Artix™ ESB

BMC Patrol Integration Guide

Version 5.1, December 2007

Making Software Work Together™

Updated: April 10, 2008

Contents

List of Figures 5

Preface 7
- What is covered in this book 7
- Who should read this book 7
- Organization of this book 7
- The Artix Documentation Library 8

Chapter 1 Integrating with BMC Patrol™ 9
- Introduction 10
- The IONA BMC Patrol Integration 14

Chapter 2 Configuring your IONA Product 17
- Setting up your Artix Environment 18
- Setting up your Orbix Environment 21

Chapter 3 Using the IONA BMC Patrol Integration 27
- Setting up your BMC Patrol Environment 28
- Using the IONA Knowledge Module 30

Chapter 4 Extending to a Production Environment 37
- Configuring an Artix Production Environment 38
- Configuring an Orbix Production Environment 42

Index 47

List of Figures

Figure 1: Overview of the IONA BMC Patrol Integration 12
Figure 2: IONA Server Running in BMC Patrol 15
Figure 3: BMC Patrol Displaying Alarms 16
Figure 4: Orbix Configuration GUI 21
Figure 5: Selecting EMS Configuration 22
Figure 6: Selecting Performance Logging 23
Figure 7: Graphing for IONAAvgResponseTime 33
Figure 8: Alarms for IONAAvgResponseTime 34

Preface

What is covered in this book

IONA products support integration with Enterprise Management Systems such as IBM Tivoli™, BMC Patrol™, and CA WSDM™. This guide explains how to integrate Artix and Orbix with BMC Patrol. This book applies to Artix applications written using C++, JAX-RPC (Java XML-Based APIs for Remote Procedure Call), and JAX-WS (Java XML-Based APIs for Web Services).

Who should read this book

This guide is aimed at system administrators using BMC Patrol to manage distributed enterprise environments, and developers writing distributed enterprise applications. Administrators do not require detailed knowledge of the technology that is used to create distributed enterprise applications.

This book assumes that you already have a good working knowledge of the BMC Patrol range of products.

Organization of this book

This book contains the following chapters:

- Chapter 1 introduces Enterprise Management Systems, and IONA's integration with BMC Patrol.
- Chapter 2 describes how to configure your IONA product for integration with BMC Patrol.
- Chapter 3 describes how to configure your BMC Patrol environment for integration with IONA products.
- Chapter 4 describes how to extend an IONA BMC Patrol integration from a test environment to a production environment

The Artix Documentation Library

For information on the organization of the Artix library, the document conventions used, and where to find additional resources, see Using the Artix Library.

CHAPTER 1

Integrating with BMC Patrol™

This chapter introduces the integration of IONA products with the BMC Patrol™ Enterprise Management System. It describes the requirements and main components of this integration.

In this chapter

This chapter contains the following sections:

Introduction	page 10
The IONA BMC Patrol Integration	page 14

Introduction

Overview

IONA products support integration with Enterprise Management Systems such as BMC Patrol. This section includes the following topics:

- "The application life cycle".
- "Enterprise Management Systems".
- "IONA EMS integration".
- "IONA BMC Patrol features".
- "How it works".

The application life cycle

Most enterprise applications go through a rigorous development and testing process before they are put into production. When applications are in production, developers rarely expect to manage those applications. They usually move on to new projects, while the day-to-day running of the applications is managed by a production team. In some cases, the applications are deployed in a data center that is owned by a third party, and the team that monitors the applications belongs to a different organization.

Enterprise Management Systems

Different organizations have different approaches to managing their production environment, but most will have at least one *Enterprise Management System* (EMS).

For example, the main Enterprise Management Systems include BMC Patrol™ and IBM Tivoli™. These systems are popular because they give a top-to-bottom view of every part of the IT infrastructure.

This means that if an application fails because the `/tmp` directory fills up on a particular host, for example, the disk space is reported as the fundamental reason for the failure. The various application errors that arise are interpreted as symptoms of the underlying problem with disk space. This is much better than being swamped by an event storm of higher-level failures that all originate from the same underlying problem. This is the fundamental strength of integrated management.

IONA EMS integration

IONA's Orbix and Artix products are designed to integrate with Enterprise Management Systems. IONA's common management instrumentation layer provides a base that can be used to integrate with any EMS.

In addition, IONA provides packaged integrations that provide out-of-the-box integration with major EMS products. This guide describes IONA's integration with BMC Patrol products.

IONA BMC Patrol features

The IONA BMC Patrol integration performs the following key enterprise management tasks:

- Posting an event when a server crashes. This enables programmed recovery actions to be taken.
- Tracking key server metrics (for example, server response times). Alarms are triggered when these go out of bounds.

The server metrics tracked by the IONA BMC Patrol integration include the number of invocations received, and the average, maximum and minimum response times. The IONA BMC Patrol integration also enables you to track these metrics for individual operations. Events can be generated when any of these parameters go out of bounds. You can also perform a number of actions on servers including stopping, starting and restarting.

How it works

In the IONA BMC Patrol integration, key server metrics are logged by the IONA performance logging plugins. Log file interpreting utilities are then used to analyze the logged data.

The IONA BMC Patrol integration provides IONA Knowledge Modules, which conform to standard BMC Knowledge Module design and operation. These modules tell the BMC Patrol console how to interpret the IONA logging data. Figure 1 on page 12 shows a simplified view of how the IONA Knowledge Modules work. In this example, an alarm is triggered in the BMC Patrol console when a locator becomes unresponsive, and this results in an action to restart the locator.

