Artix™

BMC Patrol Integration Guide

Version 4.1, September 2006

Making Software Work Together™

Updated: September 22, 2006

Contents

List of Figures

Preface

What is covered in this book

IONA's products support integration with Enterprise Management Systems such as IBM Tivoli™, BMC Patrol™, CA WSDM™, and HP OpenView™. This guide explains how to integrate Artix and Orbix with BMC Patrol.

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 Library

The Artix documentation library is organized in the following sections:

- Getting Started
- Designing Artix Solutions
- Configuring and Managing Artix Solutions
- Using Artix Services
- Integrating Artix Solutions
- Integrating with Management Systems
- Reference
- Artix Orchestration

Getting Started

The books in this section provide you with a background for working with Artix. They describe many of the concepts and technologies used by Artix. They include:

- Release Notes contains release-specific information about Artix.
- Installation Guide describes the prerequisites for installing Artix and the procedures for installing Artix on supported systems.
- Getting Started with Artix describes basic Artix and WSDL concepts.
- Using Artix Designer describes how to use Artix Designer to build Artix solutions.
- Artix Technical Use Cases provides a number of step-by-step examples of building common Artix solutions.

Designing Artix Solutions

The books in this section go into greater depth about using Artix to solve real-world problems. They describe how to build service-oriented architectures with Artix and how Artix uses WSDL to define services:

- Building Service-Oriented Infrastructures with Artix provides an overview of service-oriented architectures and describes how they can be implemented using Artix.
- Writing Artix Contracts describes the components of an Artix contract. Special attention is paid to the WSDL extensions used to define Artix-specific payload formats and transports.

Developing Artix Solutions

The books in this section how to use the Artix APIs to build new services:

- Developing Artix Applications in C++ discusses the technical aspects of programming applications using the C++ API.
- Developing Advanced Artix Plug-ins in C++ discusses the technical aspects of implementing advanced plug-ins (for example, interceptors) using the C++ API.
- Developing Artix Applications in Java discusses the technical aspects of programming applications using the Java API.

Configuring and Managing Artix Solutions

This section includes:

- Configuring and Deploying Artix Solutions explains how to set up your Artix environment and how to configure and deploy Artix services.
- Managing Artix Solutions with JMX explains how to monitor and manage an Artix runtime using Java Management Extensions.

Using Artix Services

The books in this section describe how to use the services provided with Artix:

- Artix Router Guide explains how to integrate services using the Artix router.
- Artix Locator Guide explains how clients can find services using the Artix locator.
- Artix Session Manager Guide explains how to manage client sessions using the Artix session manager.
- Artix Transactions Guide, C++ explains how to enable Artix C++ applications to participate in transacted operations.
- Artix Transactions Guide, Java explains how to enable Artix Java applications to participate in transacted operations.
- Artix Security Guide explains how to use the security features in Artix.

Integrating Artix Solutions

The books in this section describe how to integrate Artix solutions with other middleware technologies.

- Artix for CORBA provides information on using Artix in a CORBA environment.
- Artix for J2EE provides information on using Artix to integrate with J2EE applications.

For details on integrating with Microsoft's .NET technology, see the documentation for Artix Connect.

Integrating with Management Systems

The books in this section describe how to integrate Artix solutions with a range of enterprise and SOA management systems. They include:

- IBM Tivoli Integration Guide explains how to integrate Artix with the IBM Tivoli enterprise management system.
- BMC Patrol Integration Guide explains how to integrate Artix with the BMC Patrol enterprise management system.
- CA-WSDM Integration Guide explains how to integrate Artix with the CA-WSDM SOA management system.
- AmberPoint Integration Guide explains how to integrate Artix with the AmberPoint SOA management system.

