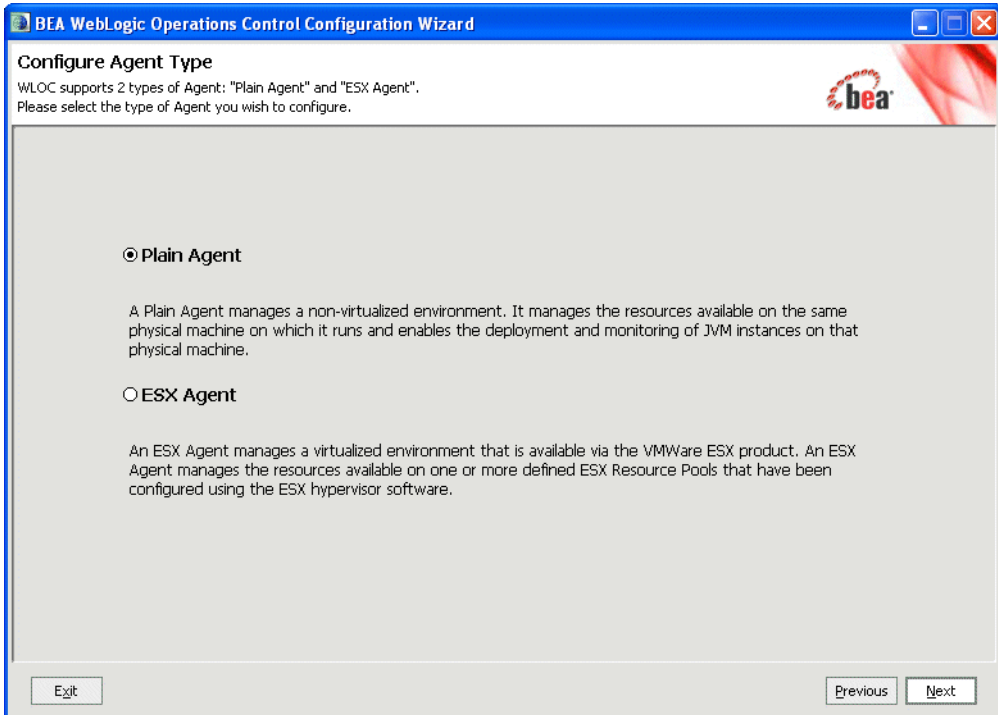
Transfer Encryption Passphrase*

Confirm Transfer Encryption Passphrase*

Security Mode*

6. Click **Next** in the following two windows to accept the defaults:
 - Configure Agent Logging
 - Configure Agent Keystore Passwords
7. In the **Configure Agent Type** window, select **Plain Agent** and click **Next**.

Configure the WLOC Resource Environment



8. In the **Configure Plain Agent (1 of 2)** window, provide a name for the resource pool associated with this Agent and the CPU capacity available to the resource pool, as shown in the following table:

Table 2-2 Plain Agent Resource Pool Configuration

In this field . . .	Enter the following value . . .
Resource Pool Name	plain-resource-pool
Description	plain resource pool
CPU capacity (MHz)	512
Stdout Directory	Accept the default
Stderr Directory	Accept the default

Step 1: Install and Create the Plain Agent

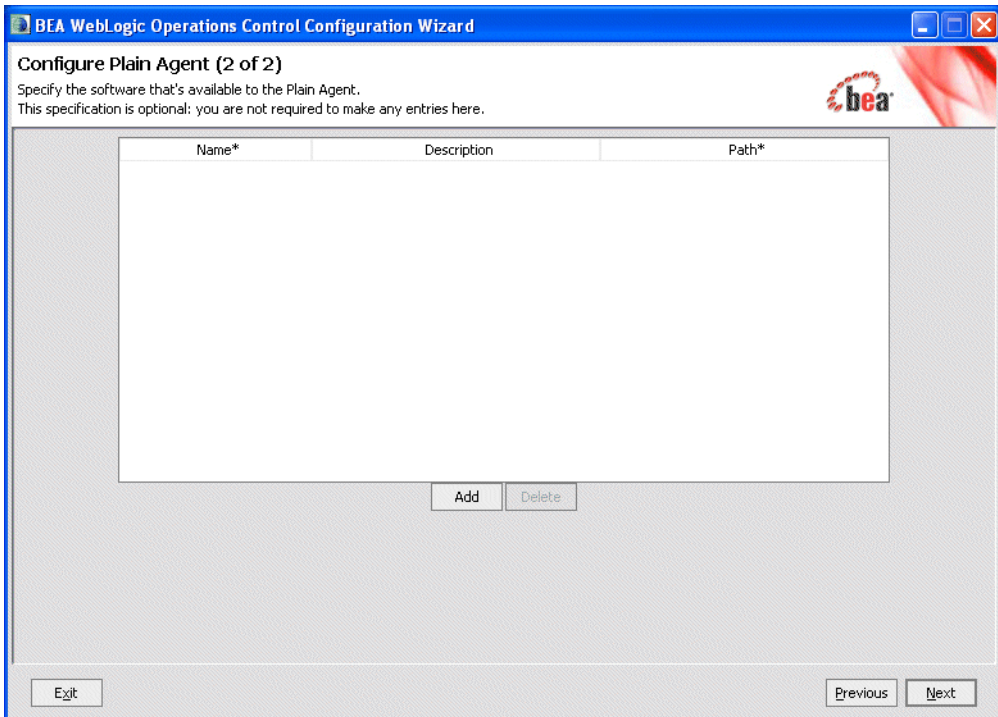
The screenshot shows a Windows-style window titled "BEA WebLogic Operations Control Configuration Wizard" with a sub-title "Configure Plain Agent (1 of 2)". The window contains the following fields and controls:

- Resource Pool Name***: A text box containing "plain-resource-pool".
- Description**: A text box containing "plain resource pool".
- CPU capacity (MHz)**: A text box containing "512".
- Stdout Directory**: A text box containing "C:\BEAHOME\user_projects\PlainAgent\stdout" with a "Browse" button to its right.
- Stderr Directory**: A text box containing "C:\BEAHOME\user_projects\PlainAgent\stderr" with a "Browse" button to its right.

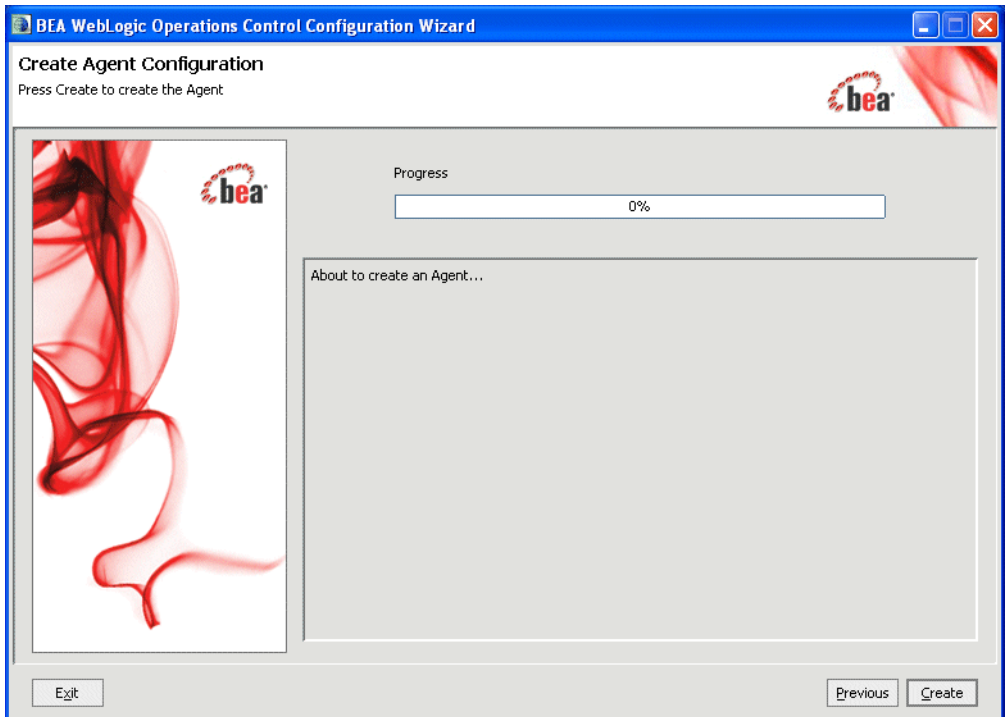
At the bottom of the window, there are three buttons: "Exit" on the left, and "Previous" and "Next" on the right.

9. In the **Configure Plain Agent (2 of 2)** window, you specify the available software you want to include in the resource pool. For this example, do not specify any additional software and click **Next**.

Configure the WLOC Resource Environment



10. In the **Create Agent Configuration** window, click **Create**.



11. After the Agent has been created, click **Done** to exit the WLOC Configuration Wizard.

Agent Directory Structure

After completing the Plain Agent installation and creation, the following directory structure is created in the `C:\BEAHOME\user_projects\PlainAgent` directory.

Configure the WLOC Resource Environment

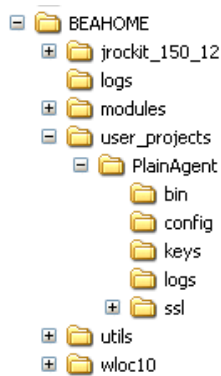


Table 2-2 describes the contents of these directories.

Table 2-3 Agent Directory Description

Directory	Description
bin	Commands to start the Agent, and to install and remove the Agent as a Windows service.
config	Agent configuration files.
keys	Encryption key used to encrypt clear text passwords.
logs	Agent log files.
ssl	Internal digital certificate and keystores for the Agent used for SSL communication with the Controller.

Agent Configuration File

When you create an Agent using the WLOC Configuration Wizard, the configuration is persisted in an XML file named `loc-agent-config.xml`. In this example, the file is created in the following directory:

```
c:\BEAHOME\user_projects\PlainAgent\config\loc-agent-config.xml
```

where:

`BEAHOME` is the BEA Home directory containing the WLOC installation, and `PlainAgent` is the name that we specified for the Agent Directory location in the Configuration Wizard.

After you have created the Agent using the Configuration Wizard, it can be modified using the Administration Console or by directly editing its configuration file.

The `loc-agent-config.xml` file created in this example is shown in [Listing 2-1](#)

Listing 2-1 Sample `loc-agent-config.xml` File

```
<?xml version="1.0" encoding="UTF-8"?><loc-agent xmlns="bea.com/loc/agent"
xmlns:loc="http://bea.com/loc">
  <name>PlainAgent</name>
  <description>PlainAgent</description>
  <network>
    <loc:host>myhost.bea.com</loc:host>
    <loc:components>
      <loc:component>
        <loc:name>ListenPorts</loc:name>
        <loc:description>ListenPorts</loc:description>
        <loc:port>8001</loc:port>
        <loc:secure-port>8002</loc:secure-port>
      </loc:component>
    </loc:components>
  </network>
  <use-secure-connections>false</use-secure-connections>
  <logging>
    <loc:file-severity>Info</loc:file-severity>

<loc:base-file-name>C:/BEAHOME/user_projects/PlainAgent/logs/Agent.log</lo
c:base-file-name>
  <loc:rotation-type>BySize</loc:rotation-type>
  <loc:rotation-size>5000</loc:rotation-size>
  <loc:rotation-time>00:00</loc:rotation-time>
  <loc:file-rotation-dir>./logs/logrotDir</loc:file-rotation-dir>
  <loc:number-of-files-limited>true</loc:number-of-files-limited>
  <loc:rotated-file-count>5</loc:rotated-file-count>
  <loc:rotation-time-span>24</loc:rotation-time-span>
  <loc:rotation-time-span-factor>3500000</loc:rotation-time-span-factor>
  <loc:rotation-on-startup-enabled>true</loc:rotation-on-startup-enabled>
  <loc:stdout-severity>Info</loc:stdout-severity>
```

Configure the WLOC Resource Environment

```
</logging>
<audit>
  <loc:base-file-name>./logs/audit.log</loc:base-file-name>
  <loc:rotation-type>BySize</loc:rotation-type>
  <loc:rotation-size>300</loc:rotation-size>
  <loc:rotation-time>00:00</loc:rotation-time>
  <loc:file-rotation-dir>./logs/logrotmdir</loc:file-rotation-dir>
  <loc:number-of-files-limited>true</loc:number-of-files-limited>
  <loc:rotated-file-count>50</loc:rotated-file-count>
  <loc:rotation-time-span>24</loc:rotation-time-span>
  <loc:rotation-time-span-factor>50</loc:rotation-time-span-factor>
  <loc:rotation-on-startup-enabled>true</loc:rotation-on-startup-enabled>
  <loc:enabled>true</loc:enabled>
  <loc:scope>
    <loc:type>All</loc:type>
  </loc:scope>
</audit>
<work-managers>
  <loc:work-manager>
    <loc:name>WM</loc:name>
    <loc:description>WM</loc:description>
    <loc:max-threads-constraint>64</loc:max-threads-constraint>
    <loc:min-threads-constraint>3</loc:min-threads-constraint>
  </loc:work-manager>
  <loc:work-manager>
    <loc:name>ResourceBrokerAgent-WM</loc:name>
    <loc:description>ResourceBrokerAgent-WM</loc:description>
    <loc:max-threads-constraint>15</loc:max-threads-constraint>
    <loc:min-threads-constraint>3</loc:min-threads-constraint>
  </loc:work-manager>
  <loc:work-manager>
    <loc:name>AgentRuntime-WM</loc:name>
    <loc:description>AgentRuntime-WM</loc:description>
    <loc:max-threads-constraint>15</loc:max-threads-constraint>
    <loc:min-threads-constraint>3</loc:min-threads-constraint>
  </loc:work-manager>
</work-managers>
<encryption>
```

```

    <password>{Salted-3DES}zwrq/caNuFEi4S5AeAA11A==</password>
  </encryption>
  <resource-pools>
    <plain-resource-pool>
      <name>plain-resource-pool</name>
      <description>plain resource pool</description>
      <cpu-capacity>512</cpu-capacity>
      <stdout-dir>C:\BEAHOME\user_projects\PlainAgent\stdout</stdout-dir>
      <stderr-dir>C:\BEAHOME\user_projects\PlainAgent\stderr</stderr-dir>
    </plain-resource-pool>
  </resource-pools>
</loc-agent>

```

For information about the elements of the `loc-agent-config.xml` Agent configuration file, see the [Agent Configuration Schema Reference](#).