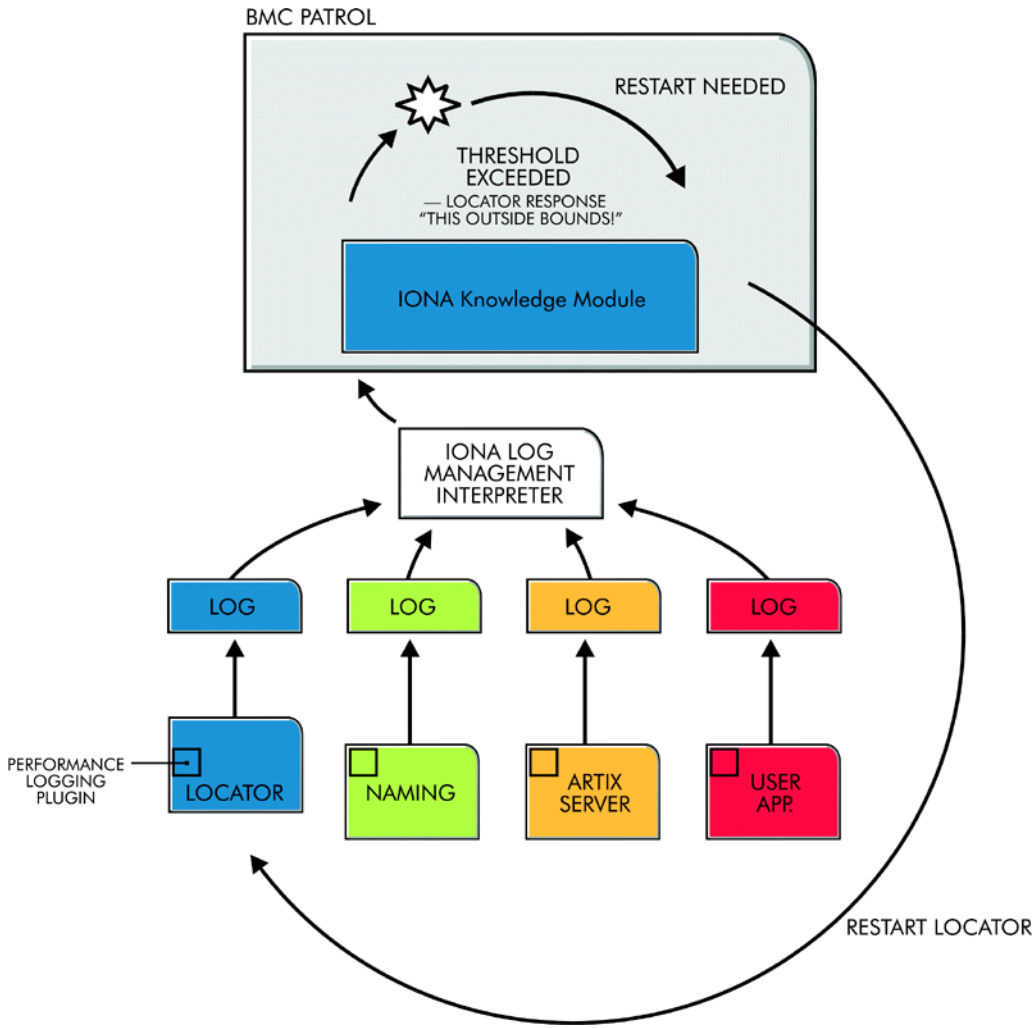

Figure 1: *Overview of the IONA BMC Patrol Integration*

The IONA performance logging plugins collect data relating to server response times and log it periodically in the performance logs. The IONA Knowledge Module executes parameter collection periodically on each host, using the IONA log file interpreter running on each host to collect and summarize the logged data.

The IONA Knowledge Module compares the response times and other values against the defined alarm ranges, and issues an alarm event if a threshold has been breached. These events can be analyzed and appropriate action taken automatically (for example, restart a server). Alternatively, the user can intervene manually and execute a BMC Patrol menu command to stop, start or restart the offending server.

Artix C++ runtime

This overview applies to Artix server applications written using C++. It also applies to Java applications written using JAX-RPC (Java API for XML-Based Remote Procedure Call).

Artix Java runtime

There are some differences for Artix Java applications written using JAX-WS (Java API for XML-Based Web Services). In this case, JMX management interceptors are used to collect response times from each message as they pass through the interceptor. From the interceptor's point of view, this is similar to the C++ runtime. The main difference is that it does not talk to the C++ transport layer using the Java Native Interface (JNI).

The Artix Java runtime's BMC integration provides the same functionality as the C++ runtime integration in terms of collecting the application performance data. However, start and stop scripts are not generated for Artix Java applications written using JAX-WS, so the start and stop functionality in the IONA Knowledge Module is not supported.

The IONA BMC Patrol Integration

Overview

This section describes the requirements and main components of IONA's BMC Patrol integration. It includes the following topics:

- "IONA requirements".
- "BMC Patrol requirements".
- "Main components".
- "Example metrics".
- "Further information".

IONA requirements

IONA's Artix and Orbix products are fully integrated with BMC Patrol. You must have at least one of the following installed:

- Artix 2.1 or higher.
- Orbix 6.1 or higher.

BMC Patrol requirements

To use the IONA BMC Patrol integration, you must have BMC Patrol 3.4 or higher. The IONA BMC Patrol integration is compatible with the BMC Patrol 7 Central Console.

Main components

The IONA BMC Patrol integration consists of the following Knowledge Modules (KM):

- `IONA_SERVERPROVIDER`
- `IONA_OPERATIONPROVIDER`

`IONA_SERVERPROVIDER.km` tracks key metrics associated with your IONA servers on a particular host. It also enables servers to be started, stopped, or restarted, if suitably configured.

`IONA_OPERATIONPROVIDER.km` tracks key metrics associated with individual operations on each server.

Example metrics

Figure 2 shows an example of the `IONA_SERVERPROVIDER` Knowledge Module displayed in BMC Patrol. The window in focus shows the IONA performance metrics that are available for an operation named `query_reservation`, running on a machine named `stimulator`.

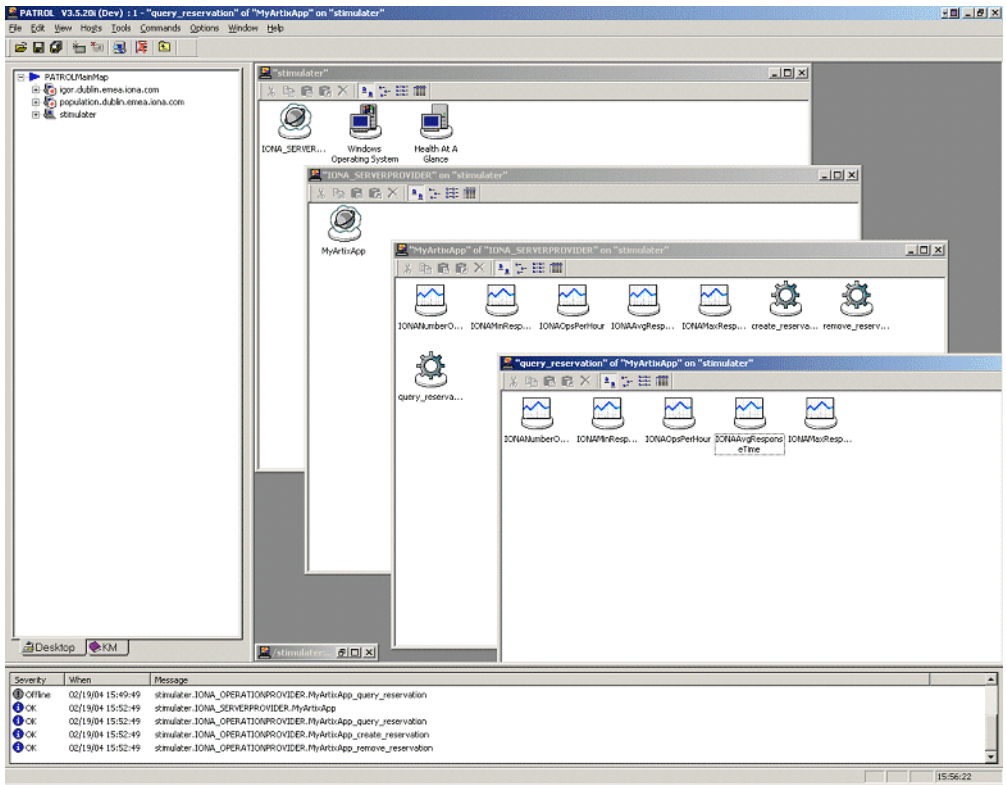

Figure 2: *IONA Server Running in BMC Patrol*

The IONA server performance metrics include the following:

- `IONAAvgResponseTime`
- `IONAMaxResponseTime`
- `IONAMinResponseTime`
- `IONANumInvocations`
- `IONAOpsPerHour`

For more details, see "Using the IONA Knowledge Module" on page 30.