Reference

These books provide detailed reference information about specific Artix APIs, WSDL extensions, configuration variables, command-line tools, and terms. The reference documentation includes:

- Artix Command Line Reference
- Artix Configuration Reference
- Artix WSDL Extension Reference
- Artix Java API Reference
- Artix C++ API Reference
- Artix .NET API Reference
- Artix Glossary

Artix Orchestration

These books describe the Artix support for Business Process Execution Language (BPEL), which is available as an add-on to Artix. These books include:

- Artix Orchestration Release Notes
- Artix Orchestration Installation Guide
- Artix Orchestration Administration Console Help

Getting the Latest Version

The latest updates to the Artix documentation can be found at http://www.iona.com/support/docs.

Compare the version dates on the web page for your product version with the date printed on the copyright page of the PDF edition of the book you are reading.

Searching the Artix Library

You can search the online documentation by using the **Search** box at the top right of the documentation home page:

http://www.iona.com/support/docs

To search a particular library version, browse to the required index page, and use the **Search** box at the top right, for example:

http://www.iona.com/support/docs/artix/4.0/index.xml

You can also search within a particular book. To search within a HTML version of a book, use the **Search** box at the top left of the page. To search within a PDF version of a book, in Adobe Acrobat, select **Edit**|**Find**, and enter your search text.

Artix Online Help

Artix Designer and Artix Orchestration Designer include comprehensive online help, providing:

- Step-by-step instructions on how to perform important tasks
- A full search feature
- Context-sensitive help for each screen

There are two ways that you can access the online help:

- Select **Help|Help Contents** from the menu bar. The help appears in the contents panel of the Eclipse help browser.
- Press **F1** for context-sensitive help.

In addition, there are a number of cheat sheets that guide you through the most important functionality in Artix Designer and Artix Orchestration Designer. To access these, select **Help|Cheat Sheets**.

Artix Glossary

The Artix Glossary is a comprehensive reference of Artix terms. It provides quick definitions of the main Artix components and concepts. All terms are defined in the context of the development and deployment of Web services using Artix.

Additional Resources

The IONA Knowledge Base (http://www.iona.com/support/knowledge_base/index.xml) contains helpful articles written by IONA experts about Artix and other products.

The IONA Update Center (http://www.iona.com/support/updates/index.xml) contains the latest releases and patches for IONA products.

If you need help with this or any other IONA product, go to IONA Online Support (http://www.iona.com/support/index.xml).

Comments, corrections, and suggestions on IONA documentation can be sent to docs-support@iona.com .

Document Conventions

Typographical conventions

This book uses the following typographical conventions:

`Fixed width`

Fixed width (courier font) in normal text represents portions of code and literal names of items such as classes, functions, variables, and data structures. For example, text might refer to the `IT_Bus::AnyType` class.

Constant width paragraphs represent code examples or information a system displays on the screen. For example:

```
#include <stdio.h>
```

Fixed width italic	Fixed width italic words or characters in code and commands represent variable values you must supply, such as arguments to commands or path names for your particular system. For example: `% cd /users/`*`YourUserName`*
Italic	Italic words in normal text represent *emphasis* and introduce *new terms*.
Bold	Bold words in normal text represent graphical user interface components such as menu commands and dialog boxes. For example: the **User Preferences** dialog.

Keying Conventions

This book uses the following keying conventions:

No prompt	When a command's format is the same for multiple platforms, the command prompt is not shown.
`%`	A percent sign represents the UNIX command shell prompt for a command that does not require root privileges.
`#`	A number sign represents the UNIX command shell prompt for a command that requires root privileges.
`>`	The notation > represents the MS-DOS or Windows command prompt.
. . . . . .	Horizontal or vertical ellipses in format and syntax descriptions indicate that material has been eliminated to simplify a discussion.
`[]`	Brackets enclose optional items in format and syntax descriptions.
`{}`	Braces enclose a list from which you must choose an item in format and syntax descriptions.
`|`	In format and syntax descriptions, a vertical bar separates items in a list of choices enclosed in `{}` (braces). In graphical user interface descriptions, a vertical bar separates menu commands (for example, select **File**|**Open**).