Step 2: Install and Create the Controller

Every WLOC environment includes a single Controller and one or more Agents. The Controller is the central component that gathers data about the operating environment from Agents. The Controller uses the data that it gathers to intelligently deploy new services and to evaluate and enforce policies for all services in the environment. The Controller also hosts the WLOC Administration Console.

After you install the Controller, you configure it using the WLOC Configuration Wizard.

Although you can install the Agent and the Controller on different physical machines, in this example, the Controller is installed on the same machine as the Plain Agent.

Use the following steps to install and configure the controller.

Install the Controller

The Controller is installed as part of a complete WLOC installation, or can be selected individually using the Custom installation option. For details about installing WLOC, see the [WLOC Installation Guide](#).

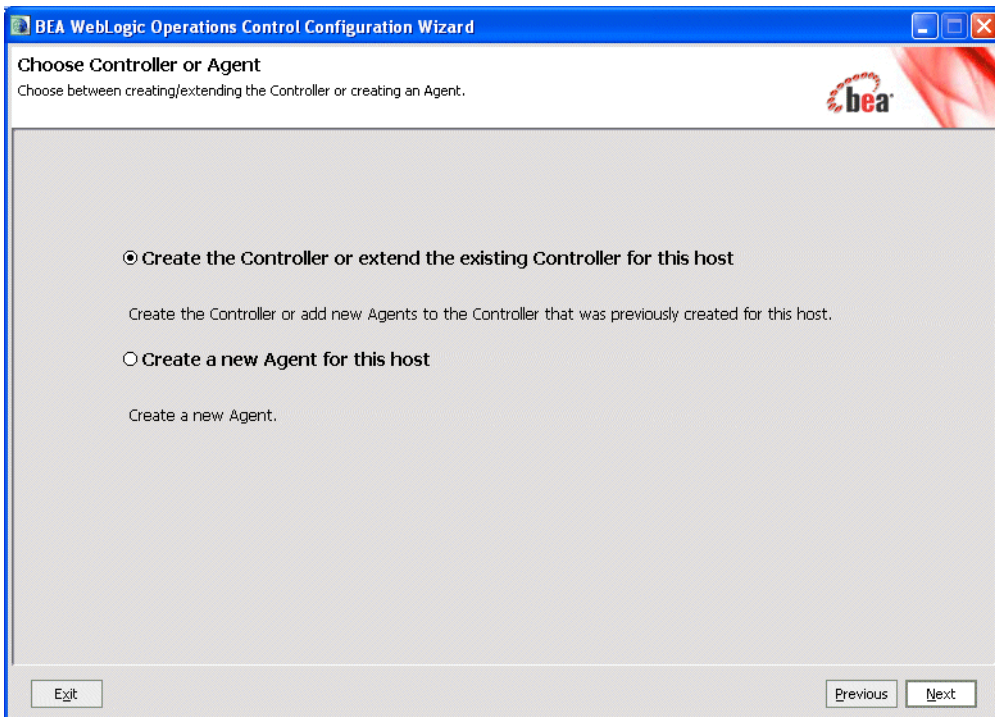
Create the Controller

To create the Controller, use the WLOC Configuration Wizard and complete the following steps:

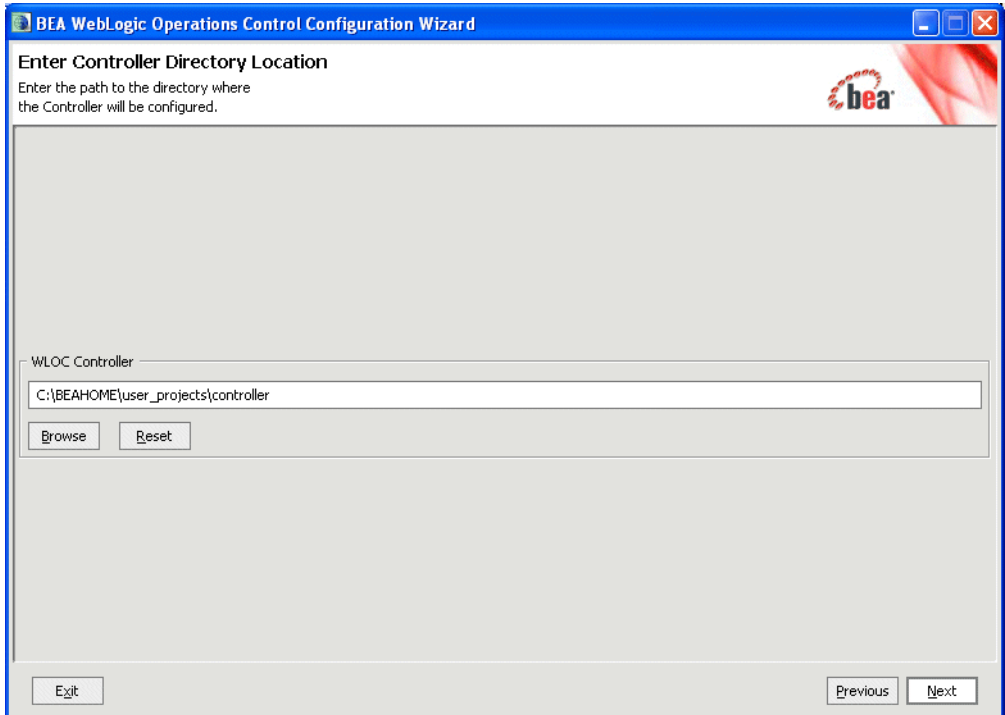
1. From the Start Menu, select Start > WebLogic Operations Control 1.0 > WLOC Configuration Wizard.



2. In the **Welcome** window, click **Next**.
3. In the **Choose Controller or Agent** window, select **Create the Controller or extend the existing Controller for this host** and click **Next**.



4. In the **Enter Controller Directory Location** window, we accept the default path and filename for the Controller and click **Next**.



5. In the Enter Controller Connection Data window, specify the following connection information for the Controller.

Table 2-4 Controller Connection Information

In this field . . .	Enter the following value . . .
Controller Host	The URL for the host machine (myhost.bea.com)
Console Port	9001 (the default)
Console Secure Port	9002 (the default)
Console Mode	Both
Internal Port	9003 (the default)

Configure the WLOC Resource Environment

Table 2-4 Controller Connection Information

In this field . . .	Enter the following value . . .
Internal Secure Port	9004 (the default)
Security Mode	Unsecure (default)

The screenshot shows a window titled "BEA WebLogic Operations Control Configuration Wizard" with the subtitle "Enter Controller Connection Data". Below the subtitle is the instruction "Enter the ports the Controller will use." and the BEA logo. The form contains the following fields and values:

- Controller Host*: myhost.bea.com
- Console Port*: 9001
- Console Secure Port*: 9002
- Console Mode*: Both (dropdown menu)
- Internal Port*: 9003
- Internal Secure Port*: 9004
- Security Mode*: Unsecure (dropdown menu)

At the bottom of the window, there are three buttons: "Exit", "Previous", and "Next".

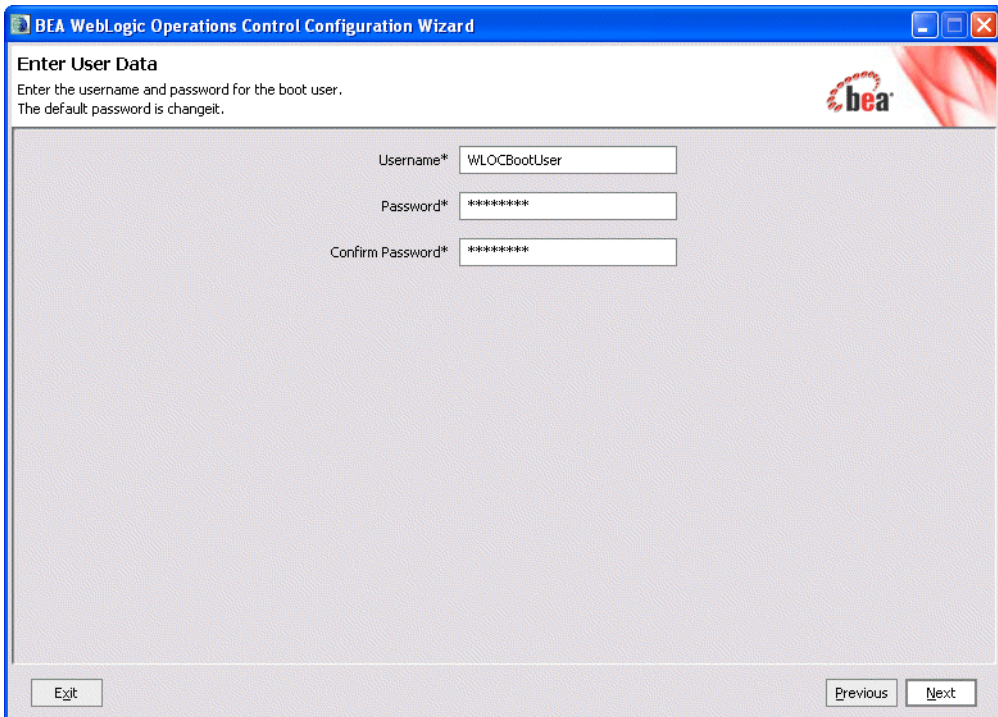
6. Accept the default options in the following windows and click **Next**:
 - **Configure Controller Logging**
 - **Configure Controller Notifications (1 of 3)**
 - **Configure Controller Notifications (2 of 3)**
 - **Configure Controller Notifications (3 of 3)**
7. In the **Configure Agents for this Controller** window, click **Add** to bind the Plain Agent created previously to this Controller.

The fields are populated with the default data for your machine. Enter the name PlainAgent name in the Name field, accept the defaults for the remaining fields and click **Next**.

Name*	Agent's Hostname*	Port*	Secure Po...	State*	Passphrase*	Confirm Passphrase*
PlainAgent	host.bea.com	8001	8002	Enabled	*****	*****

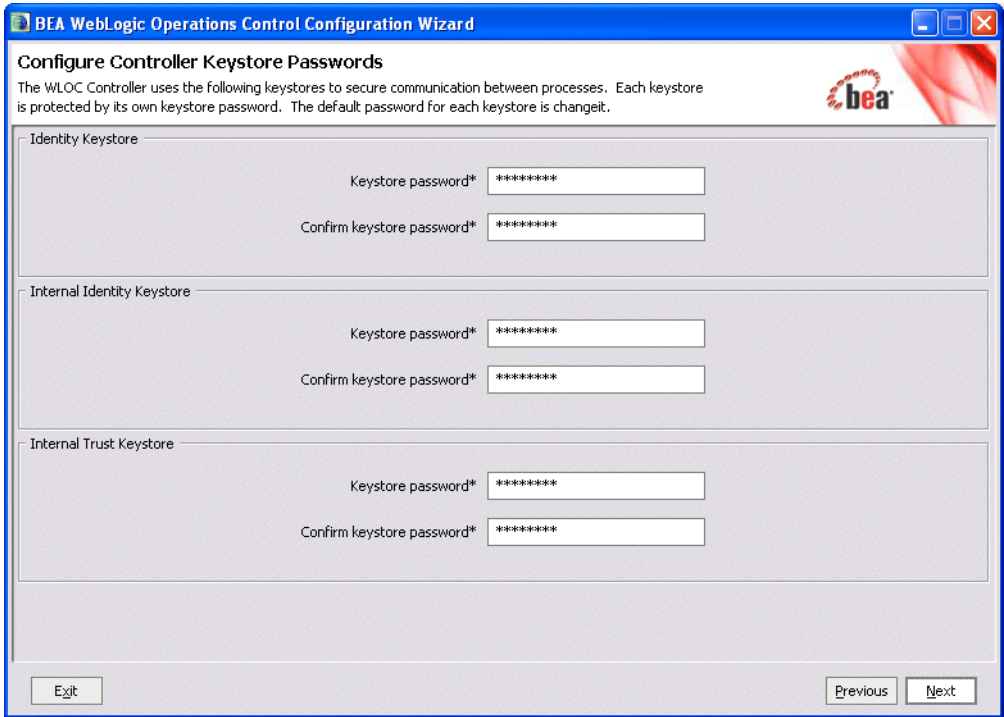
8. Click **Next** in the **Use SSH for WLOC ESX Agents** window. In this example, only a Plain Agent is configured.
9. In the **Enter User Data** window, specify a username and password for the boot user. For this example, accept the defaults. Note that the default username is WLOCBootUser and the default password is changeit:

Configure the WLOC Resource Environment



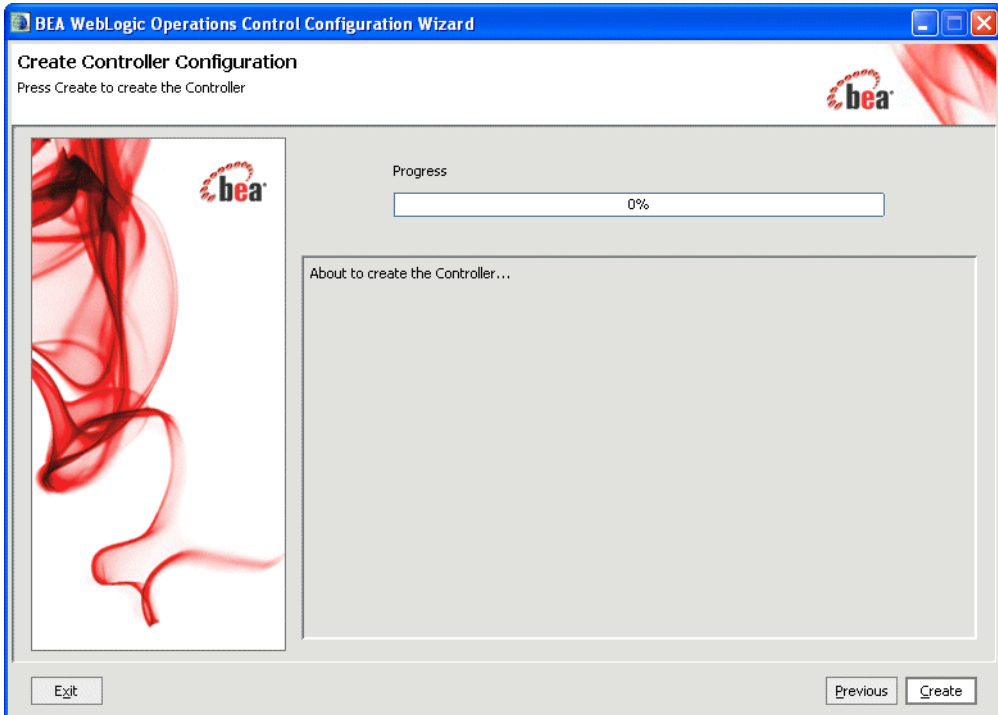
The screenshot shows a window titled "BEA WebLogic Operations Control Configuration Wizard". The main heading is "Enter User Data". Below the heading, there is a sub-heading "Enter the username and password for the boot user." and a note "The default password is changeit." in the top left corner. In the top right corner, there is the BEA logo and a red ribbon graphic. The central area contains three input fields: "Username*" with the value "WLOCBootUser", "Password*" with "*****", and "Confirm Password*" with "*****". At the bottom, there are three buttons: "Exit" on the left, and "Previous" and "Next" on the right.