Figure 3 shows alarms for server metrics, for example, `IONAAvgResponseTime`. This measures the average response time of all operations on this server during the last collection cycle.

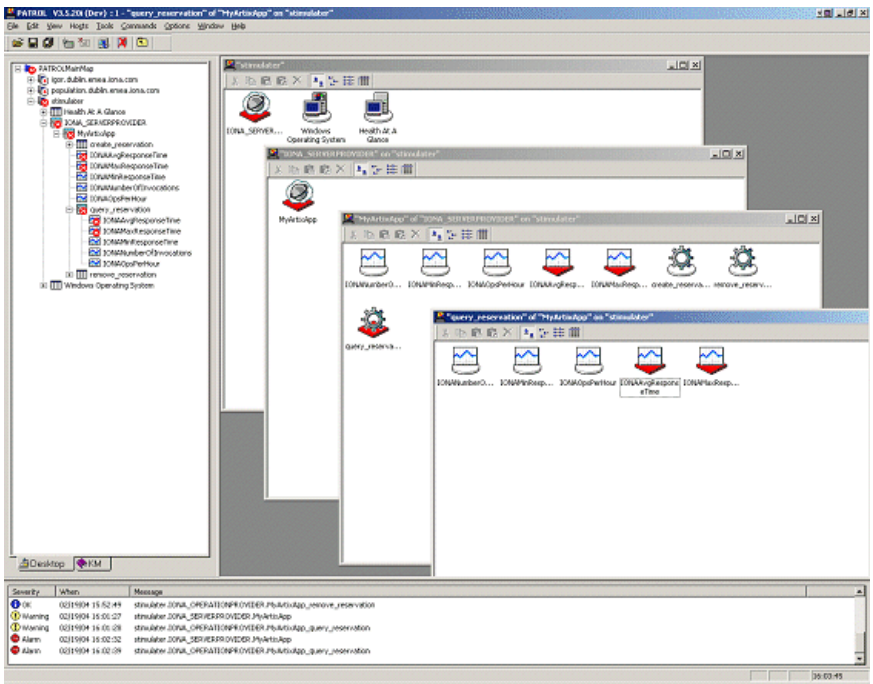

Figure 3: *BMC Patrol Displaying Alarms*

Further information

For a detailed description of Knowledge Modules, see your BMC Patrol documentation.

CHAPTER 2

Configuring your IONA Product

This chapter explains the steps that you need to perform in your IONA product to configure integration with BMC Patrol.

In this chapter

This chapter contains the following sections:

Setting up your Artix Environment	page 18
Setting up your Orbix Environment	page 21

Setting up your Artix Environment

Overview

The best way to learn how to use the BMC Patrol integration is to start with a host that has both BMC Patrol and Artix installed. This section explains how to make your Artix servers visible to BMC Patrol. It includes the following topics:

- "EMS configuration files".
- "Creating a servers.conf file".
- "Creating a server_commands.txt file".
- "Further information".

EMS configuration files

You need to create two text files that are used to configure the BMC Patrol integration:

- `servers.conf`
- `server_commands.txt`

These files are used to track your Artix applications in BMC Patrol. You will find starting point files in the `IONA_km.zip` located in the following directory of your Artix installation:

```
ArtixInstallDir\artix_Version\cxx_java\management\BMC\IONA_km.zip
```

When you unzip, the starting point files are located in the `lib/iona/conf` directory.

Creating a servers.conf file

The `servers.conf` file is used to instruct BMC Patrol to track your Artix servers. It contains the locations of performance log files for specified applications. Each entry must take the following format:

```
my_application, 1, /path/to/myproject/log/myapplication_perf.log
```

This example entry instructs BMC Patrol to track the `myapplication` server, and reads performance data from the following log file:

```
/path/to/myproject/log/myapplication_perf.log
```

You must add entries for the performance log file of each Artix server on this host that you wish BMC Patrol to track. BMC Patrol uses the `servers.conf` file to locate these log files, and then scans the logs for information about the server's key performance indicators.

The following example is taken from the Artix Java sample application for BMC Patrol integration:

```
management-bmc-patrol-demo-server,1,%ARTIX_HOME%\java\samples\advanced\
   management\bmc-patrol\BMCCounterServer.log
management-bmc-patrol-demo-client,1,%ARTIX_HOME%\java\samples\advanced\
   management\bmc-patrol\BMCCounterClient.log
```

Creating a server_commands.txt file

The `server_commands.txt` file is used to instruct BMC Patrol how to start, stop, and restart your Artix servers. It contains the locations of the relevant scripts for specified servers. Each entry must take the following format:

```
myapplication,start=/path/to/myproject/bin/start_myapplication.sh
myapplication,stop=/path/to/myproject/bin/stop_myapplication.sh
myapplication,restart=/path/to/myproject/bin/restart_myapplication.sh
```

In this example, each entry specifies a script that can be used to stop, start, or restart the `myapplication` server. When BMC Patrol receives an instruction to start `myapplication`, it looks up the `server_commands.txt` file, and executes the script specified in the appropriate entry.

You must add entries that specify the relevant scripts for each server on this host that you wish BMC Patrol to track.

Note: Start and stop scripts are generated for Artix applications written using C++ or JAX-RPC only. Applications written using JAX-WS are not supported. For more details, see "Artix Java runtime" on page 13.

Copy the EMS files to your BMC installation

When you have added content to your `servers.conf` and `server_commands.txt` files, copy these files into your BMC installation, for example:

```
$PATROL_HOME/lib/iona/conf
```

This enables tracking of your Artix server applications in BMC Patrol.

Further information

For details of how to configure your Artix servers to use performance logging, see "Configuring an Artix Production Environment" on page 38.

For a complete explanation of configuring performance logging, see the Configuring and Deploying Artix Solutions.

Setting up your Orbix Environment

Overview

The best way to learn how to use the BMC Patrol integration is to start with a host that has both BMC Patrol and Orbix installed. This section explains the configuration steps in your Orbix environment. It includes the following:

- "Creating an Orbix configuration domain".
- "Generating EMS configuration files".
- "Configuring performance logging".
- "EMS configuration files".
- "Creating a servers.conf file".
- "Creating a server_commands.txt file".
- "Further information".