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

Overview

IONA's 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™, IBM Tivoli™, and HP OpenView™. 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 Software Knowledge Module design and operation. Figure 1 on page 18 shows a simplified view of how the IONA Knowledge Modules work. In this example, an alarm is triggered when the locator becomes unresponsive, and this results in an action to restart the server.

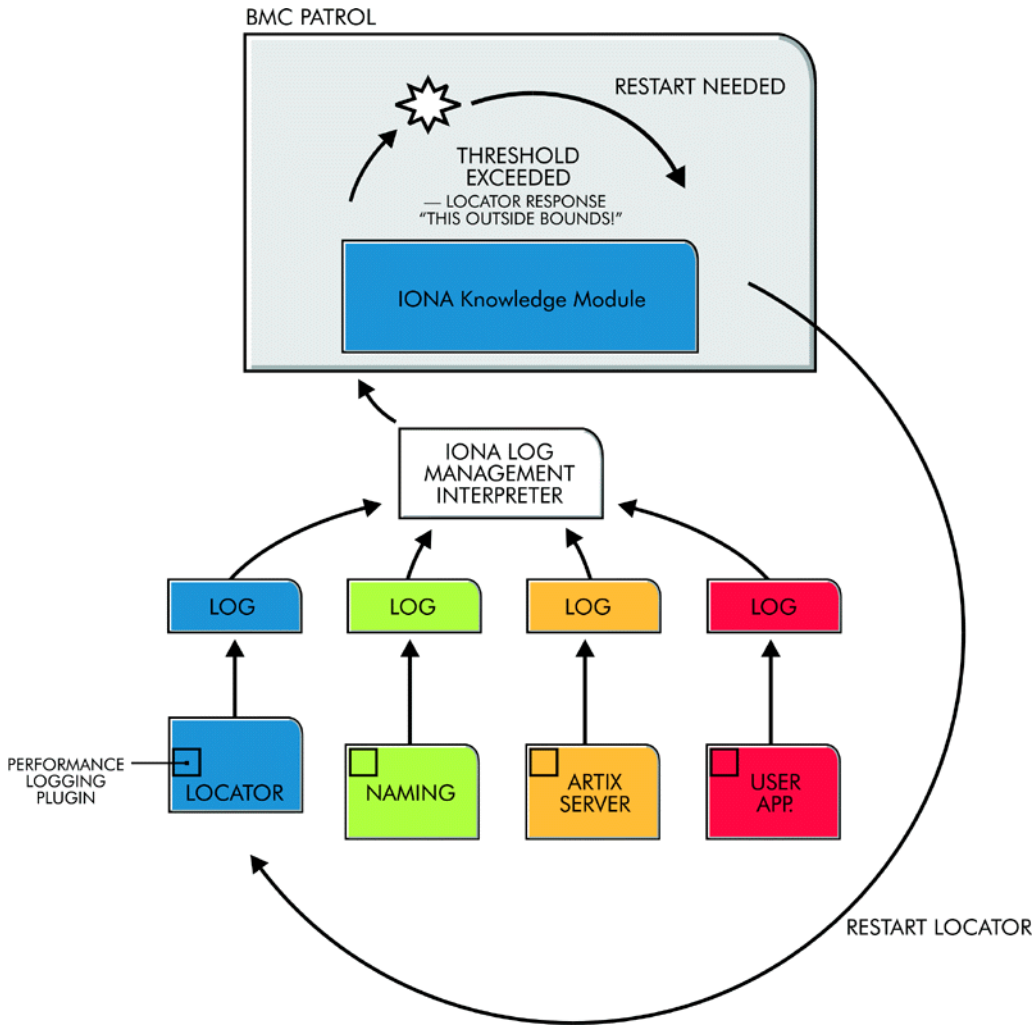

Figure 1: *Overview of the IONA BMC Patrol Integration*

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, and uses the IONA log file interpreter to collect and summarize the logged data.

