

OPTIMIZE

MERCURY DIAGNOSTICS™ FOR J2EE, .NET & ERP/CRM

VERSION 6.5

Installation and Configuration Guide

MERCURY™

BUSINESS TECHNOLOGY OPTIMIZATION

Mercury Diagnostics

Installation and Configuration Guide

Version 6.5

Document Release Date: December 21, 2006

MERCURY™

Mercury Diagnostics Installation and Configuration Guide Version 6.5

This document, and the accompanying software and other documentation, is protected by U.S. and international copyright laws, and may be used only in accordance with the accompanying license agreement. Features of the software, and of other products and services of Mercury Interactive Corporation, may be covered by one or more of the following patents: United States: 5,511,185; 5,657,438; 5,701,139; 5,870,559; 5,958,008; 5,974,572; 6,137,782; 6,138,157; 6,144,962; 6,205,122; 6,237,006; 6,341,310; 6,360,332, 6,449,739; 6,470,383; 6,477,483; 6,549,944; 6,560,564; 6,564,342; 6,587,969; 6,631,408; 6,631,411; 6,633,912; 6,694,288; 6,738,813; 6,738,933; 6,754,701; 6,792,460 and 6,810,494. Australia: 763468 and 762554. Other patents pending. All rights reserved.

U.S. GOVERNMENT RESTRICTED RIGHTS. This Software Documentation is a “commercial item” as defined at 48 C.F.R. 2.101 (October 1995). In accordance with 48 C.F.R. 12.212 (October 1995), 48 C.F.R. 27.401 through 27.404 and 52.227-14 (June 1987, as amended) and 48 C.F.R. 227.7201 through 227.7204 (June 1995), and any similar provisions in the supplements to Title 48 of the C.F.R. (the “Federal Acquisition Regulation”) of other entities of the U.S. Government, as applicable, all U.S. Government users acquire and may use this Documentation only in accordance with the restricted rights set forth in the license agreement applicable to the Computer Software to which this Documentation relates.

Mercury, Mercury Interactive, the Mercury logo, the Mercury Interactive logo, LoadRunner, WinRunner, SiteScope and TestDirector are trademarks of Mercury Interactive Corporation and may be registered in certain jurisdictions. The absence of a trademark from this list does not constitute a waiver of Mercury's intellectual property rights concerning that trademark.

All other company, brand and product names may be trademarks or registered trademarks of their respective holders. Mercury disclaims any responsibility for specifying which marks are owned by which companies or which organizations.

Mercury provides links to external third-party Web sites to help you find supplemental information. Site content and availability may change without notice. Mercury makes no representations or warranties whatsoever as to site content or availability.

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Fax: (650) 603-5300
<http://www.mercury.com>

© 2004 - 2006 Mercury Interactive Corporation, All rights reserved

If you have any comments or suggestions regarding this document, please send them by e-mail to documentation@mercury.com.

Table of Contents

Welcome to This Guide	xiii
How This Guide Is Organized	xiv
Mercury Diagnostics Online Documentation.....	xvi
Additional Online Resources.....	xviii
Documentation Updates	xviii
Typographical Conventions.....	xix

PART I: INTRODUCTION

Chapter 1: Preparing to Install Mercury Diagnostics	3
Mercury Diagnostics Components and Data Flow	4
Supported Application Servers and Environments	6
Host System Requirements for the Diagnostics Components	7
Information Required for Installation	11
Pre-installation Considerations.....	17
Recommended Order of Installation.....	18
Licensing Mercury Diagnostics	20
Upgrading from Earlier Versions of Diagnostics.....	20

PART II: INSTALLATION OF THE MERCURY DIAGNOSTICS SERVER

Chapter 2: Installing the Mercury Diagnostics Server.....	23
Installing the Diagnostics Server on a Windows Machine	24
Installing the Diagnostics Server on a UNIX Machine	35
Starting and Stopping the Diagnostics Server.....	46
Verifying the Diagnostics Server Installation	48
Licensing the Diagnostics Server in Commander Mode.....	49
Configuring the Diagnostics Server	49
Determining the Version of the Diagnostics Server	50
Uninstalling the Diagnostics Server	50
Troubleshooting Diagnostics Server Issues	52

Chapter 3: Licensing Mercury Diagnostics	53
About Mercury Diagnostics Licensing	53
Types of Licenses	54
Licensing the Diagnostics Server in Commander Mode.....	54

PART III: INSTALLATION AND CONFIGURATION OF THE MERCURY DIAGNOSTICS PROBES

Chapter 4: Installing the Diagnostics Collector.....	61
About Installing the Diagnostics Collector.....	62
Installing the Diagnostics Collector on a Windows Machine.....	62
Installing the Diagnostics Collector on a UNIX Machine	70
Silent Installation of the Diagnostics Collector	78
Configuring the Active System Property Files	79
Verifying the Diagnostics Collector Installation	82
Starting and Stopping the Diagnostics Collector.....	83
Determining the Version of the Diagnostics Collector	85
Uninstalling the Diagnostics Collector.....	85
Chapter 5: Installing the Mercury Diagnostics Probe for .NET.....	87
About the Mercury Diagnostics Probe for .NET.....	87
Installing the .NET Probe	88
Verifying the .NET Probe Installation	100
Configuring the .NET Probe.....	102
Determining the Version of the .NET Probe.....	102
Uninstalling the .NET Probe	102
Chapter 6: Installing the Mercury Diagnostics Probe for J2EE.....	103
About Installing and Configuring the J2EE Probe	104
Installing the J2EE Probe on a Windows Machine	105
Installing the J2EE Probe on a UNIX Machine	122
Installing the J2EE Probe on a z/OS Mainframe	137
Installing the J2EE Probe Using the Generic Installer	139
Silent Installation of the J2EE Probe	141

Chapter 7: Post-Installation Configuration of the Mercury	
Diagnostics Probe for J2EE	143
Configuring the Probe Manually	144
Configuring the Application Server	147
Configuring the Probe for Advanced Situations	148
Deploying BEA JRockit Mission Control.....	148
Verifying the J2EE Probe Installation.....	148
Determining the Version of the J2EE Probe.....	150
Upgrading to a Newer Version of the J2EE Probe	151
Uninstalling the J2EE Probe	154
Chapter 8: Running the JRE Instrumenter	155
About the JRE Instrumenter	155
Running the JRE Instrumenter.....	157
Chapter 9: Configuring the Application Servers	167
About Configuring the Application Server	168
Configuring WebSphere Application Servers.....	169
Configuring WebLogic Application Servers.....	184
Configuring the Oracle9i Application Server.....	193
Configuring the JBoss Application Server	196
Configuring the SAP NetWeaver Application Server	199
Configuring a Generic Application Server	201
Adjusting the Heap Size for the J2EE Probe in the Application Startup Script.....	203

PART IV: SETTING UP INTEGRATION WITH OTHER MERCURY PRODUCTS

Chapter 10: Setting Up Mercury Business Availability Center 6.1 or Later to Use Diagnostics	207
About Setting Up Mercury Business Availability Center to use Diagnostics	208
Specifying the Diagnostics Server Details	209
Changing the Diagnostics Server Details	213
Understanding the Diagnostics Configuration Page	213
Enabling the Diagnostics View	215
Enabling Diagnostics Monitoring in Business Process Monitor (BPM) Related Views	217
Assigning Permissions for Diagnostics Users	219
Accessing the Diagnostics Pages in Windows 2003.....	221

Chapter 11: Installing LoadRunner 8.1 and the LoadRunner Diagnostics AddIn	223
Understanding the LoadRunner Diagnostics AddIn.....	223
Installing Mercury LoadRunner	224
Installing the Mercury LoadRunner Diagnostics AddIn	224
Configuring LoadRunner for Mercury Diagnostics	226
Chapter 12: Setting Up LoadRunner 8.1 to use Mercury Diagnostics	227
About Setting Up LoadRunner to Use Mercury Diagnostics.....	227
Specifying the Diagnostics Server Details	228
Configuring LoadRunner Scenarios to use Mercury Diagnostics	230
Chapter 13: Setting Up Performance Center 8.1 to Use Mercury Diagnostics	231
About Setting Up Performance Center to Use Mercury Diagnostics	231
Specifying the Diagnostics Server Details	232
Configuring Performance Center Load Tests to Use Mercury Diagnostics	234

PART V: DIAGNOSTICS INSTRUMENTATION

Chapter 14: Instrumenting an Application	237
About Instrumentation and Capture Points Files	238
Locating the Capture Points Files	239
Coding Points in the Capture Points File	241
Defining Points With Code Snippets.....	247
Instrumentation Examples.....	257
Understanding the Overhead of Custom Instrumentation.....	266
Instrumentation Control	267
Advanced Instrumentation	268
Chapter 15: Instrumenting Java Applications From the Profiler	273
About the Configuration Tab.....	273
Maintaining Instrumentation from the Configuration Tab.....	275
Maintaining Probe Settings from the Configuration Tab.....	284
Chapter 16: Instrumentation Layers	289
About Instrumentation Layers	289
Understanding Diagnostics Layers.....	290

Chapter 17: Using Solution Templates	295
About Solution Templates	295
Understanding Solution Template Data	296
Viewing Solution Template Performance Metrics	296