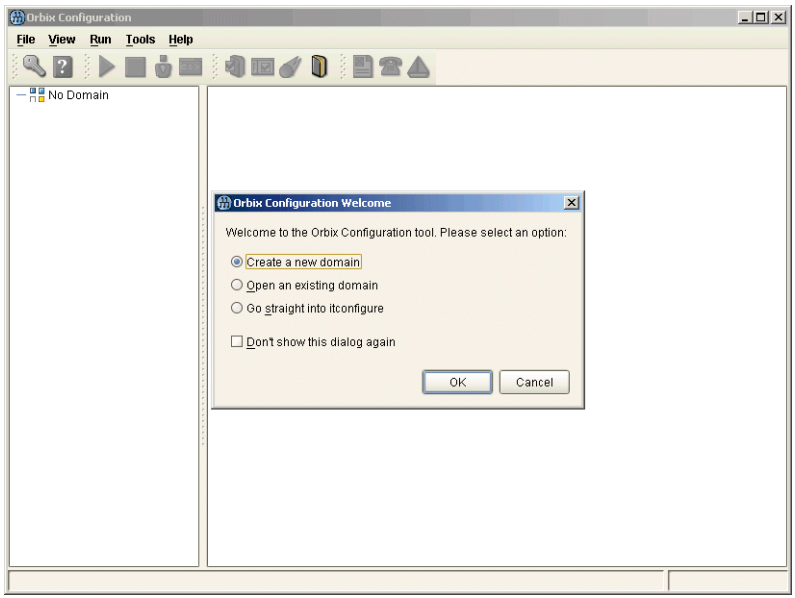

Figure 4: *Orbix Configuration GUI*

Creating an Orbix configuration domain

You must first create the Orbix configuration domain that you want to monitor using the **Orbix Configuration** GUI.

To launch this tool, enter `itconfigure` on the command line. The GUI is shown in Figure 4.

Generating EMS configuration files

To generate EMS configuration files, perform the following steps:

1. Click **Go straight into itconfigure** in the welcome dialog.
1. Select **File|New|Expert** from the GUI main menu. This displays the **Domain Details** screen, as shown in Figure 5.
2. Select the **Generate EMS Configuration Files** checkbox. This generates the configuration files required for your BMC Patrol integration.

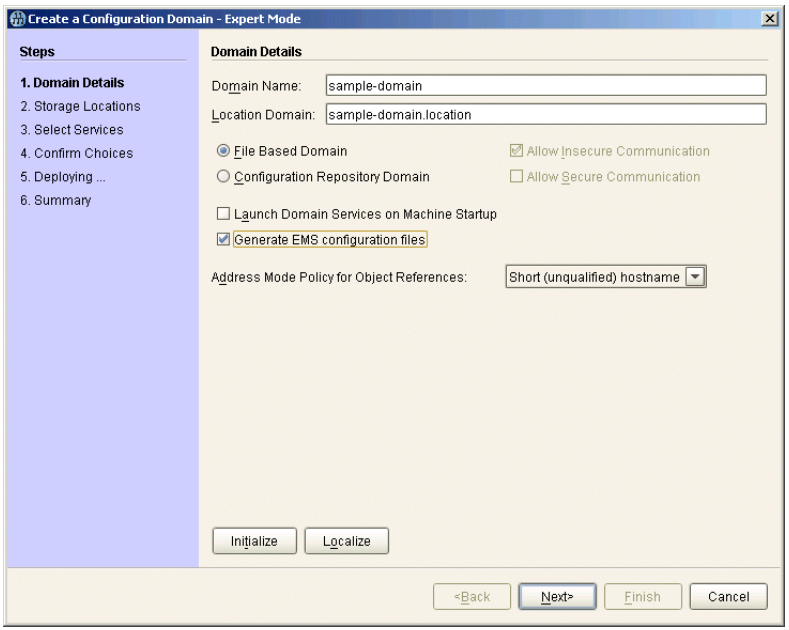

Figure 5: *Selecting EMS Configuration*

3. Proceed as normal following the steps in the wizard until you get to the **Select Services** screen (see "Configuring performance logging").

Configuring performance logging

To configure performance logging, do the following:

1. In the **Select Services** screen, click **Settings** to launch the **Domain Defaults** dialog, shown in Figure 6.
2. Select the **Performance Logging** option in the **Other Properties** box, shown in Figure 6. This ensures that, by default, all your selected services are configured for monitoring.

   If you want to enable BMC Patrol to start, stop, or restart your servers, also select the **Launch Service on Domain Startup** option in the **Service Launching** box.

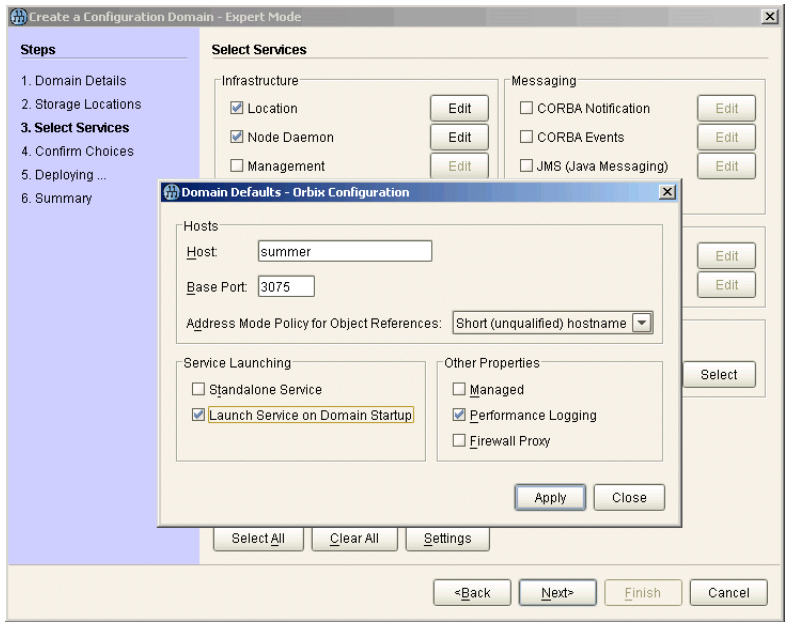

Figure 6: *Selecting Performance Logging*

   Alternatively, you can configure these settings separately for each service by selecting the service, and clicking the **Edit** button.

3. Click **Apply**, and then **Close**.

4. Click **Next** to view a **Confirmation** screen for your selected configuration.
5. Click **Next** to deploy your configuration.
6. Click **Finish** to exit.