The IONA Knowledge Module compares the response times and other values against the defined alarm ranges for each parameter 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 menu command to stop, start or restart the offending server.

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".
- "Examples".
- "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`

The `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.

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

Examples

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`.

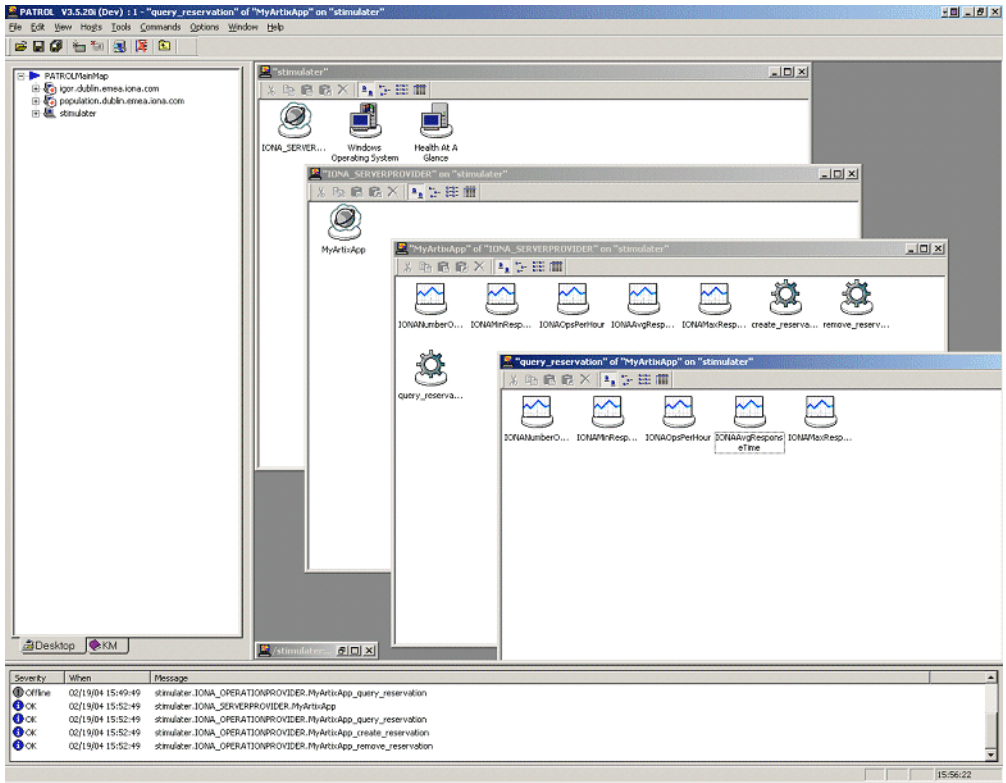

Figure 2: *IONA Server Running in BMC Patrol*

The IONA server performance metrics include the following:

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

For more details, see "Using the IONA Knowledge Module" on page 36.

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.