PART VI: ADVANCED COMPONENT CONFIGURATION

Chapter 18: Advanced Diagnostics Server Configuration.....	309
Synchronizing Time Between Diagnostics Components.....	310
Configuring the Diagnostics Server for a Large Installation.....	314
Overriding the Default Diagnostics Server Host Name.....	317
Changing the Default Diagnostics Server Port	318
Configuring the Diagnostics Server for Multi-Homed Environments.....	319
Reducing Diagnostics Server Memory Usage	323
Configuring Fragment Name Based Trimming.....	324
Preparing a High Availability Diagnostics Server.....	325
LoadRunner / Performance Center Diagnostics Server Assignments	327
Configuring the Diagnostics Server for the LoadRunner Offline File	328
Configuring Diagnostics Using the Diagnostics Server Configuration Pages.....	331
Chapter 19: Advanced .NET Probe Configuration.....	333
Enabling the Mercury Diagnostics Probe for .NET	334
Restarting IIS.....	334
Discovering ASP.NET Applications	336
.NET Probe Automatic Configuration for Discovered ASP.NET Applications	337
Enabling and Disabling Instrumentation for Applications	338
Customizing the Instrumentation for ASP.NET Applications	341
Discovering the Classes and Methods in an Application	344
Configuring Latency Trimming and Throttling	346
Configuring Depth Trimming.....	351
Configuring the .NET Probe for Light-Weight Memory Diagnostics	352
Disabling Logging.....	354
Overriding the Default Probe Host Machine Name.....	354
Authentication and Authorization for .NET Profilers in Standalone Mode	355

Chapter 20: Understanding the .NET Probe Configuration File	357
About the .NET Probe Configuration File.....	357
Understanding the .NET Probe Configuration File.....	358
Chapter 21: Advanced J2EE Probe and Application Server Configuration	387
Advanced Configuration Directory.....	388
Configuring the Probe to Work with Mercury Products	389
Configuring the Probes for Multiple Application Server JVM Instances	393
Disabling the J2EE Diagnostics Profiler.....	397
Specifying Probe Properties as Java System Properties.....	398
Controlling Probe Logging.....	399
Setting the Probe Host Machine Name.....	400
Controlling Automatic Method Trimming on the Probe.....	402
Controlling Probe Throttling	404
Configuring a Probe for a Proxy Server.....	407
Configuring Reverse HTTP for a Probe in MMS.....	407
Diagnostics Probe Administration Page.....	409
Authentication and Authorization for J2EE Profilers in Standalone Mode	412
Using BEA JRockit Mission Control	414

PART VII: ADVANCED COMPONENT COMMUNICATION

Chapter 22: Enabling HTTPS Between Diagnostics Components	417
About Configuring HTTPS Communications	418
Enabling Incoming HTTPS Communication for Diagnostics Components.....	419
Enabling Outgoing HTTPS Communication from Diagnostics Components.....	424
Enabling HTTPS Communications for the Mercury Business Availability Center Server	428
Chapter 23: Configuring Diagnostics Components for HTTP Proxy	431
Enabling HTTP Proxy Communications for the Diagnostics Servers	432
Enabling HTTP Proxy Communications for the J2EE Probe	433
Enabling HTTP Proxy Communications for a .NET Probe	433

Chapter 24: Configuring Diagnostics Components to Work with a Firewall	435
Overview of Configuring Diagnostics for a Firewall.....	436
Collating Offline Analysis Files over a Firewall	439
Installing and Configuring the Mercury MI Listener	440
Configuring the Diagnostics Server in Mediator mode to Work with a Firewall	441
Configuring LoadRunner and Performance Center to Work with Diagnostics Firewalls	448

PART VIII: CONFIGURING DIAGNOSTICS METRICS COLLECTORS

Chapter 25: Overview of Diagnostics Metrics Collectors	453
About Metrics Capture	453
Configuring Metric Collector Entries.....	454
Chapter 26: Configuring System Metrics Capture	459
About System Metrics.....	459
System Metrics Captured by Default.....	460
Configuring System Metric Collector	461
Capturing Custom System Metrics	462
Enabling z/OS System Metrics Capture.....	468
Chapter 27: Configuring JMX Metric Capture	471
About JMX Metrics	471
Configuring WebSphere for JMX Metric Collection.....	472
Configuring JMX Metric Collectors	474
Capturing Custom JMX Metrics.....	474

PART IX: APPENDIXES

Appendix A: Diagnostics Server Administration Page	481
Accessing the Diagnostics Server Administration Page	481
Configuring Diagnostics Using the Diagnostics Server Configuration Pages.....	483

Appendix B: User Authentication and Authorization	489
About User Authentication and Authorization	490
Understanding User Privileges	491
Accessing Diagnostics Using Default User Names	492
Understanding the Diagnostics Server Permissions Page	493
Creating, Editing and Deleting Users.....	496
Assigning Privileges Across the Diagnostics Deployment	498
Assigning Privileges for Probe Groups	499
Authentication and Authorization for Users of Integrated Mercury Products	502
Tracking User Administration Activity	503
Appendix C: Using the System Health Monitor	505
Introducing the System Health Monitor	506
Accessing the System Health Monitor	506
Understanding the System Health Monitor.....	509
Viewing Component Status and Host Configuration Tooltips	514
Viewing Detailed Component Information	516
Viewing Troubleshooting Tips	520
Viewing Log Information for the Whole System.....	520
Customizing the System Health Monitor Display	521
Filtering the System Health Monitor Component Map by Customer.....	523
Controlling the Refresh Rate of the System Health Monitor	524
Creating and Using System Health Monitor Snapshots	525
Appendix D: Diagnostics Data Management	527
About Diagnostics Data.....	528
Custom View Data.....	528
Performance History Data	530
Data Storage Size & Data Retention	533
Pre-Installation Data Management Considerations.....	535
Backing Up Diagnostics Data	536
Handling Diagnostics Data when Upgrading Diagnostics	539
Appendix E: Diagnostics Component Configuration and Communication Diagrams	545
Communications with Mercury Business Availability Center.....	546
Communications with LoadRunner and Performance Center.....	547

Appendix F: Upgrading Diagnostics and Other Mercury Products..	549
Overview of Product Upgrade Paths	549
General Upgrade Recommendations	550
Diagnostics Compatibility With Earlier Diagnostics Versions	551
Upgrading Diagnostics Components to Diagnostics 6.5	552
Diagnostics Compatibility with Other Mercury Products	561
Appendix G: Troubleshooting Mercury Diagnostics	563
Component Installation Interrupted on a Solaris Machine	564
J2EE Probe Fails to Operate Properly.....	564
J2EE Probe Throttling Seems Excessive	564
Error During WAS Startup with Mercury Diagnostics	
Profiler for J2EE.....	565
Missing Server-Side Transactions	566
Appendix H: Using UNIX Commands	567
Appendix I: Using Regular Expressions.....	569
Common Regular Expression Operators	570
Combining Regular Expression Operators	576
Appendix J: Multi-Lingual User Interface Support	579
Index.....	581

Table of Contents

Welcome to This Guide

Welcome to the *Mercury Diagnostics Installation and Configuration Guide*. This guide describes how to install and configure the Mercury Diagnostics components. This guide also explains how to set up other Mercury products for integration with Mercury Diagnostics.

This chapter describes:	On page:
How This Guide Is Organized	xiv
Mercury Diagnostics Online Documentation	xvi
Additional Online Resources	xviii
Documentation Updates	xviii
Typographical Conventions	xix

How This Guide Is Organized

This guide contains the following parts:

Part I Introduction

Provides the information and instructions to plan and prepare for the installation and configuration of the Diagnostics components.

Part II Installation of the Mercury Diagnostics Server

Describes how to install and configure the Mercury Diagnostics Server.

Part III Installation and Configuration of the Mercury Diagnostics Probes

Describes the processes for installing and configuring the Mercury Diagnostics Probes.

Part IV Setting Up Integration with Other Mercury Products

Describes the necessary steps to set up LoadRunner, Performance Center, and Mercury Business Availability Center for integration with Mercury Diagnostics.

Part V Diagnostics Instrumentation

Describes how to control the instrumentation that Mercury Diagnostics applies to the classes and methods of your applications to enable it to gather the performance metrics.

Part VI Advanced Component Configuration

Describes advanced configuration of the Mercury Diagnostics Server, and the Mercury .NET and J2EE Probes.

Part VII Advanced Component Communication

Describes how to set up your Mercury Diagnostics platform using different communication channels.

Part VIII Configuring Diagnostics Metrics Collectors

Describes metrics capture and how to configure the metric collectors.

Part IX Appendixes

Describes more detailed and advanced information about selected topics.

Mercury Diagnostics Online Documentation

Your Mercury Diagnostics application comes with the following documentation:

- ▶ **Mercury Diagnostics User's Guide.** Explains how to use Mercury Diagnostics to analyze the performance of your enterprise applications. You access this guide online from the **Help** button in Diagnostics or from the help menu in the integrated Mercury product. You access the PDF version of this guide from the Windows Start menu (**Start > Programs > Mercury Diagnostics Server > User's Guide**), from the **Docs** directory on the Mercury Diagnostics installation CD, or from the Diagnostics Server installation directory.
- ▶ **Mercury Diagnostics Installation and Configuration Guide.** Explains how to install and configure the Mercury Diagnostics components. You access this guide online from the **Help** button in Diagnostics or from the help menu in the integrated Mercury product. You access the PDF version of this guide from the Windows Start menu (**Start > Programs > Mercury Diagnostics Server > Installation Guide**), from the **Docs** directory on the Mercury Diagnostics installation CD, or from the Diagnostics Server installation directory.
- ▶ **Readme.** Provides last-minute technical and troubleshooting information about Mercury Diagnostics. The file is located in the Mercury Diagnostics installation CD root directory.
- ▶ **Mercury Diagnostics Probe for J2EE Installation Quick Start.** Provides the basic instructions for installing the Mercury Diagnostics Probe for J2EE and is available in the **Docs** directory on the Probe installation CD or from the Windows Start menu (**Start > Programs > Mercury Diagnostics J2EE Probe > QuickStart**).
- ▶ **Mercury Diagnostics Profiler for J2EE Installation and User's Guide.** Describes how to install, configure and use the Mercury Diagnostics Profiler for J2EE. You access this guide online by clicking **Help** in the Mercury Diagnostics Profiler for J2EE. You access the PDF version of this guide from the **Docs** directory on the Probe installation CD.