10. In the **Configure Controller KeyStore Passwords**, accept the default passwords and click **Next**.



11. In the **Create Controller Configuration** window, click **Create**.

Configure the WLOC Resource Environment



12. After the Controller has been created, click **Done** to exit the WLOC Configuration Wizard.

Controller Directory Structure

After completing the Controller installation and creation, the following directory structure is created in the `C:\BEAHOME\user_projects\controller` directory.

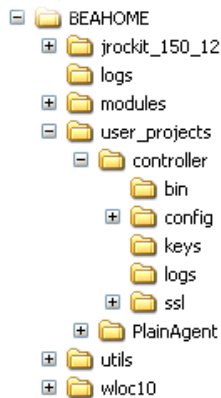


Table 2-2 describes the contents of these directories.

Table 2-5 Controller Directory Description

Directory	Description
bin	Commands to start the Controller, and to install and remove the Controller as a Windows service.
config	Controller configuration files.
keys	Encryption keys used to encrypt clear text passwords and data.
logs	Controller log files
ssl	Internal digital certificate and keystores for the Controller used for SSL communication with the Agents.

Controller Configuration File

When you create a Controller using the WLOC Configuration Wizard, the configuration is persisted in an XML file named `loc-controller-config.xml`. In this example, the file is created in the following directory:

```
BEAHOME/user_projects/controller/config/loc-controller-config.xml
```

where:

BEAHOME is the BEA Home directory containing the WLOC installation, and `controller` is the name that we specified for the Controller Directory location in the Configuration Wizard.

Configure the WLOC Resource Environment

After you have created the Controller using the Configuration Wizard, it can be modified using the Administration Console or by directly editing its configuration file.

The `loc-controller-config.xml` file created in this example is shown in [Listing 2-2](#)

Listing 2-2 Sample `loc-controller-config.xml` File

```
<?xml version="1.0" encoding="UTF-8"?><loc-controller
xmlns="bea.com/loc/controller" xmlns:loc="http://bea.com/loc">
  <network>
    <loc:host>host.bea.com</loc:host>
    <loc:components>
      <loc:component>
        <loc:name>Console</loc:name>
        <loc:description>Console</loc:description>
        <loc:port>9001</loc:port>
        <loc:secure-port>9002</loc:secure-port>
      </loc:component>
      <loc:component>
        <loc:name>InternalCommunication</loc:name>
        <loc:description>InternalCommunication</loc:description>
        <loc:port>9003</loc:port>
        <loc:secure-port>9004</loc:secure-port>
      </loc:component>
    </loc:components>
  </network>
  <use-secure-connections>>false</use-secure-connections>
  <console-mode>BOTH</console-mode>
  <logging>
    <loc:file-severity>Info</loc:file-severity>

  <loc:base-file-name>C:/BEAHOME/user_projects/controller/logs/Controller.log</loc:base-file-name>
    <loc:rotation-type>BySize</loc:rotation-type>
    <loc:rotation-size>500</loc:rotation-size>
    <loc:rotation-time>00:00</loc:rotation-time>
    <loc:file-rotation-dir>./logs/logrotmdir</loc:file-rotation-dir>
    <loc:number-of-files-limited>>true</loc:number-of-files-limited>
```

```

    <loc:rotated-file-count>5</loc:rotated-file-count>
    <loc:rotation-time-span>24</loc:rotation-time-span>
    <loc:rotation-time-span-factor>3500000</loc:rotation-time-span-factor>
    <loc:rotation-on-startup-enabled>true</loc:rotation-on-startup-enabled>
    <loc:stdout-severity>Info</loc:stdout-severity>
</logging>
<audit>
  <loc:base-file-name>./logs/audit.log</loc:base-file-name>
  <loc:rotation-type>BySize</loc:rotation-type>
  <loc:rotation-size>300</loc:rotation-size>
  <loc:rotation-time>00:00</loc:rotation-time>
  <loc:file-rotation-dir>./logs/logrotmdir</loc:file-rotation-dir>
  <loc:number-of-files-limited>true</loc:number-of-files-limited>
  <loc:rotated-file-count>50</loc:rotated-file-count>
  <loc:rotation-time-span>24</loc:rotation-time-span>
  <loc:rotation-time-span-factor>50</loc:rotation-time-span-factor>
  <loc:rotation-on-startup-enabled>true</loc:rotation-on-startup-enabled>
  <loc:enabled>true</loc:enabled>
  <loc:scope>
    <loc:type>ControllerConfiguration</loc:type>
    <loc:type>ServiceConfiguration</loc:type>
    <loc:type>Rules</loc:type>
    <loc:type>ControllerAction</loc:type>
    <loc:type>Adjudication</loc:type>
    <loc:type>AgentConfiguration</loc:type>
  </loc:scope>
</audit>
<work-managers>
  <loc:work-manager>
    <loc:name>WM</loc:name>
    <loc:description>WM</loc:description>
    <loc:max-threads-constraint>64</loc:max-threads-constraint>
    <loc:min-threads-constraint>3</loc:min-threads-constraint>
  </loc:work-manager>
  <loc:work-manager>
    <loc:name>ResourceBroker-WM</loc:name>
    <loc:description>ResourceBroker-WM</loc:description>
    <loc:max-threads-constraint>15</loc:max-threads-constraint>

```

Configure the WLOC Resource Environment

```
<loc:min-threads-constraint>3</loc:min-threads-constraint>
</loc:work-manager>
<loc:work-manager>
  <loc:name>Action-Purge-WM</loc:name>
  <loc:description>Action-Purge-WM</loc:description>
  <loc:max-threads-constraint>15</loc:max-threads-constraint>
  <loc:min-threads-constraint>3</loc:min-threads-constraint>
</loc:work-manager>
<loc:work-manager>
  <loc:name>ExecuteEngine-WM</loc:name>
  <loc:description>ExecuteEngine-WM</loc:description>
  <loc:max-threads-constraint>15</loc:max-threads-constraint>
  <loc:min-threads-constraint>3</loc:min-threads-constraint>
</loc:work-manager>
<loc:work-manager>
  <loc:name>ProcessRuntime-WM</loc:name>
  <loc:description>ProcessRuntime-WM</loc:description>
  <loc:max-threads-constraint>15</loc:max-threads-constraint>
  <loc:min-threads-constraint>3</loc:min-threads-constraint>
</loc:work-manager>
<loc:work-manager>
  <loc:name>Actions-WM</loc:name>
  <loc:description>Actions-WM</loc:description>
  <loc:max-threads-constraint>15</loc:max-threads-constraint>
  <loc:min-threads-constraint>3</loc:min-threads-constraint>
</loc:work-manager>
</work-managers>
<heartbeat-interval>20</heartbeat-interval>
<reconnect-attempts>3</reconnect-attempts>
<agents>
  <agent>
    <name>PlainAgent</name>
    <host>host.bea.com</host>
    <port>8001</port>
    <secure-port>8002</secure-port>
    <state>Enabled</state>
    <password>{Salted-3DES}8kenEcTMhnFzQI/LXLZeMQ==</password>
  </agent>
```



```

</agents>
<lvm-ssh-config>
  <public-key-file/>
</lvm-ssh-config>
<notification>
  <smtp>
    <name>LOC EMail Notification Service</name>
    <description>LOC EMail Notification Service</description>
    <to-address>somebody@somecompany.com</to-address>
    <from-address>LOCController@somecompany.com</from-address>
    <smtp-server>smtpserver.somecompany.com</smtp-server>
    <enabled>>false</enabled>
  </smtp>
  <jms>
    <name>LOC JMS Notification Service</name>
    <description>LOC JMS Notification Service</description>

<destination-jndi-name>com.bea.adaptive.loc.notification.JMSNotifier</dest
ination-jndi-name>

<connection-factory-jndi-name>QueueConnectionFactory</connection-factory-j
ndi-name>
  <jndi-properties>
    <initial-factory>org.mom4j.jndi.InitialCtxFactory</initial-factory>
    <provider-url>xcp://somehost:9911</provider-url>
    <security-principal>system</security-principal>
    <password>{Salted-3DES}+fzbeHi7Ydhh+A1csPgYPA==</password>
  </jndi-properties>
  <enabled>>false</enabled>
</jms>
<jmx>
  <name>JMX Notification Service</name>
  <description>JMX Notification Service</description>
  <enabled>>false</enabled>
</jmx>
<snmp>
  <name>LOC SNMP Notification Service</name>
  <description>LOC SNMP Notification Service</description>

```

Configure the WLOC Resource Environment

```
<agent>
  <name>MySNMPAgent</name>
  <description>MySNMPAgent</description>
  <host>somehost</host>
  <port>2002</port>
  <trap-version>SNMPv2</trap-version>
  <enable-inform>>false</enable-inform>
</agent>
<trap-destinations>
  <destination>
    <name>testTrapDest</name>
    <description>testTrapDest</description>
    <host>somehost</host>
    <port>1642</port>
    <community>public</community>
    <security-level>noAuthNoPriv</security-level>
  </destination>
</trap-destinations>
<enabled>>false</enabled>
</snmp>
</notification>
</loc-controller>
```

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

What's Next?

After installing and creating the Plain Agent and Controller, go to [Chapter 3, “Establish the WLOC Runtime Environment,”](#) which describes how to start the Agent, the Controller, and the WLOC Administration Console.

Establish the WLOC Runtime Environment

Now that we have installed and configured the Plain Agent and the Controller to establish the resource environment for this example, we need to start each of them and the Administration Console. This will establish the runtime environment needed to define the service to be managed.

The tasks in this topic include:

- [Step 1: Start the Plain Agent](#)
- [Step 2: Start the Controller](#)
- [Step 3: Start the WLOC Administration Console](#)

Step 1: Start the Plain Agent

To start the Plain Agent:

1. Open a Command Prompt window.
2. Navigate to `\bin` of the directory in which we created the Plain Agent:

```
C:\BEAHOME\user_projects\PlainAgent\bin
```

3. Enter `startAgent` at the prompt.

As the Plain Agent starts, status messages are displayed in the Command Prompt window. After the start sequence is complete, the following information message is displayed in the window:

```
<Mar 16, 2008 10:05:18 PM> <Info> <ServiceInspector> <All internal systems  
are now RUNNING.>
```

Step 2: Start the Controller

To start the Controller:

1. Open a Command Prompt window.
2. Navigate to \bin of the directory in which we created the Controller:

```
C:\BEAHOME\user_projects\controller\bin
```

3. Enter `startController` at the prompt.

As the Controller starts, status messages are displayed in the Command Prompt window. After the start sequence is complete and the Controller establishes the connection with the Plain Agent, the following information messages are displayed in the window:

```
<Mar 16, 2008 10:08:25 PM> <Info> <ServiceInspector> <All internal systems are
now RUNNING.>
STREAM: found system propertyweblogic.xml.stax.XMLStreamOutputFactory
<Mar 16, 2008 10:08:27 PM EDT> <Info> <OSGiLogReaderAdapter> <BEA-000000>
<Bundle[141] com.bea.arc.u
i, Message (ServiceEvent.REGISTERED
{objectClass=[com.bea.arc.ui.UserInterfaceService] , service.id=
123} ), Exception (null), Time (1205719707093)>
<Mar 16, 2008 10:08:27 PM EDT> <Info> <com.bea.core.tomcat.Activator>
<BEA-000000> <SimpleBundle: Se
rvice of type com.bea.arc.ui.UserInterfaceService registered.>
<Mar 16, 2008 10:08:28 PM EDT> <Info> <Configuration> <BEA-2013535>
<Successfully wrote file C:/BEAH
OME/user_projects/controller/agent-configuration-cache/PlainAgent-log-agent-co
nfig.xml.>
<Mar 16, 2008 10:08:28 PM EDT> <Info> <Configuration> <BEA-2013535>
<Successfully wrote file C:/BEAH
OME/user_projects/controller/agent-configuration-cache/PlainAgent-log-agent-co
nfig.xml.>
<Mar 16, 2008 10:08:28 PM EDT> <Info> <ResourceBrokerCommon> <BEA-2012151>
<Established connection w
ith WLOC Agent running at http://myhost.bea.com:8001.>
[AGENTS-HANDLING] : Getting the observer service for :
http://myhost.bea.com:8001/AgentLumper
rObserver
[AGENTS-HANDLING] : ObserverService URL =
http://myhost.bea.com:8001/AgentLumperObserver
<Mar 16, 2008 10:08:28 PM EDT> <Info> <ResourceBrokerPool> <BEA-2012203>
<ResourceBroker agent with
id plain-resource-pool - Registered>
```

Step 3: Start the WLOC Administration Console

To start the WLOC Administration Console:

1. Open a Web Browser.
2. Enter the following URL:

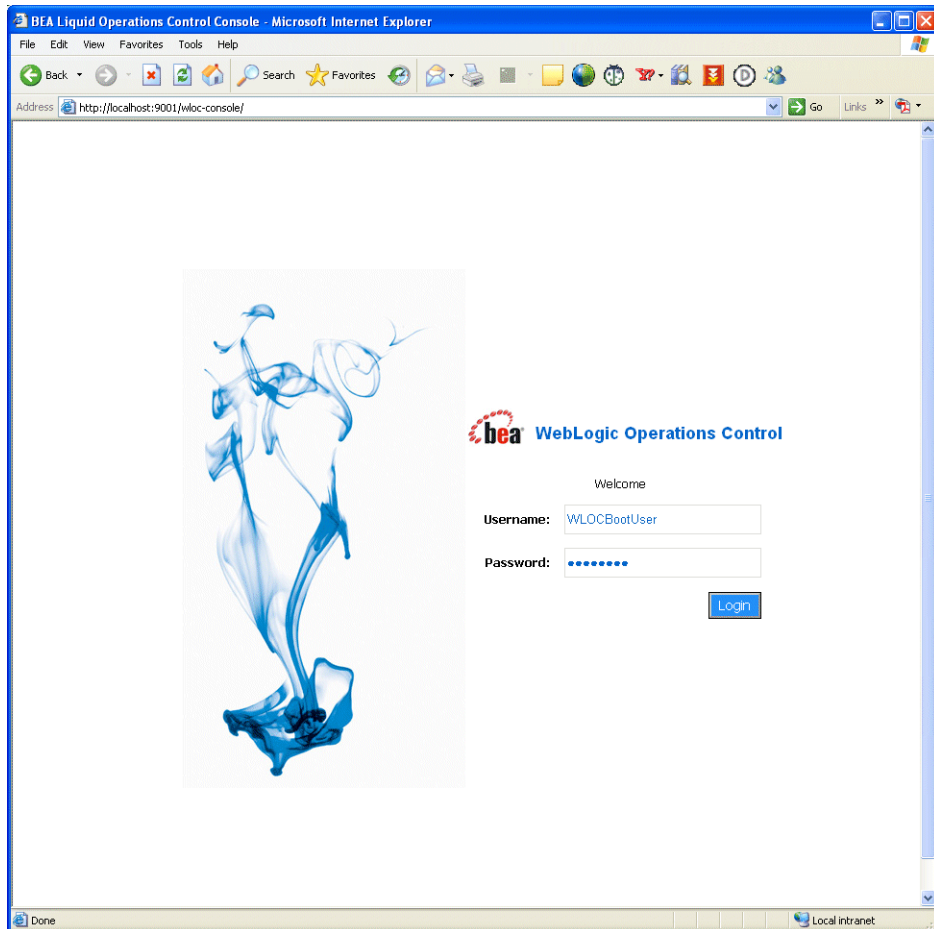
`http://localhost:9001/wloc-console`

Note: Because we are running the Controller on the local machine, we can use `localhost`. If the Controller was installed on a remote machine, you need to specify the host name for the machine hosting the Controller.

3. In the Console Welcome window, enter the user name and password required to access the Controller. Because we accepted the defaults when we created the Controller, we enter the following values in this example:

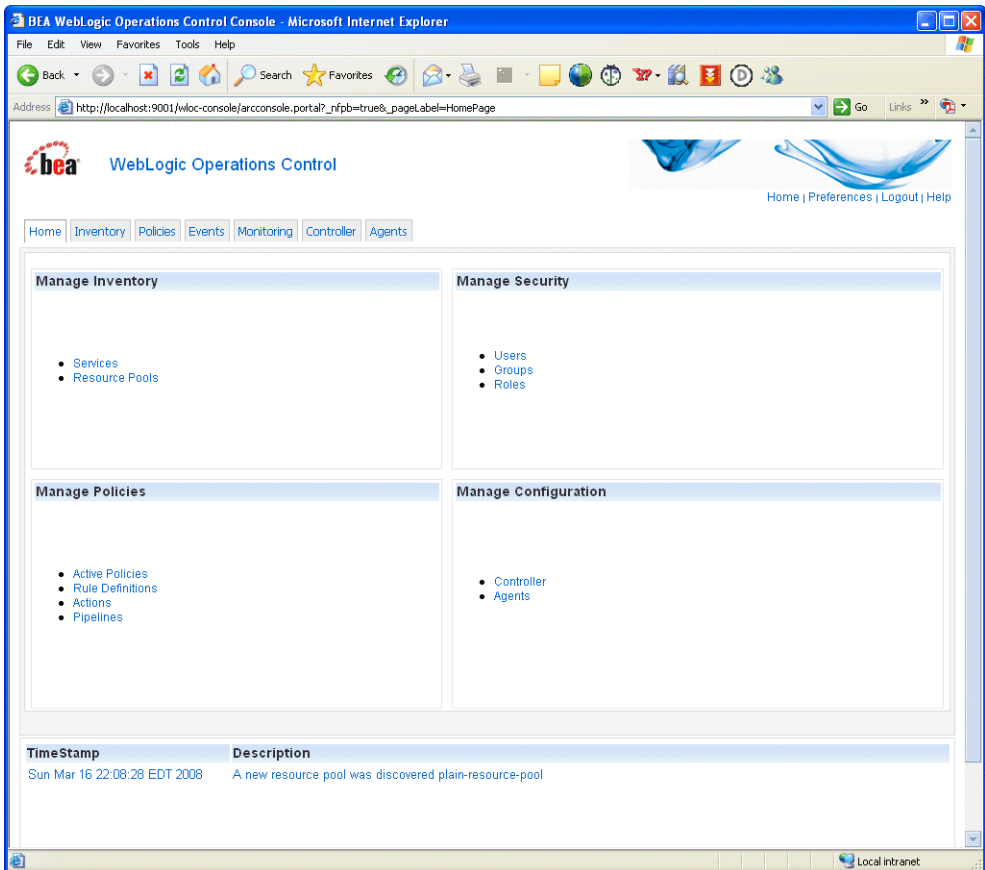
In this field	Enter the following
Username	WLOCBootUser
Password	changeit

Establish the WLOC Runtime Environment



After logging in successfully, the Home page of the WLOC Administration Console is displayed.

Step 3: Start the WLOC Administration Console



Note that the Event Viewer at the bottom of the Console indicates that the resource pool, named plain-resource-pool, that we defined when we created the Plain Agent was discovered.

What's Next?

After establishing the runtime environment, we need to define the service to be managed. For the steps in this process, go to [Chapter 4, "Define the Service Under Management."](#)

Establish the WLOC Runtime Environment

Define the Service Under Management

The next task in the use case example is to define the service to be managed, and to create and assign deployment and runtime policies that ensure the application deploys and performs as required.

A service is a grouping of processes, and a process type is a sub-group of processes within a service. (A process type is referred to as a process group in the console.) The purpose of a service is to group a collection of processes that work together. The purpose of the process group is to define instances within the service that perform the same function and can be treated as equivalent when an action is taken.

For example, you might have a two tier application with a Web app in one tier and business logic in the other. If the instances of each tier are homogeneous, then each tier could be organized as a process group and the two process groups together could comprise a service.

The distinction between these two groupings becomes important when defining policies. Generally speaking, policies related to deployment are applied across the whole service whereas runtime policies are applied to a specific process group. Furthermore, process type actions, by default, will generally pick one member of the group to act on. For example if an action is to stop an instance, the instance that gets stopped is typically not be specified and the instance is chosen by the controller.

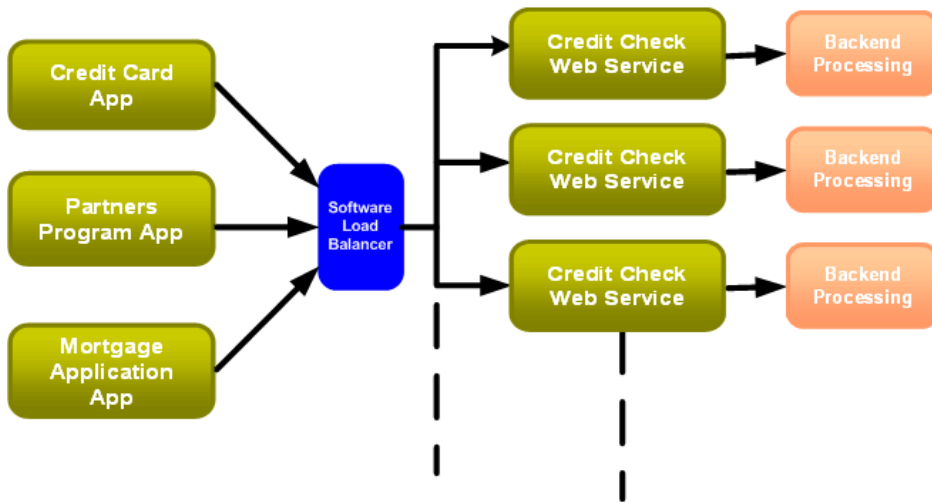
You configure services using the WLOC Administration Console, on the Inventory > Services page. A service's configuration is persisted in an XML file named `metadata-config.xml`. By default, this service metadata configuration file is located in the `BEA_HOME/user_projects/controller/config` directory. It is possible to configure a service by modifying this service metadata configuration file. Note, however, that if you

Define the Service Under Management

configure a service by modifying its metadata configuration file, you do not receive the benefits of validation and error checking that you get if you configure a service using the Administration Console.

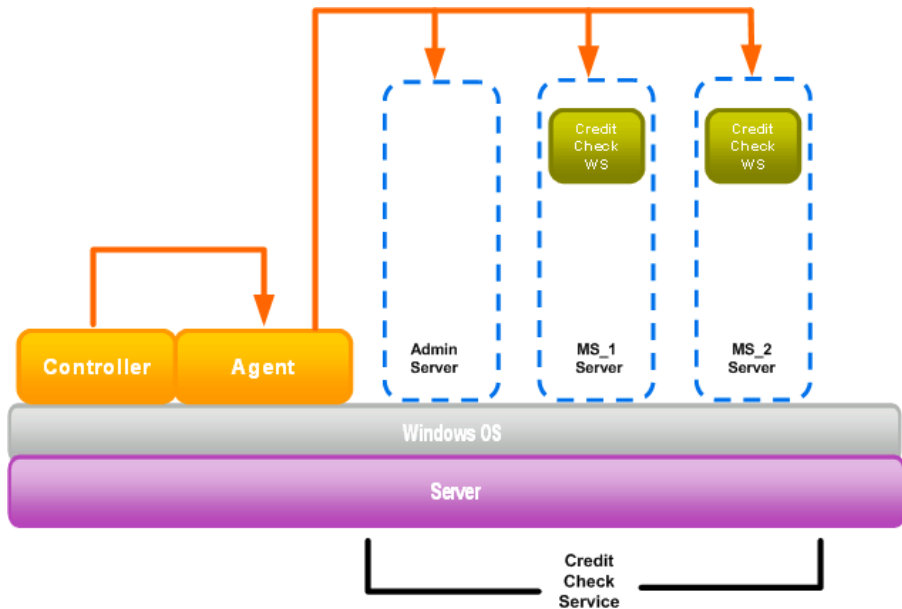
Figure 4-1 illustrates an example of a SOA application that consists of multiple client applications calling into a backend Web Service via a software load balancer. The back end Web Service can be hosted on WLS Managed Servers.

Figure 4-1 Sample SOA Application Managed by WLOC



In this use case example, we will use the WLOC Administration Console to create a service named `CreditCheckService` that will manage the backend Managed Servers, and an Administration Server. The Managed Servers host the Credit Check Web Services as shown in Figure 4-2. We will create two process groups: an Administration Server process group and a Managed Server process group. We will specify a process requirement for each of the process groups that requires a minimum of one Admin Server and one Managed Server be started before the service is deployed. We will also define a runtime policy that executes only after the service is deployed, and will start the second Managed Server if the rule definition (constraint) is violated. Figure 4-2 illustrates the topology in this example. Note that we are using a Windows environment in this example, but other platforms are supported also. For a list of supported platforms, see *BEA WebLogic Operations Control Supported Configurations*.

Figure 4-2 Use Case Example Topology



The tasks in this topic include:

- [Step 1: Create the Service and Process Groups](#)
- [Step 2: Define the Adaptive Runtime Policies](#)

Step 1: Create the Service and Process Groups

The first step in defining the service metadata is to define the general properties for the service, and then define the process groups that it will contain.

Our service will have two process groups, one that consists of a WebLogic Server Administration Server instance and one that consists of two Managed Server instances. In the WebLogic Server domain, these instances are named AdminServer, MS_1, and MS_2.

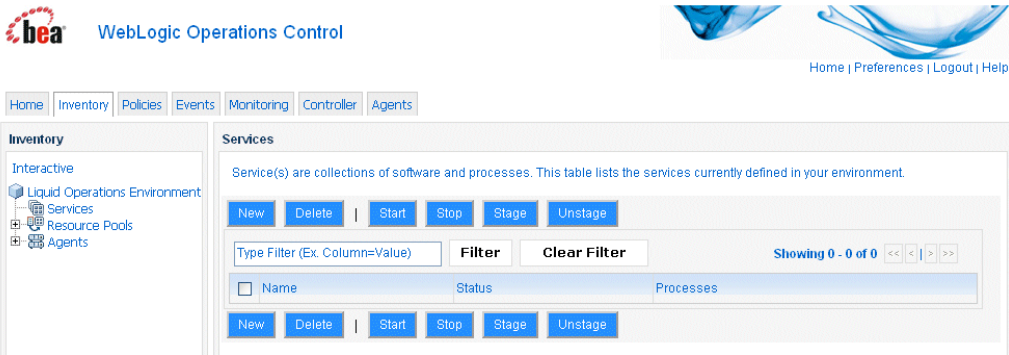
Note: Process groups are referred to as process types in the service metadata XML file.

To define the service and process groups:

1. Click the **Inventory** tab in the WLOC navigation bar and click **Services**.

Because we have not defined any services yet, there are no services listed in the table.

Define the Service Under Management



2. Click **New** to display the **Service Properties** page.
3. Enter the values shown in the following table for the general service properties.

Table 4-1 General Service Properties

In this field . . .	Specify the following information . . .
Name	CreditCheckService
Description:	Credit Check Service
Retry Count:	10 (the default)
Priority:	0 (the default)
Placement Algorithm:	Prefer resource pools with the most resources

Services

Back Next Finish Cancel

Service Properties

Specify the general properties of the service

What would you like to name your new service?

Name:

Description:

How many times would you like LOC to retry deploying this service?

Retry Count:

What priority should this service get relative to other related services?

Priority:

What preference should LOC use when selecting resource pools to place the service in?