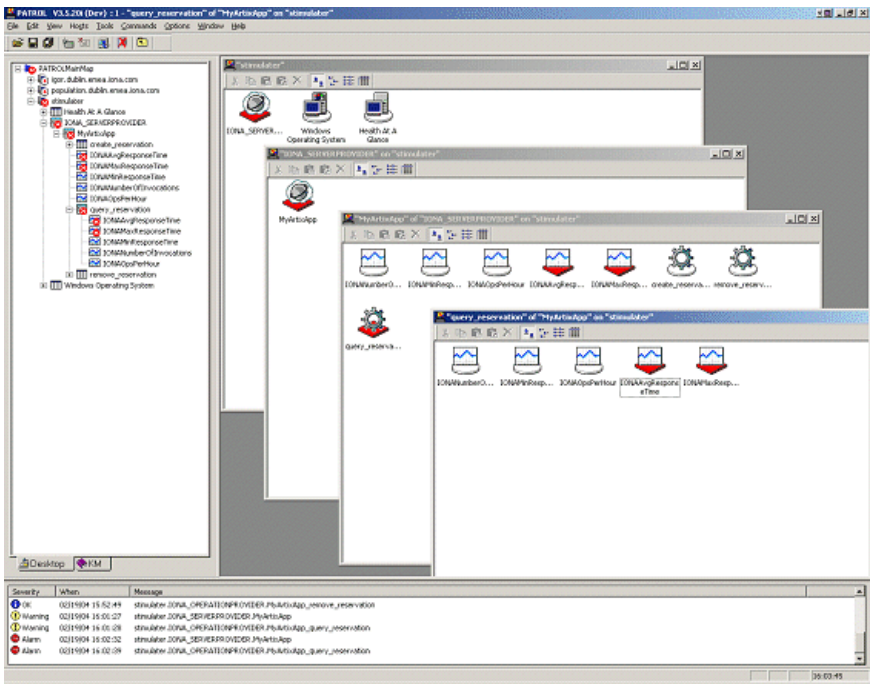

Figure 3: *BMC Patrol Displaying Alarms*

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

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:

- "Enabling management".
- "Generating EMS configuration files".
- "The servers.conf file".
- "The server_commands.txt file".
- "Further information".

Enabling management

You can use the **Artix Designer** GUI tool to enable management for your Artix applications. The **Artix Tools** dialog shown in Figure 4 allows you to do this.

Note: Before enabling management, you must have first generated a service plug-in (see Using Artix Designer).

To enable management, perform the following steps

1. Select **Artix Designer**|**Artix Tools**|**Artix Tools**.
2. In the **Artix Tools** window, create a new container deployment launch configuration.
3. In the **Advanced QoS Options** tab, select the **Enable reporting to a third-party management application** checkbox.
4. Enter an **Output file location**.

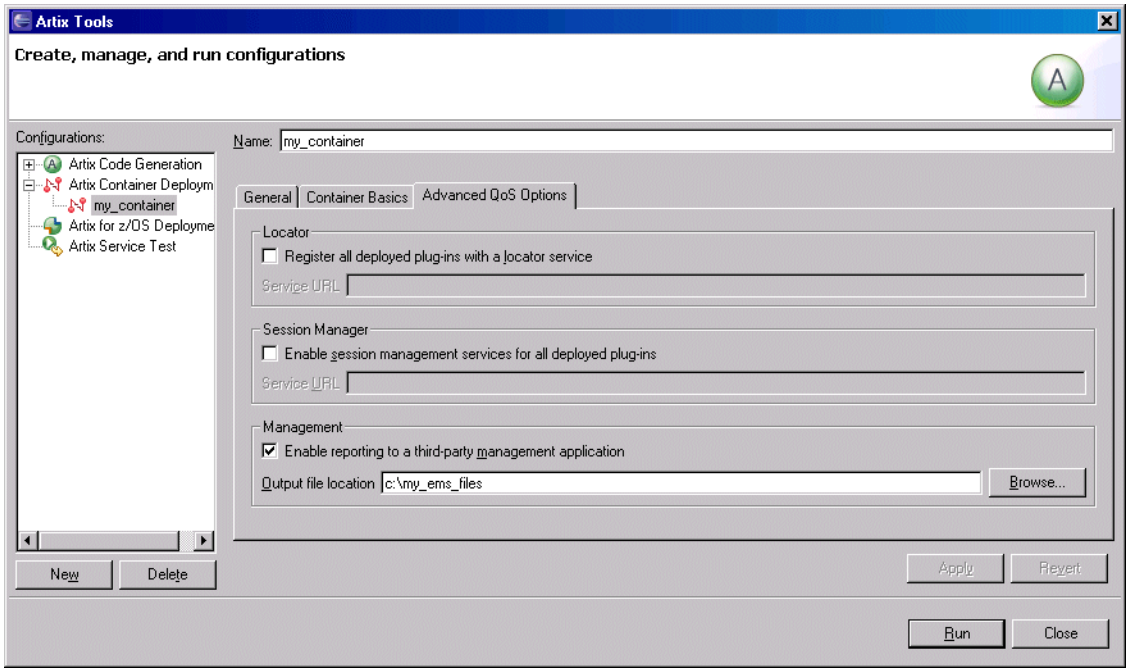

Figure 4: *Enabling Management in Artix Designer*

Generating EMS configuration files

When you click **Run**, this generates two files that are used to configure the BMC Patrol integration:

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

These files are generated in the **Output file location** specified in Figure 4.