Note: When you configure EMS integration, you must also configure performance logging. This is not optional. However, you can configure performance logging without EMS integration. For full details, see the *Orbix Management User's Guide.*

EMS configuration files

When the domain is created, you can start it like any other domain, using the start script in your *OrbixInstall*`/etc/bin` directory. Selecting the performance logging feature has enabled some extra configuration and logging. In your *OrbixInstall*`/var/domain-name` directory, you will find the following EMS configuration files:

- `servers.conf`
- `server_commands.txt`

The servers.conf file

When you open the `servers.conf` file, you will see a number of entries in the following form:

```
ServerName, Number, /Path/to/a/Log/File
```

For example:

```
mydomain_locator_myhost, 1,
   /opt/iona/var/mydomain/logs/locator_myhost_perf.log
```

The `servers.conf` file lists the servers that you want BMC Patrol to monitor on a particular host. To begin with, assume that you are running all services in the domain on one host. For example, assume your `servers.conf` file has the above entry. When you have started your domain, you should see a log file in the following location:

```
/opt/iona/var/mydomain/logs/locator_perf.log
```

There will be one of these files for each server that you want to monitor. The IONA resource model uses the `servers.conf` file to locate these logs and then scans the logs for information about the server's key performance indicators.

The server_commands.txt file

When you open the `server_commands.txt` file, you will see a number of entries of the form:

```
ServerName,Action=/Path/to/Script
```

For example:

```
mydomain_locator_myhost,start
=/opt/iona/var/mydomain/locator_myhost_start.sh
```

Each entry in this file contains a pointer to a script that implements an action on a particular server. In this example, the action is a start action for the server `mydomain_locator_myhost`. When BMC Patrol receives an instruction to start the locator in a domain named `mydomain` on a host named `myhost`, it looks up the `server_commands.txt` file on `myhost`, and execute the script pointed to in this entry.

Further information

For a complete explanation of configuring performance logging plugins, see Configuring and Deploying Artix Solutions.

CHAPTER 3

Using the IONA BMC Patrol Integration

This chapter explains the steps the that you must perform in your BMC Patrol environment to monitor IONA applications. It also describes the IONA Knowledge Module and how to use it to monitor servers and operations. It assumes that you already have a good working knowledge of BMC Patrol.

In this chapter

This chapter contains the following sections:

Setting up your BMC Patrol Environment	page 28
Using the IONA Knowledge Module	page 30

Setting up your BMC Patrol Environment

Overview

To enable monitoring of the Artix or Orbix servers on your host, you must first perform the following steps in your BMC Patrol environment:

1. "Install the IONA Knowledge Module".
2. "Set up your Java environment".
3. "Set up your EMS configuration files".
4. "View your servers in the BMC Console".

Install the IONA Knowledge Module

The IONA BMC Patrol integration is shipped in two formats:

Windows *ArtixInstallDir**Version*\cxx_java\management\BMC\IONA_km.zip

UNIX *ArtixInstallDir/Version*/cxx_java/management/BMC/IONA_km.tgz

To install the IONA Knowledge Module, do the following:

Windows

Use WinZip to unzip `IONA_km.zip`. Extract this file into your `%PATROL_HOME%` directory.

If this is successful, the following directory is created:

```
%PATROL_HOME%\lib\iona
```

UNIX

Copy the `IONA_km.tgz` file into `$PATROL_HOME`, and enter the following commands:

```
$ cd $PATROL_HOME
$ gunzip IONA_km.tgz
$ tar xvf IONA_km.tar
```

Set up your Java environment

The IONA Knowledge Module requires a Java Runtime Environment (JRE). If your BMC Patrol installation already has a `$PATROL_HOME/lib/jre` directory, it should work straightaway. If not, you must setup a JRE (version 1.3.1 or later) on your machine as follows:

1. Copy the `jre` directory from your Java installation into `$PATROL_HOME/lib`. You should now have a directory structure that includes `$PATROL_HOME/lib/jre`.
2. Confirm that you can run `$PATROL_HOME/lib/jre/bin/java`.

Set up your EMS configuration files

In Chapter 2, you generated the following EMS configuration files:

- `servers.conf`
- `server_commands.txt`

Copy these generated files to `$PATROL_HOME/lib/iona/conf`.

View your servers in the BMC Console

To view your servers in the **BMC Console**, and check that your setup is correct, perform the following steps:

1. Start your **BMC Console** and connect to the **BMC Patrol Agent** on the host where you have installed the IONA Knowledge Module.
2. In the **Load KMs** dialog, open the `$PATROL_HOME/lib/knowledge` directory, and select the `IONA_SERVER.kml` file. This will load the `IONA_SERVERPROVIDER.km` and `IONA_OPERATIONPROVIDER.km` Knowledge Modules.
3. In your **Main Map**, the list of servers that were configured in the `servers.conf` file should be displayed. If they are not currently running, they are shown as offline.

You are now ready to manage these servers using BMC Patrol.

Using the IONA Knowledge Module

Overview

This section describes the IONA Knowledge Module and explains how to use it to monitor servers and operations. It includes the following topics:

- "Server Provider parameters".
- "Monitoring servers".
- "Monitoring operations".
- "Operation parameters".
- "Starting, stopping and restarting servers".
- "Troubleshooting".

Server Provider parameters

The `IONA_SERVERPROVIDER` class represents instances of IONA server or client applications. The parameters exposed in the Knowledge Module are shown in Table 1.

Table 1: *IONA Server Provider Parameters*

Parameter Name	Default Warning	Default Alarm	Description
IONAAvgResponseTime	1000–5000	> 5000	The average response time (in milliseconds) of all operations on this server during the last collection cycle.
IONAMaxResponseTime	1000–5000	> 5000	The slowest operation response time (in milliseconds) during the last collection cycle.
IONAMinResponseTime	1000–5000	> 5000	The quickest operation response time (in milliseconds) during the last collection cycle.
IONANumInvocations	10000–100000	> 100000	The number of invocations received during the last collection period.
IONAOpsPerHour	1000000–10000000	> 10000000	The throughput (in Operations Per Hour) based on the rate calculated from the last collection cycle.

Monitoring servers

You can use the parameters shown in Table 1 to monitor the load and response times of your IONA servers.

The Default Alarm ranges can be overridden on any particular instance, or on all instances, using the BMC Patrol 7 Central console. You can do this as follows:

1. In the **PATROL Central** console's **Main Map**, right click on the selected parameter and choose the **Properties** menu item.
2. In the **Properties** pane, select the **Customization** tab.
3. In the **Properties** drop-down list, select ranges.
4. You can customize the alarm ranges for this parameter on this instance. If you want to apply the customization to all instances, select the **Override All Instances** checkbox.

Note: The `IONANumInvocations` parameter is a raw, non-normalized metric and can be subject to sampling errors. To minimize this, keep the performance logging period relatively short, compared to the poll time for the parameter collector.

Monitoring operations

In the same way that you can monitor the overall performance of your servers and clients, you can also monitor the performance of individual operations. In Orbix, an operation equates to an operation on an IDL interface. In Artix, an operation relates to a WSDL operation defined on a port.