Placement Algorithm:

Back Next Finish Cancel

4. Click **Next**.

The **Process Requirements** page is displayed. We use this page as the starting point to add the process groups in the service.

Define the Administration Server Process Group

The following steps describe how to specify the properties and define a ready metric for the Administration Server process group.

1. In the **Process Requirements** page, click **Add Process Group** to add the first process group for this service.

Define the Service Under Management

The screenshot shows a web interface titled "Services". At the top, there are navigation buttons: "Back", "Next", "Finish", "Cancel", "Add Process Group", "Remove Process Group", and "Add Resource Requirements". Below this is a section titled "Process Requirements" with the instruction: "Specify the process requirements for your service. Once you have specified the processes you may optionally define any resource requirements for a process group." Below the instruction is a table with two columns: "Process Group" and "Number of Processes". At the bottom of this section, there are the same navigation buttons as at the top.

2. Select **Start from Scratch** from the **Process Requirements** drop down menu and click **Next**.

The screenshot shows a web interface titled "Services". At the top, there are navigation buttons: "Back", "Next", "Finish", and "Cancel". Below this is a section titled "Add Processes" with the instruction: "Specify how you would like to begin adding processes". Below the instruction is a question: "How would you like to begin specifying service process requirements?". Below the question is a dropdown menu labeled "Process Requirements:". The dropdown menu is open, showing four options: "Start from Scratch", "Import from another Service", "Import from a Running WebLogic Domain", and "Import from a WebLogic Domain Configuration". At the bottom of this section, there are the same navigation buttons as at the top.

By selecting the **Start from Scratch** option, we are prompted on subsequent pages to supply properties for the process group, JVM parameters that are used when adding instances to the process group, and ready metrics that specify when an instance is available as a resource.

3. In the **Start from Scratch** page, we will first define the AdminServer process group. Enter the following values for the **Process Group Properties**:
 - **Process Group:** AdminServer
 - **Number of processes:** 1

The screenshot shows a 'Services' wizard window. At the top, there are buttons for 'Back', 'Next', 'Finish', and 'Cancel'. Below these is the 'Start from Scratch' section, which includes instructions to specify properties for a process group. The 'Process Group Properties' section is expanded, showing a question: 'What would you like to name your new process group?'. Below this, the 'Process Group' field is filled with 'AdminServer'. Another question asks 'How many processes would you like to initially create?', and the 'Number of Processes' field is filled with '1'.

4. In the **Process Group Template Properties** portion of the window, enter the values shown in [Table 4-2](#) for the AdminServer process group instance.

Table 4-2 AdminServer Process Group Template Properties

In this field . . .	Specify the following information . . .
Name	AdminServer
Description	AdminServer JVM
Main Class	Leave this option selected. Because our service is deployed on WLS, we need to invoke the <code>weblogic.Server</code> main class.
Main JAR	Leave this option unselected.
Host:	The name of the host machine. In this example, we specify localhost. The host and port number are used to determine the address the Agent uses to collect JMX metric information from the endpoint.
Starting Port#	7001
Classpath	The CLASSPATH for the domain.

Table 4-2 AdminServer Process Group Template Properties

In this field . . .	Specify the following information . . .
JVM Arguments	<p>The JVM arguments to be used when the process starts. In this example, we enter the arguments that are appended to the <code>weblogic.Server</code> start command to specify minimum and maximum heap sizes, and to set the WebLogic Server instance name, security credentials, security policy, home and domain directories:</p> <pre data-bbox="460 578 1096 855">-Xmx128m -Xms64m -da -Dwls.home=D:\wls_92\weblogic92\server -Dweblogic.management.discover=true -Dweblogic.Name=AdminServer -Dweblogic.management.username=weblogic -Dweblogic.management.password=weblogic -Djava.security.policy=D:\wls_92\weblogic92\server\lib\weblogic.policy -Dweblogic.RootDirectory=D:\wls_92\user_projects\domains\LOC_base_domain</pre>
Java Arguments	<p>In this example, no Java arguments are required.</p>
UserName	<p>The username required for authenticating JMX connections.</p> <p>In this example, we specify <code>weblogic</code> as both the user name and the password because those are the values required to authenticate to the Administration Server.</p>
Password	<p>The password required for authenticating JMX connections.</p> <p>Enter <code>weblogic</code>.</p>
Instance Directory	<p>Leave this field blank.</p>
Native Lib Directory	<p>Leave this field blank.</p>
Use Native JMX	<p>Leave this unchecked.</p>
SSH Enabled	<p>Leave this unchecked.</p>
Protocol	<p>Leave <code>iiop</code> selected.</p>

Step 1: Create the Service and Process Groups

Process Group Template Properties

The following parameters will be used as a starting point for creating processes within this Process Group.

Name:

Description:

Main Class:

Main JAR:

Host:

Starting Port#:

Classpath:

JVM Arguments:

Java Arguments:

UserName:

Password:

Instance Directory:

Native Lib Directory:

Use Native JMX:

SSH Enabled:

Determines whether SSH will be enabled for this instances.

Protocol:

Define the Service Under Management

5. Specify a ready metric for the AdminServer instance by entering the values shown in [Table 4-3](#).

A ready metric indicates when a process has been started and is ready for work. For a WebLogic Server instance, the `ServerRuntime` MBean has a `State` attribute. When `ServerRuntimeMBean.State=RUNNING`, the WebLogic Server instance is ready.

Table 4-3 AdminServer Ready Metrics

In this field . . .	Specify the following information . . .
Instance Name	<code>com.bea:Name=AdminServer,Type=ServerRuntime</code>
Attribute	<code>State</code>
Value	<code>RUNNING</code>
Operator	Value Equals (the default)
Value Type	String
Wait	300000 (This value ensures the WLS Admin Server instance has time to complete its startup.)

Ready Metric

Instance Name:

Attribute:

Value:

Operator:

Value Type:

Wait:

Back
Next
Finish
Cancel

6. Click **Next**.

Note that the AdminServer process group is listed in the **Process Group** table.

Services

Back
Next
Finish
Cancel
Add Process Group
Remove Process Group
Add Resource Requirements

Process Requirements

Specify the process requirements for your service. Once you have specified the processes you may optionally define any resource requirements for a process group.

	Process Group	Number of Processes
<input type="checkbox"/>	AdminServer	1

Back
Next
Finish
Cancel
Add Process Group
Remove Process Group
Add Resource Requirements

In the next steps we will define the Managed Server process group.

Define the Managed Servers Process Group

The following steps describe how to specify the properties and ready metrics for both Managed Server instances.

Define the Service Under Management

1. In the **Process Requirements** page, click **Add Process Group** to add the Managed Server process group.
2. Select **Start from Scratch** from the **Process Requirements** drop down menu and click **Next**.
3. In the **Start from Scratch** page, we will first define the Managed Servers process group.
 - a. Name the process group ManagedServers
 - b. Because we have two Managed Servers in this group, enter 2 in the **Number of Processes** field.
4. Define the Managed Server processes by entering the values shown in [Table 4-4](#) for the **Process Group Template** properties.

Note that the values that are populated in the template are obtained from the values we provided for the AdminServer process group. We modify them as appropriate for the ManagedServers group.

The values that we specify on this page are duplicated for all the instances in the group. Therefore, in this example, both of the Managed Server instances will initially contain the same values.

Table 4-4 ManagedServers Process Group Template Properties

In this field . . .	Specify the following information . . .
Name	MS_1. WLOC automatically appends 0 to the first process instance name. For each additional process instance that is configured, a numeric suffix is added to the name starting with 1 and incrementing by 1 for each additional process instance. Because we are defining two ManagedServers, the first instance is automatically named MS_10 and the second instance is named MS_11.
Description	MS_1
Main Class	Leave this option selected. Because our service is managing WLS Admin and Managed Servers, we need to invoke the <code>weblogic.Server</code> main class.
Main JAR	Leave this option unselected.
Host:	localhost The host and port number are used to determine the address the Agent uses to collect JMX metric information from the endpoint.

Table 4-4 ManagedServers Process Group Template Properties

In this field . . .	Specify the following information . . .
Starting Port#	7002
Classpath	The CLASSPATH for the domain. This field is prepopulated with the CLASSPATH that we entered for the AdminServer process group.
JVM Arguments	<p>The JVM arguments to be used when the process starts. This field is prepopulated with the arguments that we entered for the AdminServer process group. We need to modify them as follows for the MS_1 Managed Server:</p> <ol style="list-style-type: none"> 1. Add the following argument required to start Managed Servers: <code>-Dweblogic.management.server=http://localhost:7001</code> 2. Change the name of the server: <code>-Dweblogic.Name=MS_1</code> <p>The remaining values apply to both the AdminServer and the Managed Servers.</p>
Java Arguments	In this example, we do not need to specify any Java arguments.
UserName	<p>The username required for authenticating JMX connections.</p> <p>We use <code>weblogic</code> in this example.</p>
Password	<p>The password required for authenticating JMX connections.</p> <p>We use <code>weblogic</code> in this example.</p>
Instance Directory	Leave this field blank.
Native Lib Directory	Leave this field blank.
Use Native JMX	Leave this unchecked.
SSH Enabled	Leave this unchecked.
Protocol	Leave <code>iiop</code> selected.

5. Define the ready metric for the MS_1 process instance as follows.

Define the Service Under Management

Table 4-5 ManagedServers Ready Metrics

In this field . . .	Specify the following information . . .
Instance Name	com.bea:Name=MS_1,Type=ServerRuntime
Attribute	State
Value	RUNNING
Operator	Value Equals (the default)
Value Type	String
Wait	300000

6. Click **Next**.

The **Process Requirements** page is displayed listing both the AdminServer and ManagedServers process groups in the table.

Process Group	Number of Processes
<input type="checkbox"/> AdminServer	1
<input type="checkbox"/> ManagedServers	2

We now need to modify the properties for the second Managed Server instance, MS_2.

7. Select the ManagedServers name in the **Process Group** table. The list of defined processes in the process group is displayed. Note that the second ManagedServer instance created is named MS_11 and the port is automatically incremented to 7003.

Step 1: Create the Service and Process Groups

Services

Back Next Finish Cancel Add JVM Remove JVM Move JVM

Process Requirements > Process Group Processes

This page allows you to edit the processes you've specified for the service.

	Name	Host	Port	Priority
<input type="checkbox"/>	MS_10	localhost	7002	0
<input type="checkbox"/>	MS_11	localhost	7003	0

Back Next Finish Cancel Add JVM Remove JVM Move JVM

8. Select MS_11 in the **Name** column. The properties for the process are displayed.

Services

Save Cancel

Process Properties

Specify the properties for the selected process.

JVM Template Properties

What would you like to name the process?

Name:

Description:

Main Class:

Main JAR:

Host:

What priority should be given this process in starting relative to other processes?

Priority:

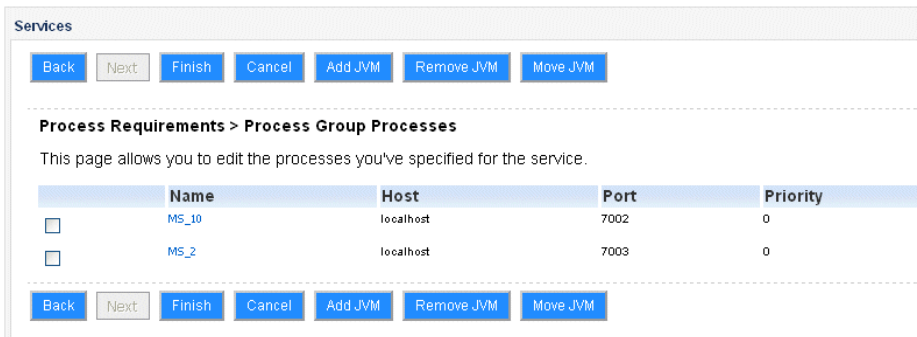
What port number would you like to start at?

Starting Port#:

Classpath:

Define the Service Under Management

- In the **Name** field, change MS_11 to MS_2.
- In the **Description** field, change MS_1 to MS_2.
- In the **JVM Args** field, change the argument `-Dweblogic.Name=MS_1` to `-Dweblogic.Name=MS_2`.
- In the **Ready Metric** section, change the **Instance Name** field from `com.bea:Name=MS_1,Type=ServerRuntime` to `com.bea:Name=MS_2,Type=ServerRuntime`.
- Click **Save**. The updated process is shown in the table.



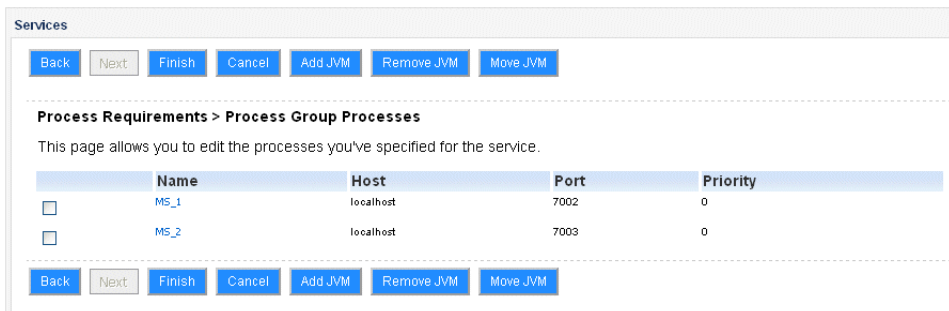
The screenshot shows the 'Services' console interface. At the top, there are navigation buttons: Back, Next, Finish, Cancel, Add JVM, Remove JVM, and Move JVM. Below this is the section 'Process Requirements > Process Group Processes'. A text box states: 'This page allows you to edit the processes you've specified for the service.' Below the text is a table with the following data:

	Name	Host	Port	Priority
<input type="checkbox"/>	MS_10	localhost	7002	0
<input type="checkbox"/>	MS_2	localhost	7003	0

At the bottom of the table, there are navigation buttons: Back, Next, Finish, Cancel, Add JVM, Remove JVM, and Move JVM.

To avoid confusion, we will also change the name of the first Managed Server from MS_10 to MS_1 to match the name of the Managed Server in the WLS domain.

- Select MS_10 in the **Name** column. The process properties are displayed.
- In the **Name** field, change MS_10 to MS_1 and click **Save**. We now have two processes in the MangedServers process group named MS_1 and MS_2.