To track your application in BMC Patrol, you must copy these files into your BMC installation, for example:

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

For more information on using Artix GUI tools, see Using Artix Designer.

The servers.conf file

When you open the `servers.conf` file, you will see an entry such as the following:

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

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

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

The server_commands.txt file

When you open the `server_commands.txt` file, you will see entries like the following:

```
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
```

Each entry in this file references a script that can be used to stop, start, or restart the `myapplication` server.

Further information

For details of how to manually configure servers to use the performance logging, see "Setting up your BMC Patrol Environment" on page 34.

For a complete explanation of configuring performance logging plugins, 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".
- "The servers.conf file".
- "The server_commands.txt file".
- "Further information".

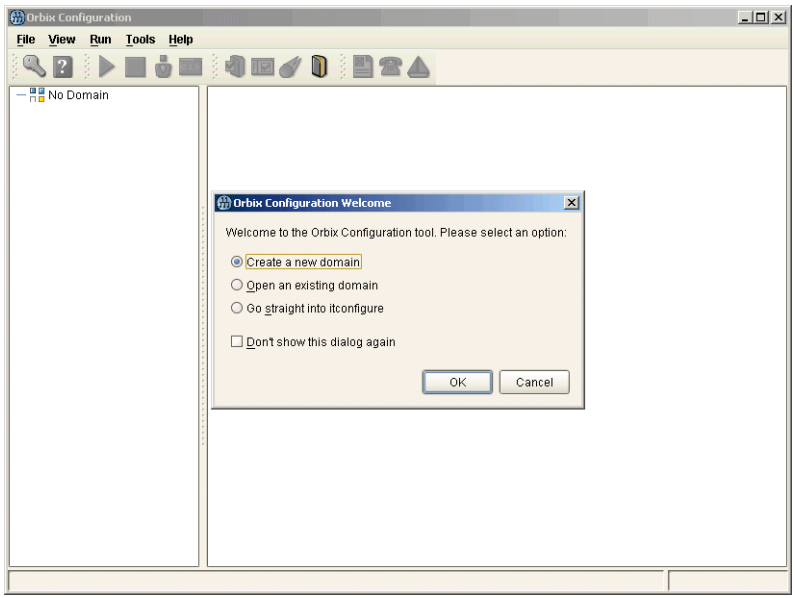

Figure 5: *Orbix Configuration GUI*

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 5.

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 6.
2. Select the **Generate EMS Configuration Files** checkbox. This generates the configuration files required for your BMC Patrol integration.