- ▶ **Mercury Diagnostics Profiler for .NET Installation and User's Guide.**
Describes how to install, configure and use the Mercury Diagnostics Profiler for .NET. You access this guide online by clicking **Help** in the Mercury Diagnostics Profiler for NET. You access the PDF version of this guide from the **Docs** directory on the Probe installation CD.

Note: The information in the Mercury Diagnostics Profiler guides is also included in the **Mercury Diagnostics Installation and User's Guide**.

- ▶ **J2EE Technical Practice Documents.** The following J2EE Technical Practice Documents are available in the **Docs** directory on the Mercury Diagnostics installation CD.
 - ▶ Advanced Instrumentation for Mercury Diagnostics for J2EE
 - ▶ Performance Impact Analysis of Mercury Diagnostics Probe for J2EE

Additional Online Resources

Customer Support Web Site uses your default Web browser to open the Mercury Customer Support Web site. This site enables you to browse the Mercury Support Knowledge Base and add your own articles. You can also post to and search user discussion forums, submit support requests, download patches and updated documentation, and more. The URL for this Web site is <http://support.mercury.com>.

Mercury Home Page uses your default Web browser to access Mercury's Web site. This site provides you with the most up-to-date information on Mercury and its products. This includes new software releases, seminars and trade shows, customer support, educational services, and more. The URL for this Web site is <http://www.mercury.com>.

Documentation Updates

Mercury is continually updating its product documentation with new information. You can download the latest version of this document from the Customer Support Web site (<http://support.mercury.com>).

To download updated documentation:

- 1** In the Customer Support Web site, click the **Documentation** link.
- 2** Under **Please Select Product**, select either **Business Availability Center**, **LoadRunner** or **Performance Center**.

Note that if one of these items does not appear in the list, you must add it to your customer profile. Click **My Account** to update your profile.

- 3** Click **Retrieve**. The Documentation page opens and lists the documentation available for the current release and for previous releases. If a document was updated recently, **Updated** appears next to the document name.
- 4** Click a document link to download the documentation.

Typographical Conventions

This guide uses the following typographical conventions:

UI Elements	This style indicates the names of interface elements on which you perform actions, file names or paths, and other items that require emphasis. For example, “Click the Save button.”
<i>Arguments</i>	This style indicates method, property, or function arguments and book titles. For example, “Refer to the <i>Mercury User’s Guide</i> .”
<Replace Value>	Angle brackets enclose a part of a file path or URL address that should be replaced with an actual value. For example, <MyProduct installation folder>\bin.
Example	This style is used for examples and text that is to be typed literally. For example, “Type Hello in the edit box.”
CTRL+C	This style indicates keyboard keys. For example, “Press ENTER.”
Function_Name	This style indicates method or function names. For example, “The wait_window statement has the following parameters:”
[]	Square brackets enclose optional arguments.
{ }	Curly brackets indicate that one of the enclosed values must be assigned to the current argument.
...	In a line of syntax, an ellipsis indicates that more items of the same format may be included. In a programming example, an ellipsis is used to indicate lines of a program that were intentionally omitted.
	A vertical bar indicates that one of the options separated by the bar should be selected.

Welcome to This Guide

Part I

Introduction

1

Preparing to Install Mercury Diagnostics

This chapter provides you with the information and instructions that will help you to plan and prepare for the installation and configuration of the Diagnostics components.

This chapter describes:	On page:
Mercury Diagnostics Components and Data Flow	4
Supported Application Servers and Environments	6
Host System Requirements for the Diagnostics Components	7
Information Required for Installation	11
Pre-installation Considerations	17
Recommended Order of Installation	18
Licensing Mercury Diagnostics	20
Upgrading from Earlier Versions of Diagnostics	20

Mercury Diagnostics Components and Data Flow

Mercury Diagnostics consists of the following components:

- ▶ **Mercury Diagnostics Probes.** Responsible for capturing events from your application, aggregating the metrics, and sending the performance metrics to a Diagnostics Server. The Probe captures events such as method invocations, the beginning and end of business transactions, and server transactions.

The .NET and J2EE Probes also provide the Profiler functionality which provides detailed diagnostics information about the application that is being monitored by the Probe. This information can be viewed from within the Diagnostics UI or from the Profiler's own UI.

To gather data from external ERP/CRM environments (SAP R/3 system or Oracle 10g Database), you install the Diagnostics Collector and define specific instances of Oracle 10g and SAP R/3 systems to be monitored. Each instance of a collector is represented as a Probe in the Mercury Diagnostics UI.

- ▶ **Mercury Diagnostics Servers.** Responsible for working with the Probes and with other Mercury products to capture, process, and present the performance metrics for your application.

The Diagnostics Server processes and further aggregates the data that it receives from each of the Probes that report to it, and formats the information so that it can be displayed in the views of the user interface.

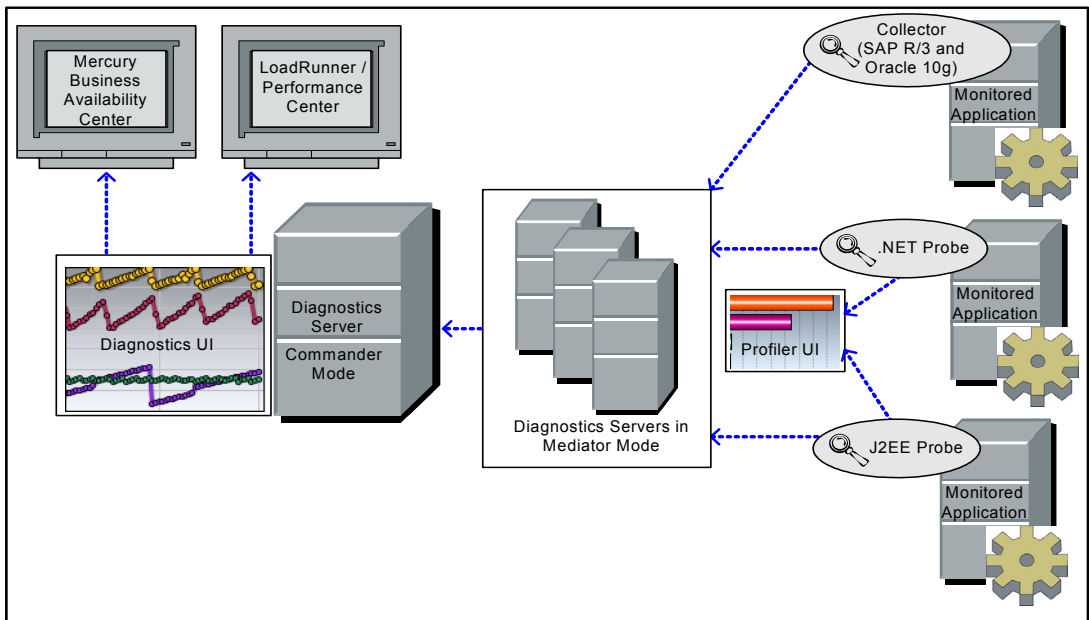
The Diagnostics Server in Commander mode is responsible for the command and control functions between the various Diagnostics components and the components of the other Mercury products with which Diagnostics is working. The Diagnostics Server in Commander mode keeps track of the location and status of the other Diagnostics components, and is the communication hub between the other components.

The Diagnostics Server in Commander mode is also responsible for displaying the performance information for the monitored applications in the charts and graphs of the Diagnostics views. If you are using Diagnostics with other Mercury products you can also access the Diagnostics views from the user interface of the other products.

A Diagnostics deployment may consist of one or many Diagnostics Servers. If there is only one Diagnostics server in your deployment, it is configured in Commander mode and must perform both the Commander and Mediator roles. If there is more than one Diagnostics Server in a deployment, one must be configured in Commander mode, and all the rest in Mediator mode.

Diagnostics Data Flow

The following diagram illustrates the data flow for a distributed Diagnostics deployment consisting of more than one Diagnostics Server.



In the diagram displayed above, there is one Diagnostics Server in Commander mode connected to a number of Diagnostics Servers in Mediator mode. Each Diagnostics Server in Mediator mode is connected to a number of Probes or Collectors. The Diagnostics Probes and the Diagnostics Collector capture events from your applications.

The Probes and Collectors send these captured performance metrics to the Diagnostics Server in Mediator mode which filters and aggregates the events. This information is sent to the Diagnostics Server in Commander mode, which displays the processed metrics in customizable views.

When you are monitoring your application in real time, you access the Diagnostics UI from Mercury Business Availability Center or directly from the Diagnostics Server. During a load test, you access the Diagnostics UI from LoadRunner or Performance Center.

You access the J2EE Diagnostics Profiler and the .NET Diagnostics Profiler directly from the Probe whose Profiler you want to view or through the Diagnostics UI.