The screenshot shows the 'Services' console interface. At the top, there are navigation buttons: Back, Next, Finish, Cancel, Add JVM, Remove JVM, and Move JVM. Below this is the section 'Process Requirements > Process Group Processes'. A text box states: 'This page allows you to edit the processes you've specified for the service.' Below the text is a table with the following data:

	Name	Host	Port	Priority
<input type="checkbox"/>	MS_1	localhost	7002	0
<input type="checkbox"/>	MS_2	localhost	7003	0

At the bottom of the table, there are navigation buttons: Back, Next, Finish, Cancel, Add JVM, Remove JVM, and Move JVM.

16. Click Finish.

17. Define the resource requirements for the process groups as described in the following section.

Define Resource Requirements for the Service

Before we finish creating the service, we need to define the resource requirements for each process group. When you define a resource requirement, you are creating a resource agreement. *Resource agreements* define requirements that are evaluated before starting a service or instance, which can occur both at deployment and runtime.

To define the resource requirements for the process groups in the CreditCheckService, we complete the following steps.

1. Select the **AdminServer** process group check box and click **Add Resource Requirements**.

The screenshot shows the 'Services' configuration window. At the top, there are navigation buttons: Back, Next, Finish, Cancel, Add Process Group, Remove Process Group, and Add Resource Requirements. Below this is the 'Process Requirements' section, which includes a descriptive text: 'Specify the process requirements for your service. Once you have specified the processes you may optionally define any resource requirements for a process group.' Below the text is a table with two columns: 'Process Group' and 'Number of Processes'. The table has two rows: 'AdminServer' with a checked checkbox and a value of 1, and 'ManagedServers' with an unchecked checkbox and a value of 2. At the bottom of the window, there are another set of navigation buttons: Back, Next, Finish, Cancel, Add Process Group, Remove Process Group, and Add Resource Requirements.

Process Group	Number of Processes
<input checked="" type="checkbox"/> AdminServer	1
<input type="checkbox"/> ManagedServers	2

2. In the **Minimum # of Processes** field, enter 1 and click **Save**.

This value specifies the number of instances that will be started when the service is deployed. It will also generate a policy that ensures that the minimum number specified is maintained while the service is running.

We do not need to specify any other requirements for the AdminServer process group for this example.

Define the Service Under Management

Services

Save Cancel

Process Requirements > Resource Agreement

This page allows you to optionally specify any resource requirements for your service and its processes

Process Requirements

What is the minimum number of processes that a required for the process group?

Minimum # of Processes:

What is the maximum number of processes that should be started for the process group?

Maximum # of Processes:

3. Repeat steps 1 and 2 for the **ManagedServers** process group.
4. Click **Finish** to create the service.

The CreditCheckService is now shown in the **Services** table and the **Task and Events** Viewer at the bottom of the Console indicates that the CreditCheckService was added.

Step 1: Create the Service and Process Groups

New service created successfully

Services

Service(s) are collections of software and processes. This table lists the services currently defined in your environment.

[New](#) [Delete](#) | [Start](#) [Stop](#) [Stage](#) [Unstage](#)

Type Filter (Ex. Column=Value) [Filter](#) [Clear Filter](#) **Showing 1 - 1 of 1** << < | > >>

<input type="checkbox"/>	Name	Status	Processes
<input type="checkbox"/>	CreditCheckService	undeployed	0

[New](#) [Delete](#) | [Start](#) [Stop](#) [Stage](#) [Unstage](#)

Description

A new service was added CreditCheckService
A new resource pool was discovered plain-resource-pool

View Deployment Policy

When you specify the resource requirements for the service, the deployment policy for the service is created automatically. A policy consists of a constraint and an action to take when the constraint is violated. In this example, we set the minimum number of processes to 1 which indicates that there must be one running instance in both the AdminServer and ManagedServers process groups. This constraint is automatically bound to the StartJavaInstanceAction

To view the deployment policy, select the **Policies** tab.

Define the Service Under Management

Active Policies

Policies

Definitions

Policies define a Quality of Service for a Service and its Processes. A policy consists of a rule and an action or pipeline. This table lists the policies currently defined in your environment.

Assign Policy Unassign Policy

<input type="checkbox"/>	Service	Process	Policy
<input type="checkbox"/>	CreditCheckService	AdminServer	CreditCheckService-AdminServer-minprocess If Number.Processes is < 1 then run action CreditCheckServicestartaction
<input type="checkbox"/>		All	CreditCheckServicedeploy-key If Service.DeploymentState is starting then run action CreditCheckServicestartserviceaction
<input type="checkbox"/>		All	CreditCheckServiceundeploy-key If Service.DeploymentState is stopping then run action CreditCheckServicestopserviceaction
<input type="checkbox"/>		ManagedServers	CreditCheckService-ManagedServers-minprocess If Number.Processes is < 1 then run action CreditCheckServicestartaction

Assign Policy Unassign Policy

Create Services Using Helper Methods

In this example, we created the service using the **Start from Scratch** option to demonstrate the complete method for defining process requirements. However, the WLOC Administration Console provides efficient helper methods that simplify this process by importing the information from the WLS domain. These helper methods include:

- Import from another Service
- Import from a Running WebLogic Domain
- Import from a WebLogic Domain Configuration

If you had selected one of these options in [step 5](#) above, much of the information that we provided manually could have been captured from the `config.xml` file for the domain.

CreditCheckService Service Metadata Configuration

When you create the service, the associated constraints, actions, and bindings are created automatically. [Listing 4-1](#) shows how the `CreditCheckService` service configuration is represented in the `metadata-config.xml` file.

Listing 4-1 CreditCheckService Metadata Configuration

```
<ns2:services>
  <ns2:service>
    <ns2:name>CreditCheckService</ns2:name>
```

Step 1: Create the Service and Process Groups

```
<ns2:description>Credit Check Service</ns2:description>
<ns2:state>undeployed</ns2:state>
<ns2:priority>0</ns2:priority>
<ns2:constraint-bindings>
  <ns2:constraint-binding>

<ns2:constraint-key>CreditCheckServiceDeploy-key</ns2:constraint-key>

<ns2:action-key>CreditCheckServiceStartServiceAction</ns2:action-key>
  </ns2:constraint-binding>
  <ns2:constraint-binding>

<ns2:constraint-key>CreditCheckServiceUndeploy-key</ns2:constraint-key>

<ns2:action-key>CreditCheckServiceStopServiceAction</ns2:action-key>
  </ns2:constraint-binding>
</ns2:constraint-bindings>
<ns2:process-types>
  <ns2:process-type>
    <ns2:constraint-bindings>
      <ns2:constraint-binding>

<ns2:constraint-key>CreditCheckService-ManagedServers-minprocess</ns2:constraint-key>

<ns2:action-key>CreditCheckServiceStartAction</ns2:action-key>
  </ns2:constraint-binding>
  </ns2:constraint-bindings>
  <ns2:name>ManagedServers</ns2:name>
  <ns2:description>ManagedServers-description</ns2:description>

<ns2:metadata-key>CreditCheckService-ManagedServersmetakey</ns2:metadata-key>
  </ns2:process-type>
  <ns2:process-type>
    <ns2:constraint-bindings>
      <ns2:constraint-binding>

<ns2:constraint-key>CreditCheckService-AdminServer-minprocess</ns2:constraint-key>

<ns2:action-key>CreditCheckServiceStartAction</ns2:action-key>
  </ns2:constraint-binding>
  </ns2:constraint-bindings>
  <ns2:name>AdminServer</ns2:name>
  <ns2:description>AdminServer-description</ns2:description>

<ns2:metadata-key>CreditCheckService-AdminServermetakey</ns2:metadata-key>
  </ns2:process-type>
</ns2:process-types>
```

Define the Service Under Management

```
<ns2:max-failed-event-retry-count>10</ns2:max-failed-event-retry-count>
  </ns2:service>
</ns2:services>
<ns2:connection-factories/>
<ns2:connection-infos/>
<ns2:constraints>
  <ns2:deployment-state-constraint>
    <ns2:name>CreditCheckService-service-deploy</ns2:name>
    <ns2:key>CreditCheckServicedeploy-key</ns2:key>
    <ns2:priority>0</ns2:priority>
    <ns2:state>starting</ns2:state>
    <ns2:evaluation-period>0</ns2:evaluation-period>
  </ns2:deployment-state-constraint>
  <ns2:deployment-state-constraint>
    <ns2:name>CreditCheckService-service-undeploy</ns2:name>
    <ns2:key>CreditCheckServiceundeploy-key</ns2:key>
    <ns2:priority>0</ns2:priority>
    <ns2:state>stopping</ns2:state>
    <ns2:evaluation-period>0</ns2:evaluation-period>
  </ns2:deployment-state-constraint>
  <ns2:min-process-constraint>
    <ns2:name>CreditCheckService-AdminServer-minprocess</ns2:name>
    <ns2:key>CreditCheckService-AdminServer-minprocess</ns2:key>
    <ns2:priority>0</ns2:priority>
    <ns2:state>deployed</ns2:state>
    <ns2:evaluation-period>0</ns2:evaluation-period>
    <ns2:value>1</ns2:value>
  </ns2:min-process-constraint>
  <ns2:min-process-constraint>
    <ns2:name>CreditCheckService-ManagedServers-minprocess</ns2:name>
    <ns2:key>CreditCheckService-ManagedServers-minprocess</ns2:key>
    <ns2:priority>0</ns2:priority>
    <ns2:state>deployed</ns2:state>
    <ns2:evaluation-period>0</ns2:evaluation-period>
    <ns2:value>1</ns2:value>
  </ns2:min-process-constraint>
</ns2:constraints>
<ns2:notifications/>
<ns2:pipelines/>
<ns2:actions>
  <ns2:action>
    <ns2:name>CreditCheckServicestartserviceaction</ns2:name>
    <ns2:key>CreditCheckServicestartserviceaction</ns2:key>

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

```

</ns2:action>
<ns2:action>
  <ns2:name>CreditCheckServicestopserviceaction</ns2:name>
  <ns2:key>CreditCheckServicestopserviceaction</ns2:key>

<ns2:impl-class>com.bea.adaptive.actions.internal.StopServiceAction</ns2:impl-
class>
  <ns2:adjudicate>>false</ns2:adjudicate>
  <ns2:properties/>
</ns2:action>
<ns2:action>
  <ns2:name>CreditCheckServicestartaction</ns2:name>
  <ns2:key>CreditCheckServicestartaction</ns2:key>

<ns2:impl-class>com.bea.adaptive.actions.internal.StartJavaInstanceAction</ns2
:impl-class>
  <ns2:adjudicate>>false</ns2:adjudicate>
  <ns2:properties/>
</ns2:action>
<ns2:action>
  <ns2:name>CreditCheckService-ManagedServers-defaultaction</ns2:name>
  <ns2:key>CreditCheckService-ManagedServers-defaultaction</ns2:key>

<ns2:impl-class>com.bea.arc.ui.actions.ConsoleNotificationAction</ns2:impl-cla
ss>
  <ns2:adjudicate>>false</ns2:adjudicate>
  <ns2:properties/>
</ns2:action>
<ns2:action>
  <ns2:name>CreditCheckService-AdminServer-defaultaction</ns2:name>
  <ns2:key>CreditCheckService-AdminServer-defaultaction</ns2:key>

<ns2:impl-class>com.bea.arc.ui.actions.ConsoleNotificationAction</ns2:impl-cla
ss>
  <ns2:adjudicate>>false</ns2:adjudicate>
  <ns2:properties/>
</ns2:action>
</ns2:actions>

```

AdminServer Process Group Metadata Configuration

[Listing 4-2](#) shows how the AdminServer process group configuration is represented in the metadata-config.xml file. The ready metric configuration is shown in bold type.

Define the Service Under Management

Listing 4-2 AdminServer Process Group Metadata Configuration

```
<ns2:metadata-group>
  <ns2:name>CreditCheckService-AdminServermetal</ns2:name>
  <ns2:key>CreditCheckService-AdminServermetakey</ns2:key>
  <ns2:instances>
    <ns2:jvm-instance>
      <ns2:name>AdminServer</ns2:name>
      <ns2:description>AdminServerJVM</ns2:description>
      <ns2:main-class>weblogic.Server</ns2:main-class>
      <ns2:ready-information>
        <ns2:check-type>ValueEquals</ns2:check-type>
        <ns2:max-wait-period>300000</ns2:max-wait-period>
      </ns2:ready-information>
    </ns2:jvm-instance>
  </ns2:instances>
  <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>
  <ns2:jvm-args>
    <ns2:arg>-Xmx128m</ns2:arg>
    <ns2:arg>-Xms64m</ns2:arg>
    <ns2:arg>-da</ns2:arg>
    <ns2:arg>-Dwls.home=D:\wls_92\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:jvm-args>
  <ns2:arg>-Djava.security.policy=D:\wls_92\weblogic92\server\lib\weblogic.policy</ns2:arg>
  <ns2:arg>-Dweblogic.RootDirectory=D:\wls_92\user_projects\domains\LOC_base_domain</ns2:arg>
  <ns2:arg>-cp</ns2:arg>
  <ns2:arg>D:\wls_92\patch_weblogic921\profiles\default\sys_manifest_classpath\weblogic_patch.jar;
  D:\wls_92\JDK150~1\lib\tools.jar;D:\wls_92\WEBLOG~1\server\lib\weblogic_sp.jar;
  ;
  D:\wls_92\WEBLOG~1\server\lib\weblogic.jar;D:\wls_92\WEBLOG~1\server\lib\webse
  rvices.jar;
  D:\wls_92\WEBLOG~1\common\eval\pointbase\lib\pbclient51.jar;D:\wls_92\WEBLOG~1
  \server\lib\xqrl.jar;</ns2:arg>
  </ns2:jvm-args>
  <ns2:java-args/>
  <ns2:native-lib-dir></ns2:native-lib-dir>
  <ns2:instance-dir></ns2:instance-dir>
```



```

<ns2:native-jmx>>false</ns2:native-jmx>
<ns2:protocol>iiop</ns2:protocol>
<ns2:host>localhost</ns2:host>
<ns2:port>7001</ns2:port>
<ns2:username>weblogic</ns2:username>