In many cases, the most important metrics relate to the execution of particular operations. For example, it could be that the `make_reservation()`, `query_reservation()` calls are the operations that you are particularly interested in measuring. This means updating your `servers.conf` file as follows:

```
mydomain_myserver,1,/var/mydomain/logs/myserver_perf.log,[make_reservation,query_reservation]
```

In this example, the addition of the bold text enables the `make_reservation` and `query_reservation` operations to be tracked by BMC Patrol.

Operation parameters

Table 2 shows the IONA parameters that are tracked for each operation instance:

Table 2: *IONA Operation Provider Parameters*

Parameter Name	Default Warning	Default Alarm	Description
IONAAvgResponseTime	1000–5000	> 5000	The average response time (in milliseconds) for this operation on this server during the last collection cycle.
IONAMaxResponseTime	1000–5000	> 5000	The slowest invocation of this operation (in milliseconds) during the last collection cycle.
IONAMinResponseTime	1000–5000	> 5000	The quickest invocation (in milliseconds) during the last collection cycle.
IONANumInvocations	10000–100000	> 100000	The number of invocations of this operation received during the last collection period.
IONAOpsPerHour	1000000–100000000	> 10000000	The number of operations invoked in a one hour period based on the rate calculated from the last collection cycle.

Figure 7 shows BMC Patrol graphing the value of the `IONAAvgResponseTime` parameter on a `query_reservation` operation call.

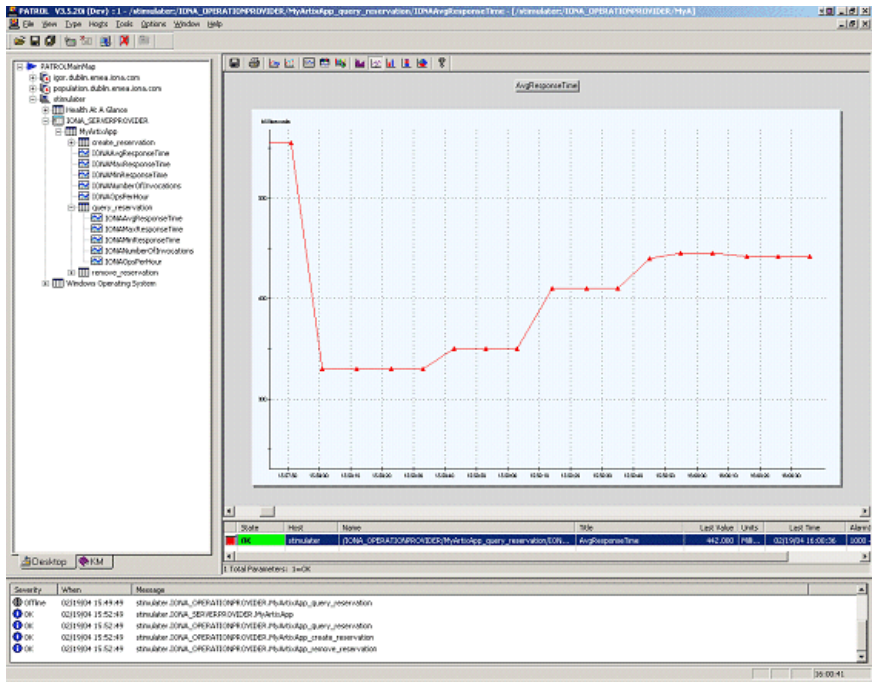

Figure 7: *Graphing for IONAAvgResponseTime*

Figure 8 shows warnings and alarms issued for the `IONAAvgResponseTime` parameter.

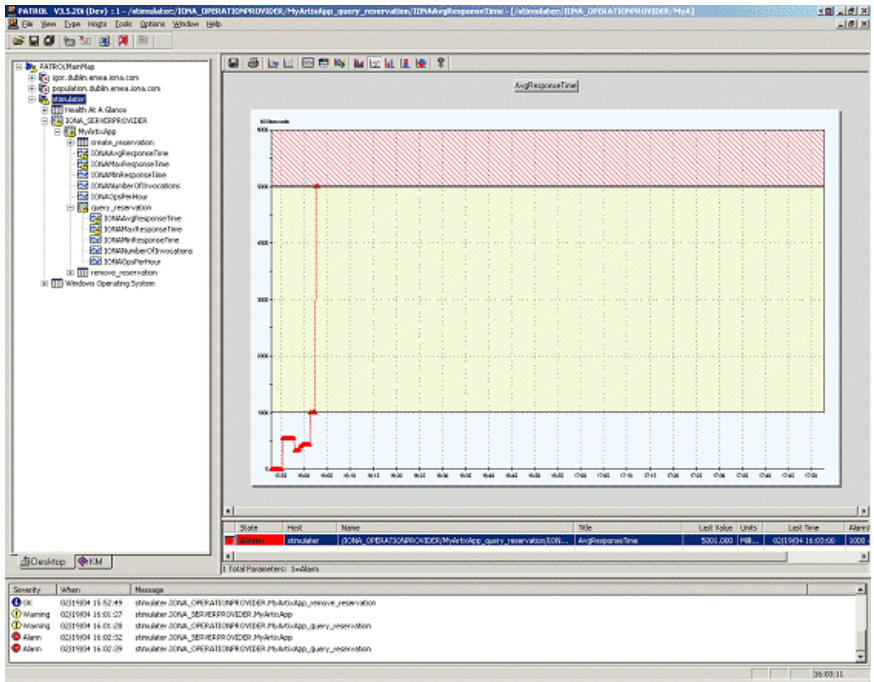

Figure 8: *Alarms for IONAAvgResponseTime*

Starting, stopping and restarting servers

The `server_commands.txt` file contains the details about the commands for services that you are deploying on your host (see Chapter 2). To execute commands in this file, perform the following steps:

1. Right click on an instance in the BMC Patrol Console **Main Map**.
2. Select **Knowledge Module Commands|IONA|Commands**.
3. Select one of the following commands:

Start	Starts a server.
Stop	Stops a server.
Restart	Executes a stop followed by a start.

Troubleshooting

If you have difficulty getting the IONA BMC Patrol integration working, you can use the menu commands to cause debug output to be sent to the system output window.

To view the system output window for a particular host, right click on the icon for your selected host in the BMC Patrol **Main Map**, and choose **System Output Window**.

You can change the level of diagnostics for a particular instance by right clicking on that instance and choosing:

Knowledge Module Commands|IONA|Log Levels

You can choose from the following levels:

- **Set to Error**
- **Set to Info**
- **Set to Debug**

Set to Debug provides the highest level of feedback and **Set to Error** provides the lowest.

CHAPTER 4

Extending to a Production Environment

This section describes how to extend an IONA BMC Patrol integration from a test environment to a production environment.