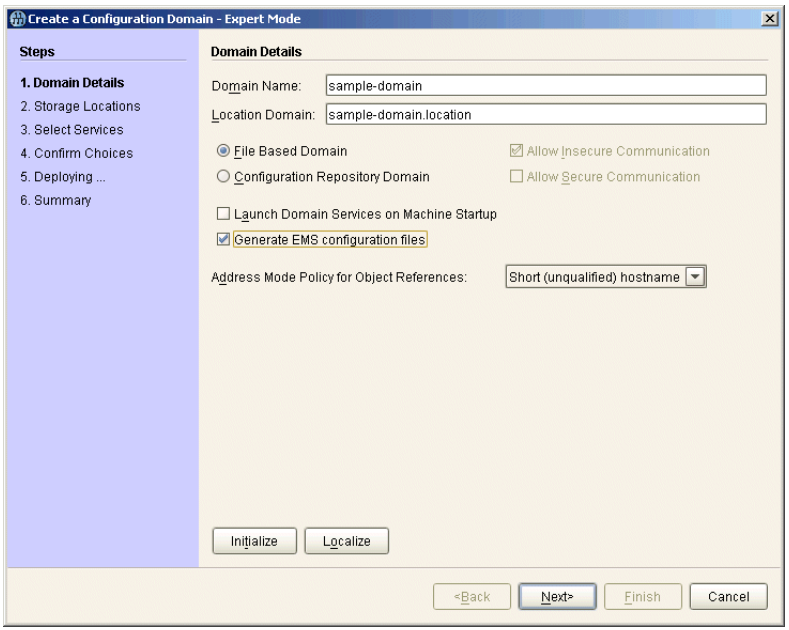

Figure 6: *Selecting EMS Configuration*

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 7.
2. Select the **Performance Logging** option in the **Other Properties** box, shown in Figure 7. 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.

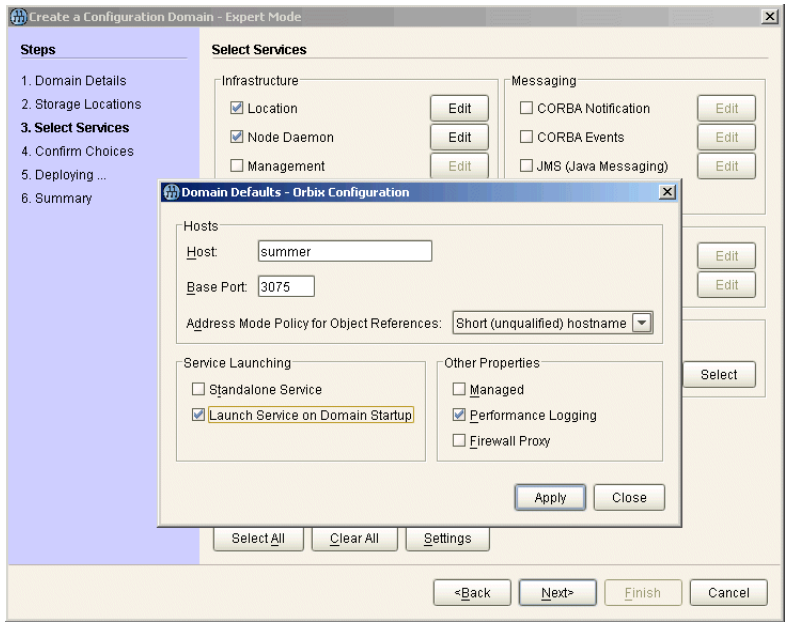

Figure 7: *Selecting Performance Logging*

   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

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	*ArtixInstall*\artix*Version*\management\BMC\IONA_km.zip
UNIX	*ArtixInstall*/artix/*Version*/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 8 shows BMC Patrol graphing the value of the `IONAAvgResponseTime` parameter on a `query_reservation` operation call.