<ns2:password>{Salted-3DES}W0v+mTzrr9u/PRD30V1xGw==</ns2:password>
<ns2:ssh-enabled>>false</ns2:ssh-enabled>
<ns2:wait-for-ssh>>false</ns2:wait-for-ssh>
<ns2:priority>0</ns2:priority>
<ns2:copies-at-create/>
<ns2:copies-at-shutdown/>
  </ns2:jvm-instance>
</ns2:instances>
</ns2:metadata-group>

```

Step 2: Define the Adaptive Runtime Policies

Now that we have defined the service and the initial deployment policies, we need to define the runtime policies.

WLOC policies specify runtime requirements (constraints or rules) for a service and actions to take when the service operates outside of the constraints. These policies define service level agreements (SLAs) for your services. For example, you can define a policy that increases the amount of memory available if the memory requirements for a specific JVM grow beyond a specified number of MBs.

WLOC contains a set of predefined constraints, called SmartPacks, that you can use to place requirements on some common measurements of service health and performance. In this example, we will define a runtime policy for the ManagedServers process group using the MaxAverageJVMProcessorLoad constraint that is provided as part of the Smart Pack constraints.

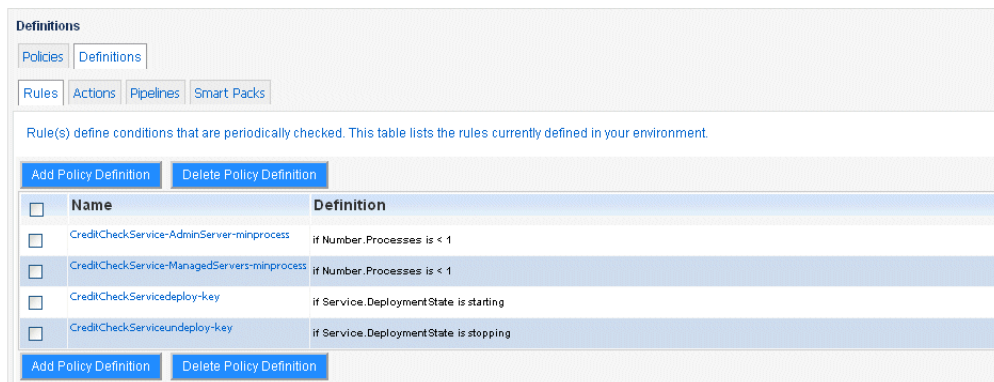
For the runtime policy, we will define a rule (constraint) in which we specify a value of 0.8 for the MaxAverageJVMProcessorLoad constraint. This value indicates that when the average JVM processor load exceeds 80%, an action must occur. In this example, a second Managed Server instance will be started.

To create the runtime policy and assign it to the ManagedServers process group:

1. Select the **Policies** tab in the WLOC navigation bar, select the **Definitions** tab, and then select the **Rules** tab.

The list of the process constraint deployment polices are shown in the table.

Define the Service Under Management



Definitions

Policies | Definitions

Rules | Actions | Pipelines | Smart Packs

Rule(s) define conditions that are periodically checked. This table lists the rules currently defined in your environment.

Add Policy Definition Delete Policy Definition

<input type="checkbox"/>	Name	Definition
<input type="checkbox"/>	CreditCheckService-AdminServer-minprocess	If Number.Processes is < 1
<input type="checkbox"/>	CreditCheckService-ManagedServers-minprocess	If Number.Processes is < 1
<input type="checkbox"/>	CreditCheckServicedeploy-key	If Service.DeploymentState is starting
<input type="checkbox"/>	CreditCheckServiceundeploy-key	If Service.DeploymentState is stopping

Add Policy Definition Delete Policy Definition

2. Click **Add Policy Definition**.
3. In the **Policy Properties** page, enter **MaxAverageJVMPprocessorLoad** in the **Name** field.
4. Select **Based on a named attribute** from the **Type** drop-down menu and click **Next**.

Definitions

Policies Definitions

Rules Actions Pipelines Smart Packs

Back Next Finish Cancel

Policy Properties

Specify the properties for the policy. You can change the rule and action or pipelines to run.

New Policy Definition

What would you like to name your new policy?

Name:

What type of policy would you like to create?

Type:

- Based on the value of an attribute or function
- Based on the value of an attribute or function
- Based on firing a service event at a date/time
- Based on firing a process group event at a date/time
- Based on deploying a service at a date/time
- Based on undeploying a service 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 a service deployment state
- Based on a minimum amount of CPU
- 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 a minimum number of processes
- Based on a maximum number of processes
- Based on software availability
- Based on an IP Address
- Based on ISO software availability
- Based on an amount of Local Disk Size required

Back Next Finish

Description

5. In the **Rule Properties** page, accept the default for the **Name** and **Priority** fields.
6. In the **Attribute** field, select **MaxAverageJVMPProcessorLoad** from the drop-down menu.

Define the Service Under Management

Definitions

Policies Definitions

Rules Actions Pipelines Smart Packs

Back Next Finish Cancel

Rule Properties

This page allows you to specify the parameters for the Named Policy definition.

New Named Policy Definition

What would you like to name your new rule definition?

Name:

What priority should instances of this rule be given over others?

Priority:

Please specify the properties of the managed object this rule applies to.

Attribute:

Value:

How often should this rule be evaluated?

Evaluation Period:

- MinFreeHeapSize
- MaxFreeHeapSize
- MinAverageFreeHeapSize
- MaxAverageFreeHeapSize
- MinFreePhysicalMemory
- MaxFreePhysicalMemory
- MinAverageFreePhysicalMemory
- MaxAverageFreePhysicalMemory
- MinUsedPhysicalMemory
- MaxUsedPhysicalMemory
- MinAverageUsedPhysicalMemory
- MaxAverageUsedPhysicalMemory
- MinJVMProcessorLoad
- MaxJVMProcessorLoad
- MinAverageJVMProcessorLoad
- MaxAverageJVMProcessorLoad

De

- Specify the remaining values as shown in the following table and click **Finish**.

In this field . . .	Do the following . . .
Value	Enter 0.8.
Evaluation Period	Enter 10000. This is the amount of time, in milliseconds, to wait before reevaluating the constraint.
Service State	Select deployed from the drop-down menu. This indicates that the service must be deployed before the constraint will be evaluated.

Definitions

Policies Definitions

Rules Actions Pipelines Smart Packs

maxAverageJvmProcessorLoad

What priority should instances of this rule be given over others?

Priority:

Please specify the properties of the managed object this rule applies to.

Attribute:

Value:

How often should this rule be evaluated? Specify interval in milliseconds.

Evaluation Period:

What state should the service be in for this rule to be applicable?

Service State

Back Next Finish Cancel

The new policy rule is added to the **Definitions** table.

Define the Service Under Management

Definitions

Policies Definitions

Rules Actions Pipelines Smart Packs

Rule(s) define conditions that are periodically checked. This table lists the rules currently defined in your environment.

Add Policy Definition Delete Policy Definition

<input type="checkbox"/>	Name	Definition
<input type="checkbox"/>	CreditCheckService-AdminServer-minprocess	if Number.Processes is < 1
<input type="checkbox"/>	CreditCheckService-ManagedServers-minprocess	if Number.Processes is < 1
<input type="checkbox"/>	CreditCheckService-deploy-key	if Service.DeploymentState is starting
<input type="checkbox"/>	CreditCheckService-undeploy-key	if Service.DeploymentState is stopping
<input type="checkbox"/>	MaxAverageJVMProcessorLoad	if MaxAverageJvmProcessorLoad > 0.8

Add Policy Definition Delete Policy Definition

8. Select the **Policies** subtab to return to the **Active Policies** page.

Active Policies

Policies Definitions

Policies define a Quality of Service for a Service and its Processes. A policy consists of a rule and an action or pipeline. This table lists the policies currently defined in your environment.

Assign Policy Unassign Policy

<input type="checkbox"/>	Service	Process	Policy
<input type="checkbox"/>	CreditCheckService	AdminServer	CreditCheckService-AdminServer-minprocess if Number.Processes is < 1 then run action CreditCheckServicestartaction
<input type="checkbox"/>		All	CreditCheckService-deploy-key if Service.DeploymentState is starting then run action CreditCheckServicestartserviceaction
<input type="checkbox"/>		All	CreditCheckService-undeploy-key if Service.DeploymentState is stopping then run action CreditCheckServicestopserviceaction
<input type="checkbox"/>		ManagedServers	CreditCheckService-ManagedServers-minprocess if Number.Processes is < 1 then run action CreditCheckServicestartaction

Assign Policy Unassign Policy

9. Click **Assign Policy** to assign the policy (the rule and the action to take) to the **ManagedServers** process group.

10. In the **Policy Properties** page, specify the properties of the policy as follows:

- In the **Service** drop-down menu, select **CreditCheckService** to assign the policy to the service.
- In the **Process** drop-down menu, select **ManagedServers** to assign the policy to the ManagedServers process group.
- In the **Instance** drop-down menu, select **All Instances for ManagedServers** to assign the policy to all the Managed Server instances in the process group.

- d. In the **Definitions** drop-down menu, select **MaxAverageJVMProcessorLoad** to assign the rule definition to the policy.
- e. In the **Run Action** drop-down menu, select **CreditCheckServicestartaction** to specify the action to take when the constraint (rule) is violated.
- f. Because we did not define an action pipeline in this example, leave **None** selected in the **Run Pipeline** drop-down menu.

The screenshot shows the 'Active Policies' configuration interface. At the top, there are tabs for 'Policies' and 'Definitions'. Below the tabs, there is a section titled 'Assign Policy' with the following fields:

- What service will this policy apply to?**
Service: CreditCheckService
- Which process in the service will this policy apply to?**
Process: ManagedServers
- Which process instance in the process will this apply to?**
Instance: All instances for ManagedServers
- Which rule definition would you like to use?**
Definitions: MaxAverageJVMProcessorLoad
- Which action would you like to run if the rule is true?**
Run Action: CreditCheckServicestartaction
- Which pipeline would you like to run if the rule is true?**
Run Pipeline: NONE

At the bottom of the form, there are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'.

11. Click **Finish** to finish assigning the policy to the process group.

The policy assignment is created and the `Binding created successfully` confirmation message is displayed.

Note that the **MaxAverageJVMProcessorLoad** policy is now assigned to the `ManagedServers` process group in the **Active Policies** table.

Define the Service Under Management

Binding created successfully

Active Policies

Policies define a Quality of Service for a Service and its Processes. A policy consists of a rule and an action or pipeline. This table lists the policies currently defined in your environment.

<input type="checkbox"/>	Service	Process	Policy
<input type="checkbox"/>	CreditCheckService	AdminServer	CreditCheckService-AdminServer-minprocess if Number.Processes is < 1 then run action CreditCheckServicestartaction
<input type="checkbox"/>		All	CreditCheckService-Deploy-key if Service.DeploymentState is starting then run action CreditCheckServicestartserviceaction
<input type="checkbox"/>		All	CreditCheckService-Undeploy-key if Service.DeploymentState is stopping then run action CreditCheckServicestopserviceaction
<input type="checkbox"/>		ManagedServers	CreditCheckService-ManagedServers-minprocess if Number.Processes is < 1 then run action CreditCheckServicestartaction
<input type="checkbox"/>		ManagedServers	MaxAverageJVMProcessorLoad if MaxAverageJvmProcessorLoad > 0.8 then run action CreditCheckServicestartaction

Runtime Policy Metadata Configuration

Listing 4-3 shows how the MaxAverageJVMProcessorLoad configuration for the constraint binding, custom constraint, and associated action is represented in the metadata-config.xml file.

Listing 4-3 Runtime Policy Metadata Configuration

```
.
.
.
<ns2:process-types>
  <ns2:process-type>
    <ns2:constraint-bindings>
      <ns2:constraint-binding>

<ns2:constraint-key>CreditCheckService-ManagedServers-minprocess</ns2:constraint-key>

<ns2:action-key>CreditCheckServicestartaction</ns2:action-key>
  </ns2:constraint-binding>
  <ns2:constraint-binding>

<ns2:constraint-key>MaxAverageJVMProcessorLoad</ns2:constraint-key>

<ns2:action-key>CreditCheckServicestartaction</ns2:action-key>
  </ns2:constraint-binding>
</ns2:constraint-bindings>
<ns2:name>ManagedServers</ns2:name>
```



```

        <ns2:description>ManagedServers-description</ns2:description>
<ns2:metadata-key>CreditCheckService-ManagedServersmetakey</ns2:metadata-key>
    </ns2:process-type>
.
.
.
<ns2:custom-constraint>
    <ns2:name>MaxAverageJVMPprocessorLoad</ns2:name>
    <ns2:key>MaxAverageJVMPprocessorLoad</ns2:key>
    <ns2:priority>0</ns2:priority>
    <ns2:state>deployed</ns2:state>
    <ns2:evaluation-period>10000</ns2:evaluation-period>
    <ns2:constraint>MaxAverageJVMprocessorLoad</ns2:constraint>
    <ns2:value>0.8</ns2:value>
</ns2:custom-constraint>
.
.
.
<ns2:action>
    <ns2:name>CreditCheckServicestartaction</ns2:name>
    <ns2:key>CreditCheckServicestartaction</ns2:key>