In this chapter

This chapter contains the following sections:

Configuring an Artix Production Environment	page 38
Configuring an Orbix Production Environment	page 42

Configuring an Artix Production Environment

Overview

This section describes the steps that you need to take when extending the IONA BMC Patrol integration from an Artix test environment to a production environment. It includes the following sections:

- "Monitoring your Artix applications".
- "Monitoring Artix applications on multiple hosts".
- "Monitoring multiple Artix applications on the same host".

Monitoring your Artix applications

You must add configuration settings to your Artix server configuration files.

C++ and JAX-RPC applications

For C++ and JAX-RPC applications, add the following example configuration settings to your Artix application's `.cfg` file:

```
// my_app.cfg

my_application {

# Ensure that it_response_time_collector is in your orb_plugins list.
orb_plugins = [ ...,"it_response_time_collector"];

# Enable performance logging.
use_performance_logging = true;

# Collector period (in seconds). How often performance information is logged.
plugins:it_response_time_collector:period = "60";

# Set the name of the file which holds the performance log
plugins:it_response_time_collector:filename =
   "/opt/myapplication/log/myapplication_perf.log"

};
```

Note: The specified `plugins:it_response_time_collector:period` should divide evenly into your cycle time (for example, a `period` of `20` and a cycle time of `60`).

JAX-WS applications

For JAX-WS applications, add the following example settings to your Artix application's Spring-based `.xml` configuration file:

```
// managed_spring_server.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- -->
<!-- Copyright (c) 1993-2007 IONA Technologies PLC. -->
<!--            All Rights Reserved. -->
<!-- -->
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:im="http://cxf.apache.org/management"
       xsi:schemaLocation="http://www.springframework.org/schema/beans/
   http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- JMX InstrumetationManager settings -->
    <bean id="InstrumentationManager"
   class="org.apache.cxf.management.jmx.InstrumentationManagerImpl">
        <property name="bus" ref="cxf" />
        <property name="enabled" value="true" />
        <property name="JMXServiceURL"
   value="service:jmx:rmi:///jndi/rmi://localhost:9914/jmxrmi" />
    </bean>

    <!-- Wiring the counter repository -->
    <bean id="CounterRepository"
   class="org.apache.cxf.management.counters.CounterRepository">
        <property name="bus" ref="cxf" />
    </bean>

    <!-- BMC counter monitor setting for writing the performance log file-->
    <bean id="BMCCounterMonitor"
   class="com.iona.cxf.management.bmc.counters.BMCCounterMonitor">
    <property name="bus" ref="cxf" />
    <property name="serverID" value="management-bmc-patrol-demo-server" />
    <property name="fileName" value="BMCCounterServer.log" />
    <property name="granularityPeriod" value="30" />
    </bean>
</beans>
```

The performance log file location is specified in the `servers.conf` configuration file (see "Creating a servers.conf file" on page 18).

For a detailed JAX-WS sample application, see the following Artix demo:

```
ArtixInstallDir\Version\java\samples\advanced\management\bmc-patrol
```

Monitoring Artix applications on multiple hosts

To monitor your Artix applications on multiple hosts, you must distribute the IONA KM to your hosts. The best approach to distributing the IONA Knowledge Module to a large number of machines is to use the Knowledge Module Distribution Service (KMDS).

Using the KMDS to distribute the IONA KM

To create a deployment set for machines that run Patrol Agents (but not the Patrol Console), perform the following steps:

1. Choose a machine with the Patrol Developer Console installed. Follow the procedure for installing the IONA KM on this machine (see "Setting up your BMC Patrol Environment" on page 28).
2. Start the Patrol Developer Console and choose **Edit Package** from the list of menu Items.
3. Open the following file:

```
$PATROL_HOME/archives/IONA_Server_KM_Agent_Resources.pkg file
```

   You will see a list of all the files that need to be installed on machines that run the Patrol Agent.
4. Now select **Check In Package** from the **File** menu to check the package into the KMDS.
5. You can now use the KMDS Manager to create a deployment set based on this KM package, and distribute it to all the machines that have IONA software installed and that also have a Patrol Agent.
6. You repeat this process for the `IONA_Server_KM_Console_Resources.pkg` file.

This creates a deployment set for all machines that have both the Patrol Agent and Patrol Console installed, and which will be used to monitor IONA software.

For further details about using the KMDS, see your BMC Patrol documentation.

Monitoring multiple Artix applications on the same host

Sometimes you may need to deploy multiple Artix applications on the same host. The solution is simply to merge the `servers.conf` and `server_commands.txt` files from each of the applications into single `servers.conf` and `server_commands.txt` files.

For example, if the `servers.conf` file from the `UnderwriterCalc` application looks as follows:

```
UnderwriterCalc,1,/opt/myAppUnderwritierCalc/log/UnderwriterCalc_perf.log
```

And the `servers.conf` file for the `ManagePolicy` application looks as follows:

```
ManagePolicy, 1, /opt/ManagePolicyApp/log/ManagePolicy_perf.log
```

The merged `servers.conf` file will then include the following two lines:

```
UnderwriterCalc,1,/opt/myAppUnderwritierCalc/log/UnderwriterCalc_perf.log
ManagePolicy, 1, /opt/ManagePolicyApp/log/ManagePolicy_perf.log
```

You can now copy this merged file to your `$PATROL_HOME/lib/iona/conf` directory and BMC Patrol will monitor both applications.

Exactly the same procedure applies to the `server_commands.txt` file.

Further information

For more detailed information on the BMC Patrol consoles, see your BMC Patrol documentation.

Configuring an Orbix Production Environment

Overview

This section describes the steps that you need to take when extending the IONA BMC Patrol integration from a test environment to a production environment. It includes the following sections:

- "Monitoring your own Orbix applications".
- "Monitoring Orbix servers on multiple hosts".
- "Monitoring multiple Orbix domains on the same host".

Monitoring your own Orbix applications

You can use the **Orbix Configuration** tool to enable BMC Patrol management of Orbix services. However, enabling BMC Patrol to manage your own applications involves the following steps:

1. You must configure your application to use performance logging (see the *Orbix Management User's Guide* for a full description).

   For example, suppose you have a server executable named `myapplication_prdserver` that executes with the ORB name `myapplication.prdserver`. The typical configuration for C++ and Java applications is as follows:

C++ applications

```
myapplication {
  prdserver {
     binding:server_binding_list = ["it_response_time_logger+OTS", ""];
         plugins:it_response_time_collector:period = "30";
         plugins:it_response_time_collector:server-id =
        "myapplication_prdserver";
         plugins:it_response_time_collector:filename =
        "/opt/myapplication/logs/prdserver/prdserver_perf.log";
       }
     }
```

Java applications

```
myapplication {
  prdserver {
    binding:server_binding_list = ["it_response_time_logger+OTS", ""];
    plugins:it_response_time_collector:period = "30";
    plugins:it_response_time_collector:server-id = "myapplication_prdserver";
    plugins:it_response_time_collector:log_properties = ["log4j.rootCategory=INFO, A1",
       "log4j.appender.A1=com.iona.management.logging.log4jappender.TimeBasedRollingFile
       Appender",
    "log4j.appender.A1.File=/opt/myapplications/logs/prdserver_perf.log",
    "log4j.appender.A1.layout=org.apache.log4j.PatternLayout",
    "log4j.appender.A1.layout.ConversionPattern=%d{ISO8601} %-80m %n"];
    }
}
```

Note: The specified `plugins:it_response_time_collector:period` should divide evenly into your cycle time (for example, a `period` of `20` and a cycle time of `60`).