Supported Application Servers and Environments

Mercury Diagnostics supports the monitoring of:

- ▶ **J2EE-based application servers.** Including: WebLogic, WebSphere, Oracle9iAS, Sun Java System, JBoss, and more.
- ▶ **.NET-based application servers.** Mercury Diagnostics supports the Microsoft IIS .NET Framework.
- ▶ **SAP R/3 system.**
- ▶ **Oracle 10g Database.**

Note: Mercury Diagnostics monitors only those methods that are invoked by the application server, and are instrumented by the Probe. For information about instrumentation and how to instrument for additional methods, see Chapter 14, “Instrumenting an Application.”

Host System Requirements for the Diagnostics Components

The following section describes the recommended system configurations for hosting the components of the Mercury Diagnostics. Refer to the deployment diagram in the previous section to understand the component hosts that are described in this section.

When you select the machines that will host the Diagnostics components, you must make sure that the system configuration of the machines supports the processing load and the number of applications that you will be monitoring.

Requirements for the Mercury Diagnostics Client Host

The user interface for Diagnostics is presented in a web browser using a JAVA applet that requires JRE 1.4.2_10 or above to be installed on the machine that displays UI.

Requirements for the Mercury Diagnostics Server Host

The system configuration requirements for the host of the Diagnostics Server depends upon the number of probes and Diagnostics Server in Mediator mode that are reporting to it.

Note: When a Diagnostics Server is designated as the Diagnostics Server in Commander mode, each Diagnostics Server in Mediator mode that reports to it counts as one half of a probe when determining the system configuration requirements.

For example if a Diagnostics Server in Commander mode had 20 Diagnostics Server in Mediator mode reporting to it, in addition to 5 probes, the table below indicates that it needs -Xmx700M as a parameter to its JVM.

The following table lists the required system configuration for the host of the Diagnostics Server:

Platform	Item	Up to 10 Probes	Up to 20 Probes	Up to 30 Probes
Windows	CPU	1x 1.0 GHz	1x 2.0 GHz	2x 2.4 GHz
Windows	Memory	768 MB	1 GB	2 GB
Solaris	CPU	1x Ultra Sparc 2	2x Ultra Sparc 2	2x Ultra Sparc 3
Solaris	RAM	1 GB	1.5 GB	2 GB
All	Java Heap	350 M	700 M	1400 M
All	Disk	TBD	TBD	TBD

Note: For configuration considerations related to the Diagnostics performance data that is stored on the host for the Diagnostics Server in Commander mode, see “Pre-Installation Data Management Considerations” on page 535.

Requirements for the Mercury Diagnostics Probe for J2EE Host

The overhead that the Mercury Diagnostics Probe for J2EE (J2EE Probe) imposes on the system being monitored is extremely low. The following are the recommendations for free memory and disk space that will support the Probe’s processing:

Platform:	All Platforms
Memory:	50 MB Additional RAM
Free Hard Disk Space:	200 MB Additional Space

Note: The additional memory must be allocated to the max heap for the JVM by adding `-Xmx???m` to the java settings in the application's start up script.

For information on setting the max heap for the J2EE Probe, see “Adjusting the Heap Size for the J2EE Probe in the Application Startup Script” on page 203.

Requirements for the J2EE Diagnostics Profiler User Interface Host

The user interface for the J2EE Diagnostics Profiler is presented in a web browser using a JAVA applet that requires JRE 1.4.2_10 or above to be installed on the machine that displays UI. This machine must be able to access the J2EE Diagnostics Profiler URL:

http://<probe_host>:<probeport>/profiler. The probes are assigned to the first available port beginning at 35000.

Requirements for the Mercury Diagnostics Probe for .NET Host

The overhead that the Mercury Diagnostics Probe for .NET (.NET Probe) imposes on the system being monitored is extremely low. The following are the recommendations for free memory and disk space that will support the Probe's processing:

Platform:	All Platforms
Memory:	60 MB Additional RAM
Free Hard Disk Space:	200 MB Additional Space

Note: .NET Framework 1.1 or later needs to be installed on your machine before you run the .NET Probe installation.

Requirements for the .NET Diagnostics Profiler User Interface Host

The user interface for the .NET Diagnostics Profiler is presented using DHTML/XML/XSLT/JScript technology that requires IE6 or later. The machine that is to be used to present the UI must be able to access the .NET Diagnostics Profiler URL: <http://<probehost>:<probeport>/profiler>. The probes are assigned to the first available port within the range defined during the Probe installation. The default port range is 35000 - 35100.

Requirements for the Diagnostics Collector Host

The Collector can be installed on any machine that can interact with the host machines of the SAP or Oracle application from which it is collecting data.

The following table lists the required system configuration for the host of the Diagnostics Collector:

Platform	Item	Up to 50 Probes
Windows	CPU	1x 1.0 GHz
Windows	Memory	1 GB
Solaris	CPU	1x Ultra Sparc 2
Solaris	RAM	1 GB
All	Java Heap	350 M
All	Disk	TBD

Information Required for Installation

Before you start installing the Diagnostics components, you should carefully plan the configuration of the Diagnostics components and the machines that will host them. You should also consider the location of the component hosts within your network topography.

The tables in the following sections have been provided to help you gather the information that you must provide during the installation of the Diagnostics components.

Note: When you are installing Mercury Business Availability Center with Diagnostics, when entering the names for the hosts of the Diagnostics components it is strongly recommended that you use fully qualified host names, that is, the machine name and the domain name.

Diagnostics Server

Information Required	Where to find it	Value
<p>For a Diagnostics Server in Commander mode, the location of the Mercury Diagnostics license that has been generated for the machine that will host the server.</p>	<p>Contact your Mercury support person to request a license and place it in a folder where it can be accessed from the Diagnostics Server installer.</p>	
<p>For a Diagnostics Server in Mediator mode, the URL for the Diagnostics Server in Commander mode.</p>	<p>After the Diagnostics Server in Commander mode has been installed, available from the System Health Monitor. (See Appendix C, “Using the System Health Monitor.”)</p>	
<p>Will the Diagnostics Server be used in a Mercury Managed Services environment or in the Mercury Business Availability Center environment?</p>		

J2EE Probe➤ **Mercury Product and Diagnostics Server Information**

Information Required	Where to find it	Value
Name of Mercury product(s) that will use the Probe	Choose according to product license. ➤ Performance Center 8.1 / LoadRunner 8.1 ➤ Mercury Business Availability Center 6.1 or later	
Diagnostics Server in Commander mode URL	System Health Monitor (See Appendix C, "Using the System Health Monitor.")	
Diagnostics Server Event Host	System Health Monitor (See Appendix C, "Using the System Health Monitor.")	
Diagnostics Server Event Port	System Health Monitor (See Appendix C, "Using the System Health Monitor.")	Default value: 2612

► **Probe and Application Server Information**

Information Required	Where to find it	Value
probe name	A <i>unique</i> string; Created by user. Note: The name of the probe should indicate the probe type, to help you distinguish between the different types of probes	For example: J2EEProbe1
probe group	Used to relate probes so that their metrics can be reported as a logical group. This is user defined at the time that the probe is installed.	Default value: DefaultProbeGroup
Type of application server that the J2EE Probe will be monitoring	The host system administrator.	
Application Server configuration properties	The host system administrator. The details will vary according to the application server that you are using.	
Location of the JRE executable	The host system administrator. This depends upon the type of application server you are configuring. See “Running the JRE Instrumenter” on page 155.	

.NET Probe► **Diagnostics Server Information**

Information Required	Where to find it	Value
URL of Diagnostics Server in Commander mode	System Health Monitor URL that the Probe will use to register with the Diagnostics Server. This is not required for using the.NET Diagnostics Profiler in a standalone mode.	
Diagnostics Server Event Host	System Health Monitor This is not required for using the.NET Diagnostics Profiler in a standalone mode.	
Diagnostics Server Event Port	System Health Monitor Default value: 2612 This is not required for using the.NET Diagnostics Profiler in a standalone mode.	

► **Probe and Port Information**

Information Required	Where to find it	Value
probe group	<p>Used to relate probes so that their metrics can be reported as a group.</p> <p>This is user defined at the time that the probe is installed.</p> <p>Default value: Default</p>	
Web Port Min	<p>System Administrator.</p> <p>The lowest port number in a range of ports on the probe host that can be assigned to the probe.</p> <p>Default value: 35000</p>	
Web Port Max	<p>System Administrator</p> <p>The highest port number in a range of ports on the probe host that can be assigned to the probe.</p> <p>Default value: 35100</p>	

Pre-installation Considerations

Note: Before you install any of the Diagnostics components on a Windows machine, make sure that the **Start > Settings > Control Panel > Administrative Tools > Services** window is not open.

LoadRunner and Performance Center Host Machines

- If LoadRunner 8.1 is already installed, ensure that the Controller/Console and main Mercury LoadRunner window are closed before you install the LoadRunner Diagnostics AddIn.
- The LoadRunner Diagnostics AddIn is not required for Performance Center.
- The time and time-zone settings of the host machines for the Diagnostics components must be consistent. You will encounter time-difference problems if the time is not properly set.

Mercury Diagnostics Server

- The performance metrics for Mercury Diagnostics cannot be displayed until the Diagnostics Server in Commander mode has been licensed with a valid license. For more information on obtaining a license and other licensing issues, see Chapter 3, “Licensing Mercury Diagnostics.”

Note: For optimal display of the Diagnostics views, ensure that your screen resolution is at least 1024x768.