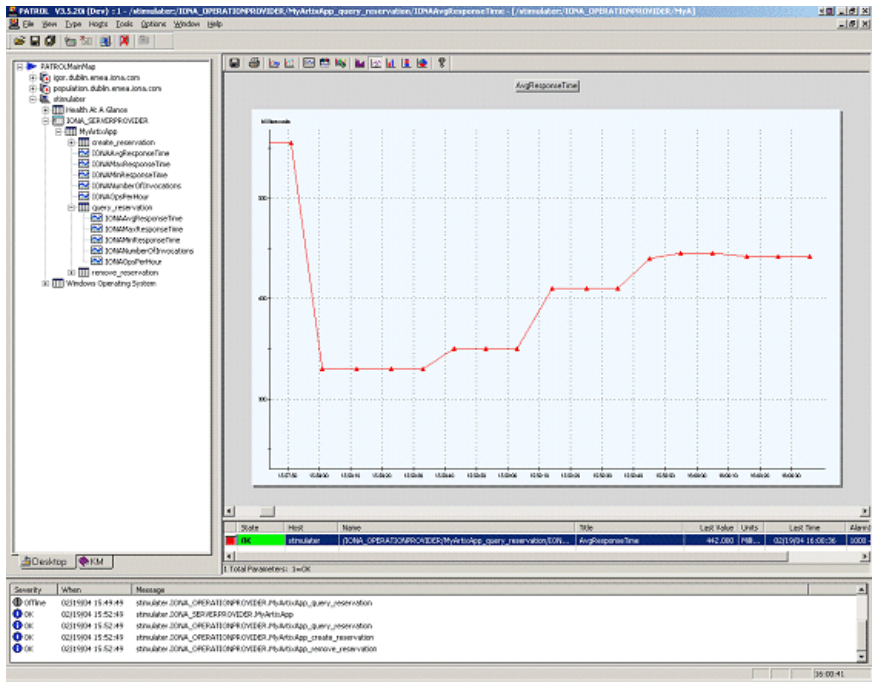

Figure 8: *Graphing for IONAAvgResponseTime*

Figure 9 shows warnings and alarms issued for the `IONAAvgResponseTime` parameter.

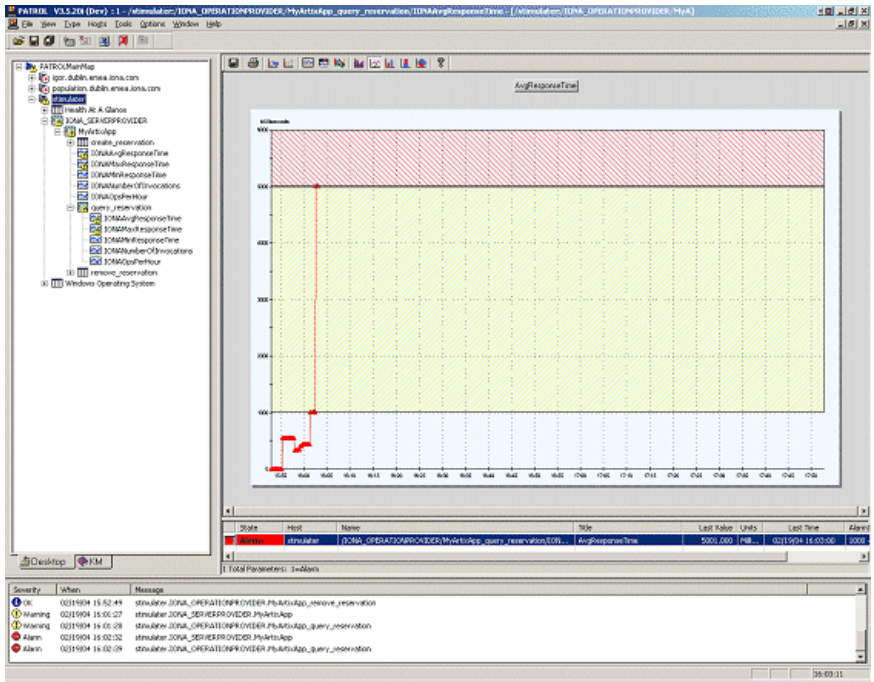

Figure 9: *Alarms for IONAAvgResponseTime*

Starting, stopping and restarting servers

The **Orbix Configuration** and **Artix Designer** GUIs will generate a `server_commands.txt` for the services that you are deploying on your host. 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

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

Monitoring your own Artix applications

Using the **Artix Designer** GUI to enable BMC Patrol to manage your applications is straightforward. For for details, see "Setting up your Artix Environment" on page 24.

Manual configuration

If you do not use **Artix Designer**, you must manually add the following settings to your Artix server's configuration file:

```
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`).

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 34).
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. However, the **Artix Designer** only generates a `servers.conf` and `server_commands.txt` file for a single application.

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 you 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 34).
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

R

S

T

U

W