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

```

What's Next?

Now that we have defined the service to be managed and assigned deployment and runtime policies, we need to deploy the service as described in [Chapter 5, “Deploy the Service Against Available Resources.”](#)

Define the Service Under Management

Deploy the Service Against Available Resources

The next task in the use case example is to deploy the service to the resource pools on which it will run. To start a WLOC service, you deploy it using the WLOC Administration Console, or configure it for auto-deployment. WLOC chooses one or more resource pools for the initial deployment.

To choose resources pools for an initial deployment, WLOC follows this procedure:

1. The Controller examines the process requirements that you configured for the service.
2. The Controller examines all resource pools that are currently active—including those that are hosting other services—and uses the following process of elimination to determine which resource pools are candidates for hosting the service:
 - If the service specifies software requirements, resource pools that do not offer access to all of the required software are eliminated as candidates.
 - If the service consists of a single process, resource pools that offer fewer computing resources than the service's minimum resource requirements are eliminated.
 - If the service consists of multiple processes, WLOC may use multiple resource pools to run the service.
3. After this process of elimination, WLOC determines which resource pool or combination of resource pools can be used to host the service. Then, it uses one of the following placement algorithms that you configure when you create the service to choose a resource pool or collection of resource pools:
 - **Prefer resource pools with the most resources:** WLOC selects the resource pool combination that provides the greatest amount of computing resources.

- **Prefer resource pools with fewer resources:** WLOC selects the resource pool that most closely matches the minimum resource requirements of the service. This algorithm ensures the most efficient use of resources in your data center.

Deployment Scenario

In this example, we will deploy the CreditCheckService from the Administration Console. Note that before we start the service, the state of the service is undeployed, which indicates that there are no instances running. When we start the service, the following occurs:

1. The Controller evaluates the process requirements for each process group in the service. When we specified the resource requirements in [“Define Resource Requirements for the Service” on page 4-17](#), we specified that there should be a minimum of 1 process for each process group.
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. Because the only requirement that we specified is minimum number of processes, it will randomly choose one of the two Managed Servers for the ManagedServers process group.
3. Use the placement algorithm that we specified when we defined the service properties, **Prefer resource pools with the most resources**, to determine the resource pool on which to place the service.
4. The Controller stages each of the instances individually.
5. The Controller starts each of the instances individually. When the minimum number of processes from each group is started, the service is deployed.
6. The Controller evaluates the runtime policy. When the average load across all the instances in the ManagedServers process group exceeds the defined value, another JVM instance is started.

In a real world client application, a typical setting for the `MaxAverageJVMProcessorLoad` might be `.8`, indicating that when the average load across all the instances in the ManagedServers process group exceeds 80%, an additional JVM instance is started. For the purposes of this example, we set the value of `MaxAverageJVMProcessorLoad` to `0` to trigger the required action.

7. The runtime policy is continually evaluated. When the constraint is violated again and there are no other available processes to start, the action fails.

Deploy the Service

To deploy the CreditCheckService:

1. Click the **Inventory** tab in the WLOC navigation bar and click **Services**.

The **CreditCheckService** is displayed in the **Services** table.

2. Select the **CreditCheckService** check box and click **Start**.

The screenshot shows the WLOC interface. The top navigation bar includes 'Home', 'Inventory', 'Policies', 'Events', 'Monitoring', 'Controller', and 'Agents'. The 'Inventory' tab is active, and the 'Services' sub-tab is selected. The 'Services' table lists the following service:

Name	Status	Processes
<input checked="" type="checkbox"/> CreditCheckService	undeployed	0

Buttons for 'New', 'Delete', 'Start', 'Stop', 'Stage', and 'Unstage' are visible above and below the table. The 'Start' button is highlighted in the original image.

The message Request to start service was successfully sent is displayed, and the **Status** column in the table reflects the transition states of the service.

3. As the service is proceeding through the various transition states of the deployment process, view the following in the console:
 - The runtime status changes in the **Status** column to reflect the current or transitional state as follows:
 - undeployed—No JVM instances are running.
 - staging—The service is transitioning from undeployed to staged state.
 - staged—No JVM instances are running (same as undeployed for a Plain Agent.)
 - starting—The service is transitioning from staged to deployed state.
 - deployed—The minimum number (at least) of JVMs associated with the service are running.

Deploy the Service Against Available Resources

Request to start service was successfully sent

Services

Service(s) are collections of software and processes. This table lists the services currently defined in your environment.

New Delete | Start Stop Stage Unstage

Type Filter (Ex. Column=Value) Filter Clear Filter Showing 1 - 1 of 1 << < > >>

<input type="checkbox"/> Name	Status	Processes
<input type="checkbox"/> CreditCheckService	starting	2

New Delete | Start Stop Stage Unstage

– The **Task and Event Viewer** displays each event as it occurs.

Tasks and Events	
Time Stamp	Description
Tue Apr 01 17:20:32 EDT 2008	A JVM AdminServerJVM has started...
Tue Apr 01 17:20:32 EDT 2008	Action Succeeded...FindInstancesAction
Tue Apr 01 17:20:32 EDT 2008	Action Succeeded...FindPlacementsAction
Tue Apr 01 17:20:32 EDT 2008	Action Succeeded...ReserveInstancesAction
Tue Apr 01 17:20:31 EDT 2008	Service CreditCheckService is ...starting
Tue Apr 01 17:20:22 EDT 2008	Action Succeeded...DeployService-Pipeline

When the minimum number of instances are started, the service state is changed to deployed and the runtime policy is evaluated. The Task and Event Viewer indicates a Quality of Service (QOS) violation and the second ManagedServers JVM instance is started.

Home | Inventory | Policies | Events | Monitoring | Controller | Agents

Request to start service was successfully sent

Inventory

- Interactive
 - Liquid Operations Environment
 - Services
 - Resource Pools
 - Agents

Services

Service(s) are collections of software and processes. This table lists the services currently defined in your environment.

New | Delete | Start | Stop | Stage | Unstage

Type Filter (Ex. Column=Value) | Filter | Clear Filter | Showing 1 - 1 of 1 << < | > >>

<input type="checkbox"/> Name	Status	Processes
<input checked="" type="checkbox"/> CreditCheckService	deployed	3

New | Delete | Start | Stop | Stage | Unstage

Tasks and Events

TimeStamp	Description
Tue Apr 01 17:22:23 EDT 2008	A JVM MS_2 has started...
Tue Apr 01 17:22:23 EDT 2008	A JVM MS_2 has been initialized...
Tue Apr 01 17:22:23 EDT 2008	A new JVM MS_2 was created...
Tue Apr 01 17:22:23 EDT 2008	QOS violation policy MaxAverageJVMProcessorLoad MaxAverageJvmProcessorLoad.MaxAverageJVMProcessorLoad is not greater than 0 - current value is 0.004004004004004004
Tue Apr 01 17:22:12 EDT 2008	Action Succeeded...CreditCheckServicestartserviceaction

The runtime policy is continually evaluated. Another QOS violation occurs. Because there are no other available processes to start, the action fails.

Tasks and Events

TimeStamp	Description
Tue Apr 01 17:23:12 EDT 2008	Action Failed... CreditCheckServicestartaction Could not reserve process CreditCheckService:ManagedServers
Tue Apr 01 17:23:12 EDT 2008	QOS violation policy MaxAverageJVMProcessorLoad MaxAverageJvmProcessorLoad.MaxAverageJVMProcessorLoad is not greater than 0 - current value is 0.02719713375298957
Tue Apr 01 17:23:08 EDT 2008	Action Succeeded...CreditCheckServicestartaction
Tue Apr 01 17:23:08 EDT 2008	Action Succeeded...CreditCheckServicestartaction
Tue Apr 01 17:22:23 EDT 2008	A JVM MS_2 has started...

What's Next?

After the service has deployed successfully, we can monitor the available resources as described in [Chapter 6, “Monitor WLOC Services and Resources.”](#)

Deploy the Service Against Available Resources

Monitor WLOC Services and Resources

Now that the service is deployed, we can use the WLOC Administration Console to monitor how well the service is meeting its service level agreement, and to see which resource pools are hosting services.

To gather monitoring data, the WLOC Controller either actively polls the monitored object or passively listens for changes to a monitored object, depending on the type of data that it is gathering.

The Monitoring tab of the WLOC Administration Console contains a dashboard that enables you to construct graphs to chart metrics of services. Specifically, you can:

- Create views to organize charts according to your needs. For example, you can define one view that contains a set of charts to monitor your environment at a high-level. You can then define additional views to access more specific details, such as CPU usage on a specific set of JVMs.
- Develop custom metrics that are defined as functions of monitorable resource attributes.
- Set preferences to customize charts and graphs as desired.

The tasks described in this topic include:

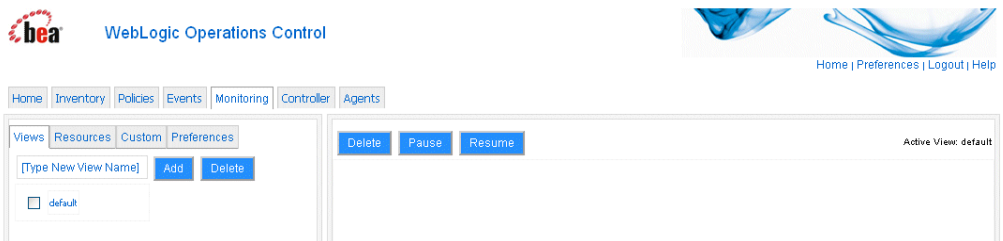
- [Create a View](#)
- [Browse the Resources Pane](#)

Create a View

To create a view:

1. Click the **Monitoring** tab in the WLOC navigation bar.

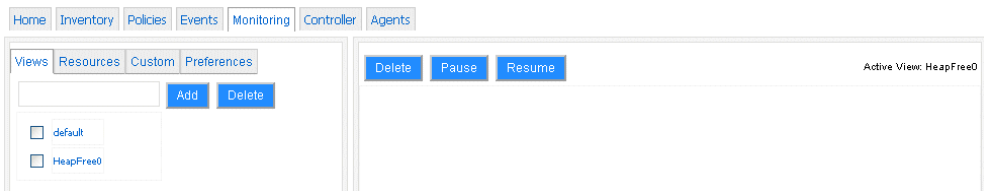
The **Views** tab is selected by default. Because we have not created any views yet, only the Default view is listed.



2. In the text box, enter the name of the view that you want to create and click **Add**.

In this example, we will create a view named **HeapFree** which will show the current free heap metrics for the MS_1 JVM. The view name is added to the list.

3. Click the **HeapFree** view name to make it the active view.



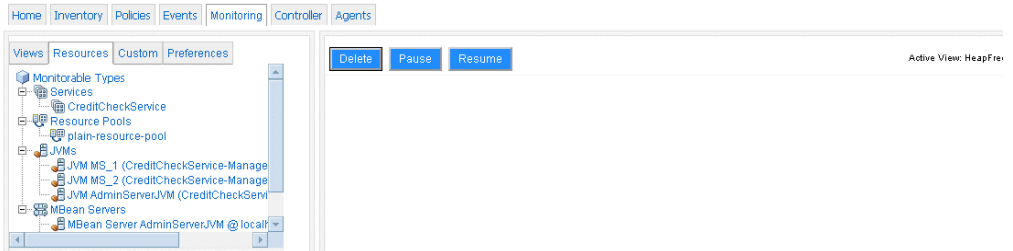
4. Add monitoring charts to a view by selecting resources to monitor using standard metrics available from the Resources pane. For instructions on browsing the Resources pane and adding charts to a view, see [“Browse the Resources Pane” on page 6-2](#).

Browse the Resources Pane

When you select the **Resources** tab, it displays a list of the resources that you can monitor. Using this pane, you can monitor the CPU and memory usage for the services, resource pools, and JVMs. You can also monitor the values of one or more MBean attributes, over time, for each of the MBean Servers.

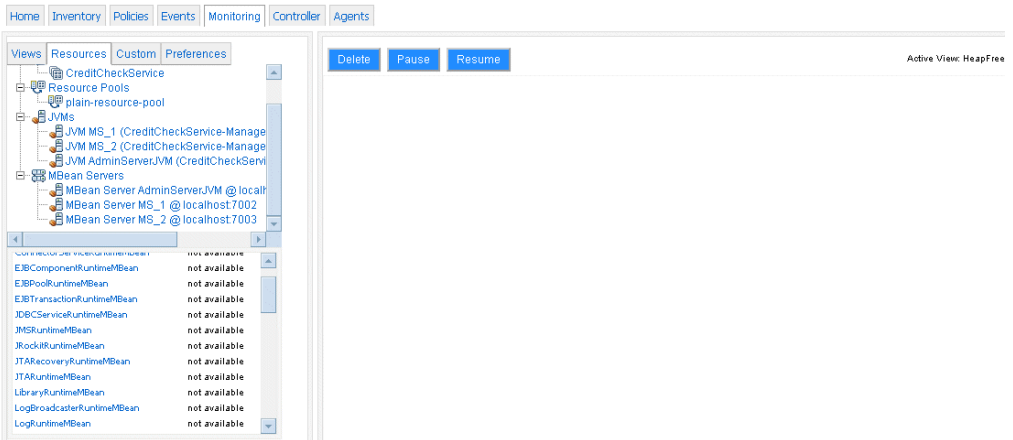
To browse the Resources pane and add a chart to the HeapFree view:

1. Select the **Resources** tab.
2. Expand the list of resources in the **Resources** pane to view the monitorable types for each resource.



3. Scroll down to view the available MBean Servers and select **MBean Server MS_1** from the list.

A list of the available MBeans types is displayed.



4. Select the **JRockitRuntimeMBean** type from the list.

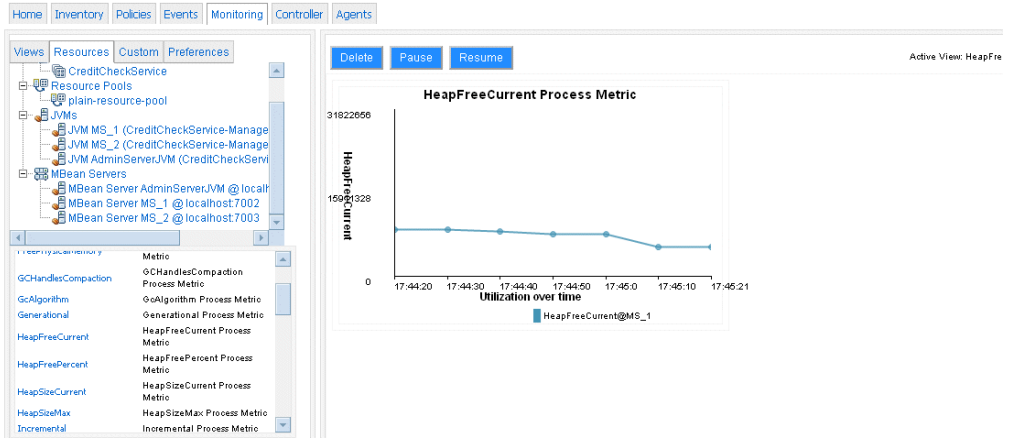
The MBean instance **MS_1** is displayed at the bottom of the **Resources** pane.

5. Click **MS_1**.

A list of the monitorable MBean attributes for **MS_1** is displayed.

6. Scroll down the list of MBean attributes and select the **HeapFreeCurrent** attribute. The metric chart illustrating the current free heap utilization, over time, is added to the **HeapFree** view.

Monitor WLOC Services and Resources



The HeapFree view is saved automatically.