Mercury Diagnostics Probe for J2EE

- ▶ The J2EE Probe must be installed on the same machine as the Java application under test.
- ▶ The Mercury Diagnostics Profiler for J2EE will operate in an unlicensed mode with load restrictions until it is able to connect to a Diagnostics Server in Commander mode that has been properly licensed. For more information on obtaining a license and other licensing issues, see Chapter 3, “Licensing Mercury Diagnostics.”

Mercury Diagnostics Probe for .NET

- ▶ The .NET Probe must be installed on the same machine as the .NET application under test.
- ▶ The Mercury Diagnostics Profiler for .NET will operate in an unlicensed mode with load restrictions until it is able to connect to a Diagnostics Server in Commander mode that has been properly licensed. For more information on obtaining a license and other licensing issues, see Chapter 3, “Licensing Mercury Diagnostics.”

Recommended Order of Installation

Careful planning and preparation for installing the components of Mercury Diagnostics can help you to avoid complications and errors, and allow you to complete the installation and configuration steps quickly.

Before you start, review the following information to get an overview of the entire installation and configuration process.

Note: The order of the installation presented here is the recommended order of installation for the products and components. Deviation from this recommended order of installation could increase the complexity of the installation process and produce unpredictable results.

1 Check the system requirements and installation considerations.

See “Host System Requirements for the Diagnostics Components” on page 7.

2 Install the Mercury Diagnostics Server.

See Chapter 2, “Installing the Mercury Diagnostics Server.”

3 Install and configure the Mercury Diagnostics Probe(s) and/or Collector.

- ▶ For a J2EE environment, see Chapter 6, “Installing the Mercury Diagnostics Probe for J2EE.”
- ▶ For a .NET environment, see Chapter 5, “Installing the Mercury Diagnostics Probe for .NET.”
- ▶ For Oracle and SAP R3 environments, see Chapter 4, “Installing the Diagnostics Collector.”

4 For J2EE Probes, configure the application server to work with the probes.

For more information, see Chapter 9, “Configuring the Application Servers.”

5 If Mercury Diagnostics is integrated with Mercury LoadRunner, Performance Center or Business Availability Center, it must be set up to use Mercury Diagnostics.

- ▶ For Mercury Business Availability Center, see Chapter 10, “Setting Up Mercury Business Availability Center 6.1 or Later to Use Diagnostics.”
- ▶ For LoadRunner integration, install the LoadRunner Diagnostics AddIn (see Chapter 11, “Installing LoadRunner 8.1 and the LoadRunner Diagnostics AddIn”) and set up LoadRunner to use Mercury Diagnostics (see Chapter 12, “Setting Up LoadRunner 8.1 to use Mercury Diagnostics”).
- ▶ For Performance Center, see Chapter 13, “Setting Up Performance Center 8.1 to Use Mercury Diagnostics.”

Licensing Mercury Diagnostics

To be able to see the performance metrics for your applications in the Diagnostics views, you must obtain a valid license for the Diagnostics Server in Commander mode. The license is a node-locking license that unlocks the Diagnostics views that are displayed by the Diagnostics Server in Commander mode and removes the load restriction from the Profiler views that are accessed from the Diagnostics Probes. For more information on obtaining a license and other licensing issues, see Chapter 3, “Licensing Mercury Diagnostics.”

Upgrading from Earlier Versions of Diagnostics

If you are installing Mercury Diagnostics into an environment where a previous version of the product was installed, or in an environment where other Mercury products need to be upgraded so that the features of Diagnostics can be accessed, follow the instructions in Appendix F, “Upgrading Diagnostics and Other Mercury Products.” These instructions guide you to the appropriate instructions for upgrading your current Mercury products and the Diagnostics components.

Part II

Installation of the Mercury Diagnostics Server

2

Installing the Mercury Diagnostics Server

This chapter describes how to install the Mercury Diagnostics Server on Windows and UNIX machines.

This chapter describes:	On page:
Installing the Diagnostics Server on a Windows Machine	24
Installing the Diagnostics Server on a UNIX Machine	35
Starting and Stopping the Diagnostics Server	46
Verifying the Diagnostics Server Installation	48
Licensing the Diagnostics Server in Commander Mode	49
Configuring the Diagnostics Server	49
Determining the Version of the Diagnostics Server	50
Uninstalling the Diagnostics Server	50
Troubleshooting Diagnostics Server Issues	52

Installing the Diagnostics Server on a Windows Machine

The following steps provide detailed instructions for installing the Diagnostics Server on a Windows machine.

To install the Diagnostics Server on a Windows machine:

1 Launch the installation program for the Diagnostics Server.

Insert the Mercury Diagnostics Servers installation CD into the CD-ROM drive of the Diagnostics Server host machine, and double-click **setup.exe** in the root folder of the CD.

Note: To launch the installer from a different location, locate **\Diagnostics_Server\DiagnosticsServerSetupWin.exe** on the CD, copy it to the new location, and then run it.

The installer displays the Mercury Diagnostics main installation menu.

Click **Diagnostics Server** to launch the installer.

2 Accept the software license agreement.

A dialog box opens displaying the license agreement.

Read the agreement and select **I accept the terms of the license agreement**.

Click **Next** to continue.

3 Specify the location where the Diagnostics Server is to be installed.

In the **Installation Directory Name** box, type the name of the directory where you want to install the Diagnostics Server. Alternatively, accept the default directory, or click **Browse** to navigate to another directory.

Note: If an earlier version of the Diagnostics Server is installed on your machine, you can choose to upgrade the earlier version by installing the new version in the same directory as the earlier one. Alternatively, you can install the new version in a different directory without upgrading the earlier version. For more information, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

Click **Next** to continue.

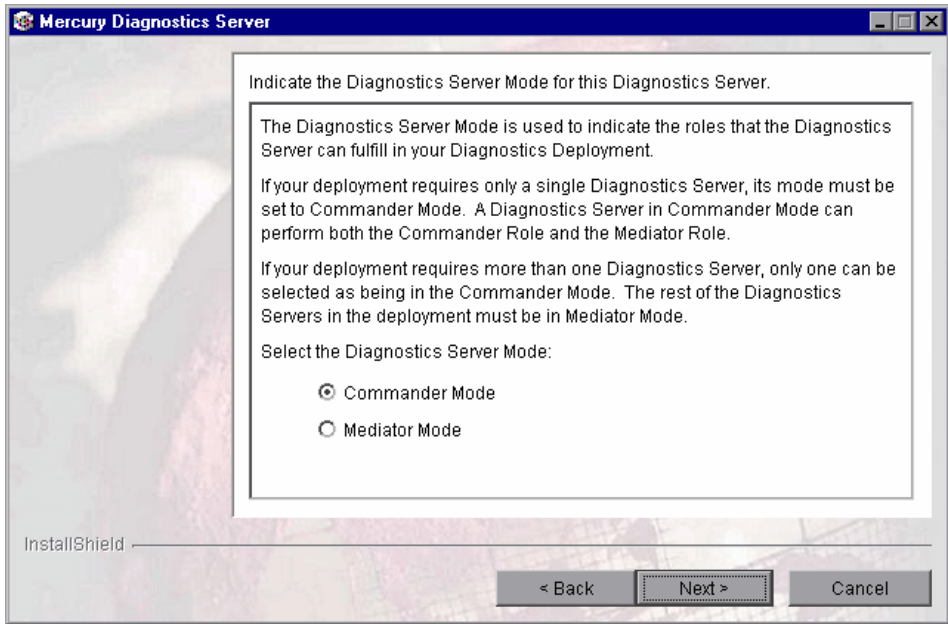
Note: If you specified a directory that contains an existing Diagnostics Server installation, the installation process backs up any existing property settings in a directory called **etc.old**.

4 Indicate the mode of the Diagnostics Server that you are installing.

The deployment that you are setting up may consist of one or many Diagnostics Servers. If there is only one Diagnostics server in your deployment, it is configured in Commander mode and can perform both Commander and Mediator roles. If there is more than one Diagnostics Server in a deployment, one must be configured in Commander mode, and all the rest in Mediator mode.

- ▶ If this is the only Diagnostics Server in your deployment, select **Commander Mode**.

- ▶ If there is more than one Diagnostics Server in your deployment, and the one that you are currently installing is to be configured in Commander mode, then select **Commander Mode**. Otherwise select **Mediator Mode**.



Click **Next** to continue.

Note: At this stage, the installation differs depending on whether you are installing the Diagnostics Server in Commander mode or in Mediator mode.