2. The most important configuration values are the `server-id` and the C++ `filename` or Java `log_properties` used by the `response_time_collector`. You can add these values to the `servers.conf` file to make BMC Patrol aware of your application as follows:

```
myapplication_prdserver, 1,
   /opt/myapplication/logs/prdserver/prdserver_perf.log
```

3. To control the `myapplication_prdserver` server through the `server_command` task, edit the `server_commands.txt` file. For example you could add the following entries to `server_commands.txt`:

```
myapplication_prdserver,start =
   /opt/myapplication/scripts/prdserver_start.sh
myapplication_prdserver,stop =
   /opt/myapplication/scripts/prdserver_stop.sh
myapplication_prdserver,restart =
   /opt/myapplication/scripts/prdserver_restart.sh
```

The `prdserver_start.sh`, `prdserver_stop.sh` and `prdserver_restart.sh` scripts will be written by you.

Monitoring Orbix servers on multiple hosts

To monitor your Orbix servers on multiple hosts, you must distribute the IONA KM to your hosts. The best approach to distributing the IONA Knowledge Module to a large number of machines is to use the Knowledge Module Distribution Service (KMDS).

Using the KMDS to distribute the IONA KM

To create a deployment set for machines that run Patrol Agents (but not the Patrol Console), perform the following steps:

1. Choose a machine with the Patrol Developer Console installed. Follow the procedure for installing the IONA KM on this machine (see "Setting up your BMC Patrol Environment" on page 28).
2. Start the Patrol Developer Console and choose **Edit Package** from the list of menu Items.
3. Open the following file:

```
$PATROL_HOME/archives/IONA_Server_KM_Agent_Resources.pkg file
```

   You will see a list of all the files that need to be installed on machines that run the Patrol Agent.
4. Now select **Check In Package** from the **File** menu to check the package into the KMDS.
5. You can now use the KMDS Manager to create a deployment set based on this KM package, and distribute it to all the machines that have IONA software installed and that also have a Patrol Agent.
6. You repeat this process for the `IONA_Server_KM_Console_Resources.pkg` file.

This creates a deployment set for all machines that have both the Patrol Agent and Patrol Console installed, and which will be used to monitor IONA software.

For further details about using the KMDS, see the BMC Patrol documentation.

Monitoring multiple Orbix domains on the same host

You may have more than one Orbix configuration domain running on the same host. However, BMC Patrol is not aware of concepts like Orbix configuration domains. The current solution is to have the BMC Patrol perform monitoring of all domains on the same host. This means having only one `servers.conf` or `server_commands.txt` file for each host.

This could potentially cause problems if you have servers on the same host that have the same ORB name and by extension the same default value for the following variable:

`plugins:it_response_time_collector:server-id`

This is why, by default, the server IDs are generated with the domain name added as prefix and the host name added as suffix (for example, `mydomain_locator_myhost`).

A typical `servers.conf` file might look as follows:

```
mydomain_locator, 1,
/opt/iona/var/domains/mydomain/logs/locator_myhost_perf.log
...
yourdomain_locator, 1,
/opt/iona/var/domains/yourdomain/logs/locator_yourhost_perf.log
```

Similarly for the task library:

```
mydomain_locator_myhost , start,
/opt/iona/etc/bin/mydomain_locator_start.sh
...
yourdomain.locator_myhost , start,
/opt/iona/etc/bin/yourdomain_locator_start.sh
```

Further information

For more detailed information on the BMC Patrol Console, see you BMC Patrol documentation.

Index

A
alarms 11, 13, 34
Artix C++ runtime 13
Artix Java runtime 13

B
binding:server_binding_list 42, 43
BMC Console 29
BMC Patrol Agent 29

C
C++ configuration 38, 42
C++ runtime 13
Check In Package 40, 44
collector 31
commands 35
Customization tab 31
cycle time 38, 43

D
diagnostics 35
Domain Settings 22

E
Edit Package 40, 44
EMS 10
Enterprise Management System 10

F
File menu 40, 44
filename 43

G
Generate EMS Configuration Files 22

I
IDL, interface 31
IONAAvgResponseTime 16, 30, 32, 33, 34
IONA_km.tgz 28
IONA_km.zip 18, 28
IONAMaxResponseTime 16, 30, 32
IONAMinResponseTime 16, 30, 32
IONANumInvocations 16, 30, 31, 32
IONA_OPERATIONPROVIDER 14, 29
IONAOpsPerHour 16, 30, 32
IONA_SERVER.kml 29
IONA_Server_KM_Agent_Resources.pkg 40, 44
IONA_Server_KM_Console_Resources.pkg 40, 44
IONA_SERVERPROVIDER 14, 29, 30
itconfigure tool 22
it_response_time_collector 38
it_response_time_logger 42, 43

J
Java, requirements 29
Java configuration 43
JAX-RPC 7, 13
JAX-RPC configuration 38
JAX-WS 7, 13
JAX-WS configuration 39
JMX 13

K
KMDS 40, 44
Knowledge Module Distribution Service 40, 44
Knowledge Modules 16

L
Launch Service on Domain Startup 23
Load KMs dialog 29
log file interpreter 13
logging period 31
Log Levels 35
log_properties 43

M
Main Map 29, 35
menu commands 13, 35

O
operation

parameters 32
WSDL 31
Orbix Configuration tool 22, 42
orb_plugins 38
Other Properties 23
Override All Instances checkbox 31

P

parameter collector 31
parameters 30, 32
Patrol Agents 40, 44
PATROL Central 31
Patrol Developer Console 40, 44
performance log files 18
performance logging
configuration 23
period 31
plugins 13
plugins:it_response_time_collector:filename 38, 42
plugins:it_response_time_collector:log_properties 43
plugins:it_response_time_collector:period 38, 42, 43
plugins:it_response_time_collector:server-id 42, 43, 45
port, WSDL 31
Properties 31
Properties menu 31

R

response_time_collector 43
response times 11
Restart 35

S

server_commands.txt 19, 25, 41, 43, 45
server_command task 43
server-id 43
server parameters 30
servers.conf 18, 24, 41, 45
Service Launching 23
Set to Debug 35
Set to Error 35
Set to Info 35
Start 35
Stop 35
System Output Window 35

T

troubleshooting 35

U

UNIX 28
use_performance_logging 38

W

warnings 34
Windows 28
WSDL
operation 31
port 31