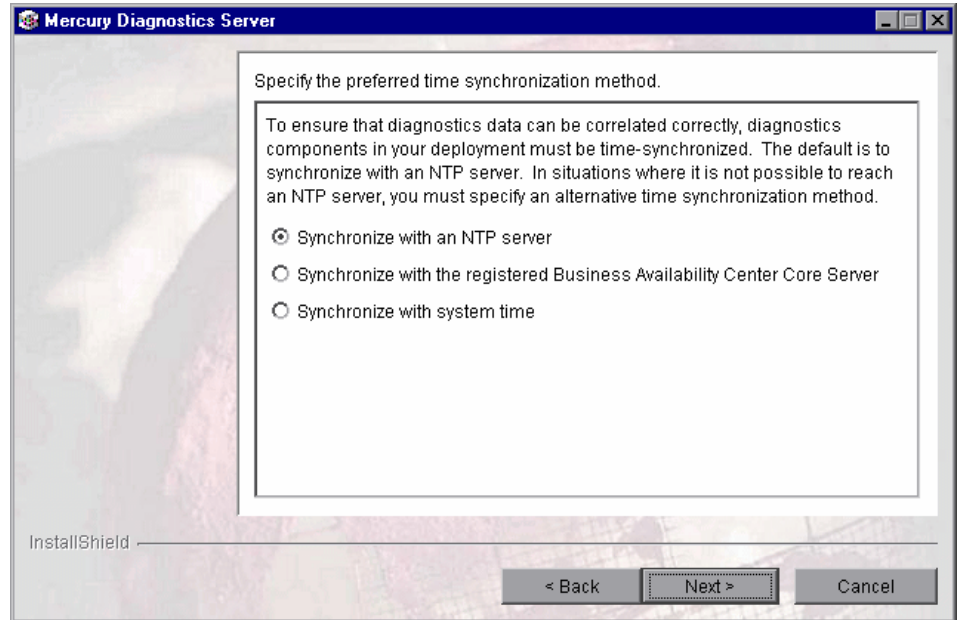
- ▶ To install the Diagnostics Server in Commander mode, continue with “Installing the Diagnostics Server in Commander Mode” on page 27.
 - ▶ To install the Diagnostics Server in Mediator mode, continue with “Installing the Diagnostics Server in Mediator Mode” on page 32.
-

Installing the Diagnostics Server in Commander Mode

If you are installing the Diagnostics Server in Commander mode, continue as follows:

1 Select a time synchronization method.

In order for diagnostics data to be correlated properly, all the components in the Diagnostics deployment must be time-synchronized.



Select one of the following time synchronization methods:

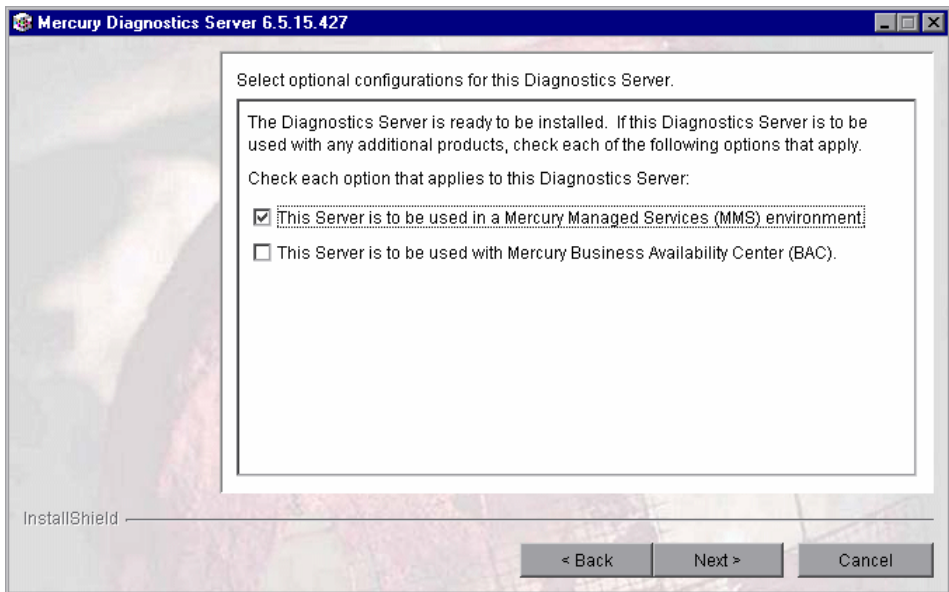
- ▶ **Synchronize with an NTP server.** This option applies only if the Diagnostics Server can access an NTP Server outside the firewall. This is the default method.
- ▶ **Synchronize with the registered Mercury Business Availability Center Core Server.** If the Diagnostics Server is to work in a Mercury Business Availability Center environment, select this option to synchronize with the Mercury Business Availability Center core server.

- **Synchronize with system time.** Select this option if the Diagnostics Server is to work in an environment other than Mercury Business Availability Center and there is no access to an NTP server.

Click **Next** to continue.

2 Indicate whether the Diagnostics Server will be used in a Mercury Managed Services environment or with Mercury Business Availability Center.

Select the options that apply to this Diagnostics Server.



Click **Next** to continue.

Note: If you selected only the **Mercury Managed Services** option, skip to step 4.

3 Mercury Business Availability Center environment only: Provide the path to the Probe installers.

Note: You need to have the Mercury Diagnostics installation CD available for this step.

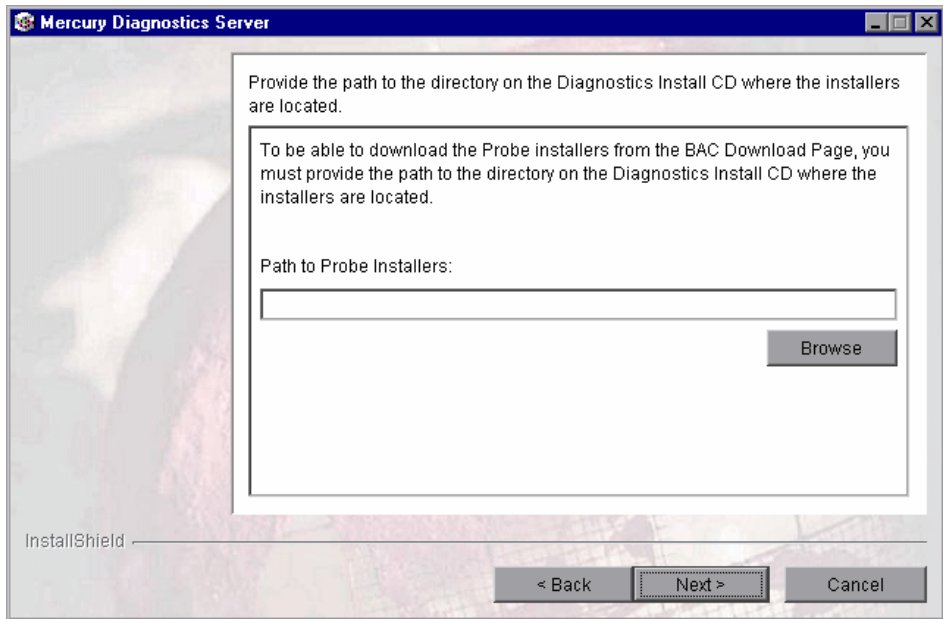
To be able to download the Diagnostics Probe installers from the Diagnostics Configuration page in Mercury Business Availability Center, you need to specify the path to the directory on the Mercury Diagnostics installation CD where these installers are located (**\Diagnostics_Probes**).

The Probe installers are copied to the Diagnostics Server installation directory, which Mercury Business Availability Center can access.

Note: You can choose to skip this step and always access the Probe installers directly from the product CD.

Alternatively, you can perform this step manually at a later stage, by copying the probe installers (**<Mercury Diagnostics installation CD>/Diagnostics_Probes**) to the Diagnostics Server installation directory (**<diag_server_install_dir>/html/opal/downloads**).

In the **Path to Probe Installers** box, type the path, or click **Browse** to navigate to the **Diagnostics_Probes** directory on the installation CD.



Click **Next** to continue.

4 Review the pre-installation summary.

The installation settings that you selected are displayed in a read-only window. Review the information to make sure that you are satisfied.

Note: The estimated total size of the Diagnostics Server in Commander mode installation does not include the size of the Probe installers, if they were made available for Mercury Business Availability Center.

To select different installation settings, click **Back**.

To begin installing the Diagnostics Server, click **Next**.

5 Close the installation wizard.

When the installation has finished running, a message is displayed, confirming that the Diagnostics Server was successfully installed. Click **Finish** to exit the installation.

Note: On Windows machines, the Diagnostics Server attempts to start automatically. The Diagnostics Server will not start if any other applications are using the default Diagnostics Server ports. For instructions on starting the Diagnostics Server manually, see “Starting and Stopping the Diagnostics Server” on page 46.

6 Upload the Diagnostics license file.

In order to view performance metrics, you must upload a valid Mercury Diagnostics license file after you have installed the Diagnostics Server in Commander mode.

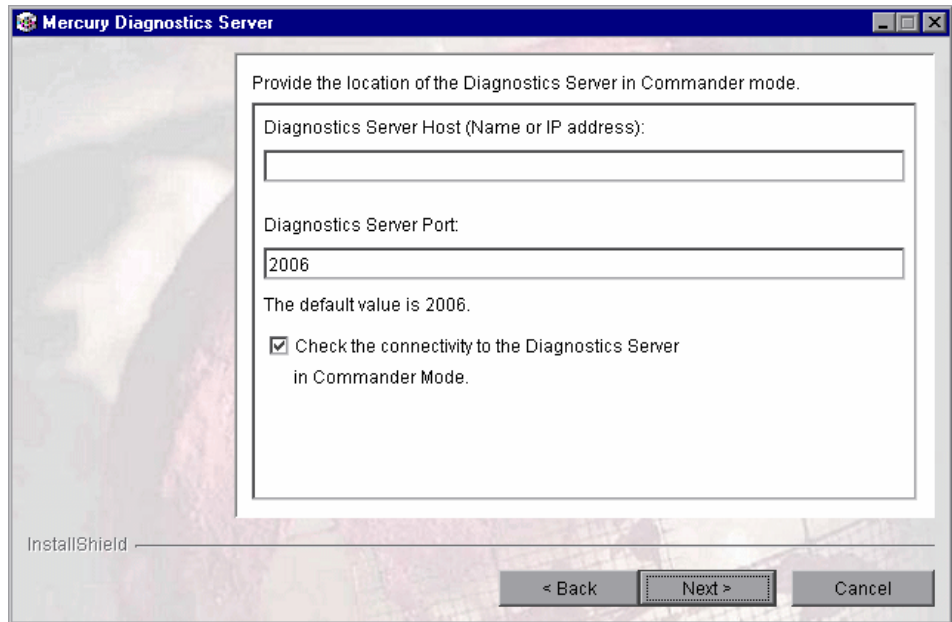
For instructions on requesting and uploading a valid license file, see Chapter 3, “Licensing Mercury Diagnostics.”

Installing the Diagnostics Server in Mediator Mode

If you are installing the Diagnostics Server in Mediator mode, continue the installation as follows:

1 Provide the location of the Diagnostics Server in Commander mode.

Provide the information that enables the Diagnostics Server in Mediator mode to connect to the Diagnostics Server in Commander mode.



- a** In the **Diagnostics Server Host** box, type the host name or IP address of the host for the Diagnostics Server in Commander mode.
- b** In the **Diagnostics Server Port** box, type the port number of the Diagnostics Server in Commander mode. The default port for the Diagnostics Server in Commander mode is **2006**. If you have changed the port of the Diagnostics Server in Commander mode since it was installed, you should specify that port number instead of the default. For information on changing the Diagnostics Server port, see “Changing the Default Diagnostics Server Port” on page 318.

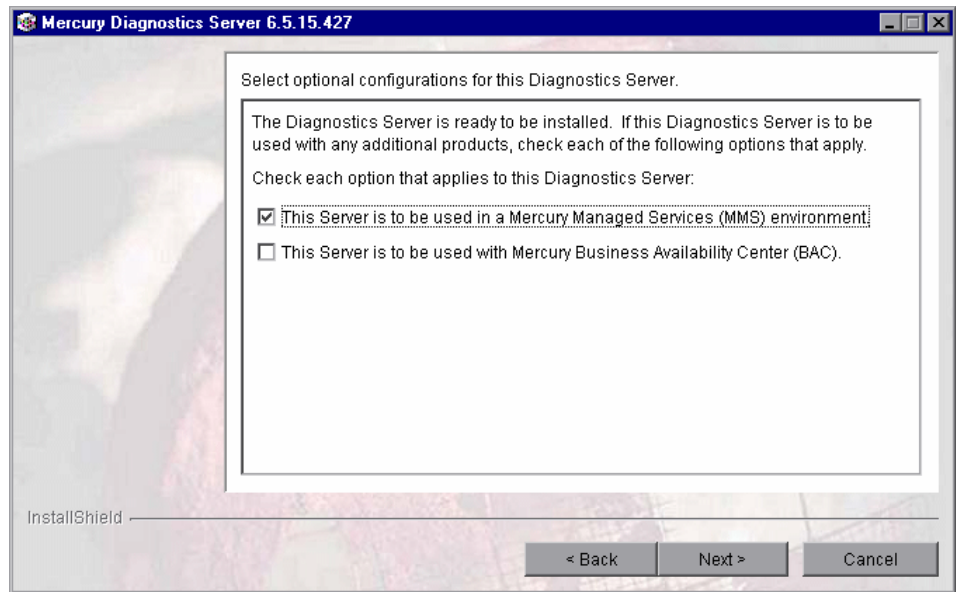
- c If you want to make sure that the Diagnostics Server is running and accessible from the installation host, select **Check the connectivity to the Diagnostics Server in Commander Mode**. (This is selected by default.)

Click **Next** to continue.

Note: If you selected **Check the connectivity to the Diagnostics Server in Commander Mode** and connectivity problems were encountered, the installer provides you with the results of the connectivity check. If you do not want to address these problems at this stage, clear the **Check the connectivity to the Diagnostics Server in Commander Mode** check box, proceed with the installation, and address the problem later.

2 Indicate whether the Diagnostics Server will be used in a Mercury Managed Services environment or with Mercury Business Availability Center.

Select that options that apply to this Diagnostics Server.

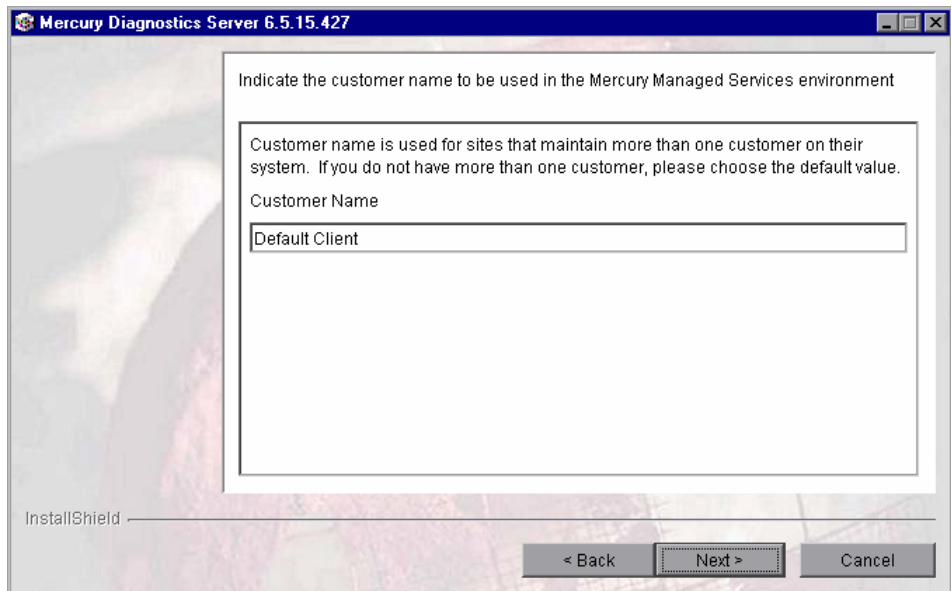


Click **Next** to continue.

Note: If you selected only the **Mercury Business Availability Center** option, skip to step 4.

3 If you indicated that the Diagnostics Server is to be used in a Mercury Managed Services environment, provide the customer name for the Mercury Managed Services environment.

Enter a unique name for the customer in the Mercury Managed Services environment.



Click **Next** to continue.

4 Review the pre-installation summary.

The installation settings that you selected are displayed in a read-only window. Review the information to make sure that you are satisfied.

To select different installation settings, click **Back**.

To begin installation, click **Next**.

5 Close the installation wizard.

When the installation has completed, review the post-installation summary information displayed to make sure that the installation completed successfully, and click **Finish**.

Note: On Windows machines, the Diagnostics Server attempts to start automatically. The Diagnostics Server will not start if any other applications are using the default Diagnostics Server ports. For instructions on starting the Diagnostics Server manually, see “Starting and Stopping the Diagnostics Server” on page 46.

Installing the Diagnostics Server on a UNIX Machine

This section provides instructions for installing the Diagnostics Server in most UNIX environments using either a graphical installation or a console mode installation.

Notes:

- ▶ The following instructions are for component installation on a Solaris machine. These same instructions should apply for the other certified UNIX platforms.
 - ▶ The following instructions assume an understanding of UNIX console screens and commands. For more information about UNIX screens and commands, see Appendix H, “Using UNIX Commands.”
-

To install the Diagnostics Server on a UNIX machine in console mode:

1 Launch the installation program for the Diagnostics Server.

- a** On the Mercury Diagnostics installation CD for UNIX, locate the **Diagnostics_Server** directory.
- b** Copy the installer, **DiagnosticsServerSetup<platform>.bin**, to the machine where the Diagnostics Server is to be installed.
- c** Change the mode of the installer file to make it executable.
- d** Run the installer.

- To run the installer in the graphical mode enter the following at the UNIX command prompt:

```
./DiagnosticsServerSetupSolaris.bin
```

The installer screens that are displayed in a graphical installation are similar to those documented for the Windows installer.

Continue with step 2 of “Installing the Diagnostics Server on a Windows Machine” on page 24.

- To run the installer in console mode enter the following at the UNIX command prompt:

```
./DiagnosticsServerSetupSolaris.bin -console
```

The installer continues as shown in the next step.

2 Accept the software license agreement.

The software license agreement is displayed.

Read the agreement. As you read, you can press **ENTER** to move to the next page of text, or type **q** to jump to the end of the license agreement.

Accept the terms of the agreement, and select **Next** to continue with the installation.

3 Specify the location where the Diagnostics Server is to be installed.

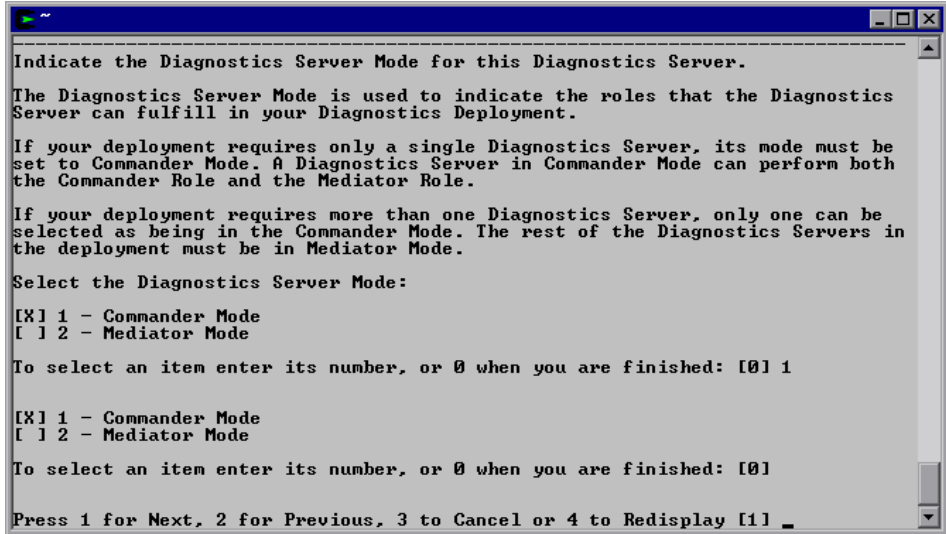
At the **Directory Name** prompt, accept the default installation location shown in brackets, or enter the path to a different location.

Note: If an earlier version of the Diagnostics Server is installed on your machine, you can choose to upgrade the earlier version by installing the new version in the same directory as the earlier one. Alternatively, you can install the new version in a different directory without upgrading the earlier version. For more information, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

Select **Next** to continue with the installation.

Note: If you specify a directory that contains an existing Diagnostics Server installation, the installation process backs up any existing property settings in a directory called **etc.old**.

4 Indicate the mode of the Diagnostics Server that you are installing.



The Diagnostics deployment that you are creating may consist of one or many Diagnostics Servers. If there is only one Diagnostics Server in your deployment, it is installed in Commander mode and can perform both Commander and Mediator roles. When there is more than one Diagnostics Server in a deployment, one is configured in Commander mode, and all the rest in Mediator mode.

- ▶ If this is the only Diagnostics Server in your deployment, select **Commander Mode**.
- ▶ If there is more than one Diagnostics Server in your deployment, and the one that you are currently installing is to be configured in Commander mode, then select **Commander Mode**. Otherwise select **Mediator Mode**.

Select **Next** to continue with the installation.

Note: At this stage, the installation differs depending on whether you are installing the Diagnostics Server in Commander mode or in Mediator mode.

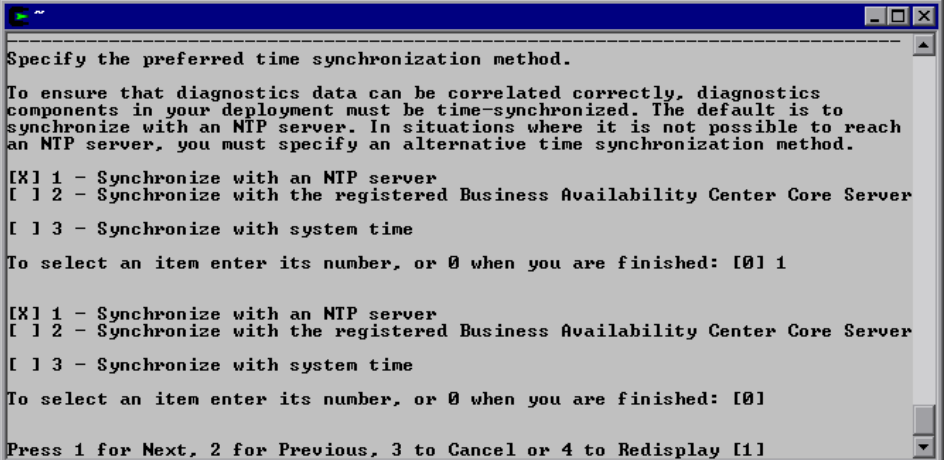
- ▶ To install the Diagnostics Server in Commander mode, continue with “Installing the Diagnostics Server in Commander Mode” on page 39.
- ▶ To install the Diagnostics Server in Mediator mode, continue with “Installing the Diagnostics Server in Mediator mode” on page 43.

Installing the Diagnostics Server in Commander Mode

If you are installing the Diagnostics Server in Commander mode, continue as follows:

1 Select a time synchronization method.

In order for diagnostics data to be correlated properly, all the components in your deployment must be time-synchronized.



```

Specify the preferred time synchronization method.

To ensure that diagnostics data can be correlated correctly, diagnostics
components in your deployment must be time-synchronized. The default is to
synchronize with an NTP server. In situations where it is not possible to reach
an NTP server, you must specify an alternative time synchronization method.

[X] 1 - Synchronize with an NTP server
[ ] 2 - Synchronize with the registered Business Availability Center Core Server
[ ] 3 - Synchronize with system time

To select an item enter its number, or 0 when you are finished: [0] 1

[X] 1 - Synchronize with an NTP server
[ ] 2 - Synchronize with the registered Business Availability Center Core Server
[ ] 3 - Synchronize with system time

To select an item enter its number, or 0 when you are finished: [0]

Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1]

```

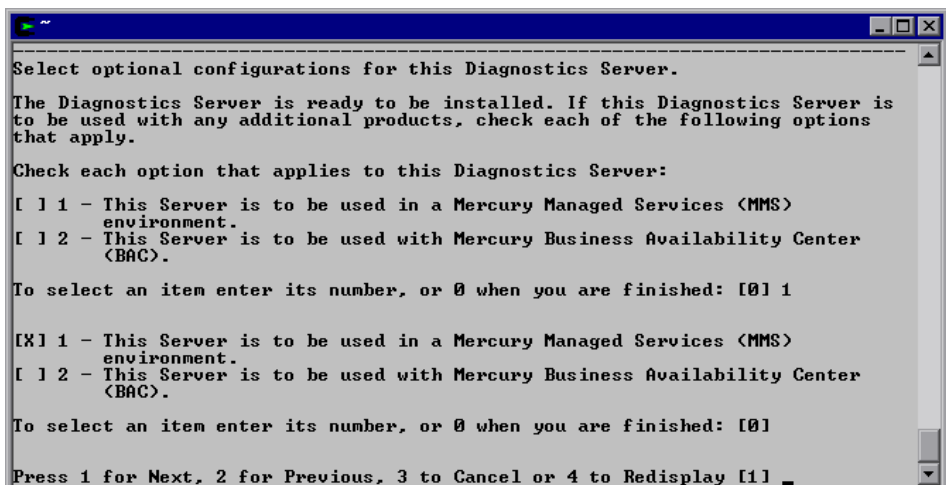
The following are the available synchronization options:

- ▶ **Synchronize with an NTP server.** This is the default method. This option applies only if the Diagnostics Server can access an NTP Server outside the firewall.

- **Synchronize with the registered Mercury Business Availability Center core server.** If the Diagnostics Server is to work in a Mercury Business Availability Center environment, select this option to synchronize with the Mercury Business Availability Center core server.
- **Synchronize with system time.** Select this option, if the Diagnostics Server is to work in an environment other than Mercury Business Availability Center and there is no access to an NTP server.

Select a time-synchronization option, and select **Next** and continue with the installation.

2 Indicate whether the Diagnostics Server will be used in a Mercury Managed Services environment or with Mercury Business Availability Center.



Select the options that apply to this Diagnostics Server, and select **Next** to continue with the installation.

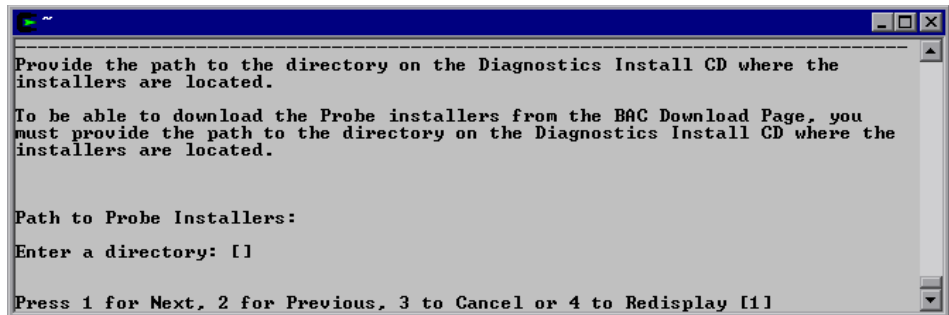
Note: If you selected only the Mercury Managed Services (MMS) option, skip to step 4.

3 Mercury Business Availability Center environment only: Provide the path to the Probe installers.

Note:

- ▶ You need to have the Mercury Diagnostics installation CD available for this step.
 - ▶ You can skip this step at this stage, and enter the path manually when you configure the Diagnostics Server, or you can access the Probe installers directly from the product CD.
-

To be able to download the Diagnostics Probe installers from the Diagnostics Configuration page in Mercury Business Availability Center, you need specify the path to the directory on the Mercury Diagnostics installation CD where the Probe installers are located (**\Diagnostics_Probes**). The Probe installers are copied to the Diagnostics Server installation directory, which Mercury Business Availability Center can access.



Note: The installers for the Diagnostics Collectors are not stored on the same CD as the Probe installers. For this reason, the Collector Installers are not automatically copied by the installers.

If you would like to be able to download the Collector installers from Mercury Business Availability Center, you must copy the installers to `<diag_server_install_dir>/html/opal/downloads`.

At the **Enter a directory** prompt, type the path to the Probe installers on the Diagnostics Installer CD, and select **Next** to continue with the installation.

4 Review the pre-installation summary.

The installation settings that you selected are displayed. Review the information to make sure that you are satisfied.

Note: The estimated total size of the Diagnostics Server in Commander mode installation does not include the size of the Probe installers, if they were made available for Mercury Business Availability Center.

To change your settings, select **Previous** to return to the previous prompts.

To start the installation of the Diagnostics Server, select **Next**.

5 Close the installation wizard.

When the installation has completed, review the post-installation summary information to make sure that the installation completed successfully.

Select **Finish** to exit the installation.

6 Upload the Diagnostics license file.

In order to view performance metrics, you must upload a valid Mercury Diagnostics license file after you have installed the Diagnostics Server in Commander mode.

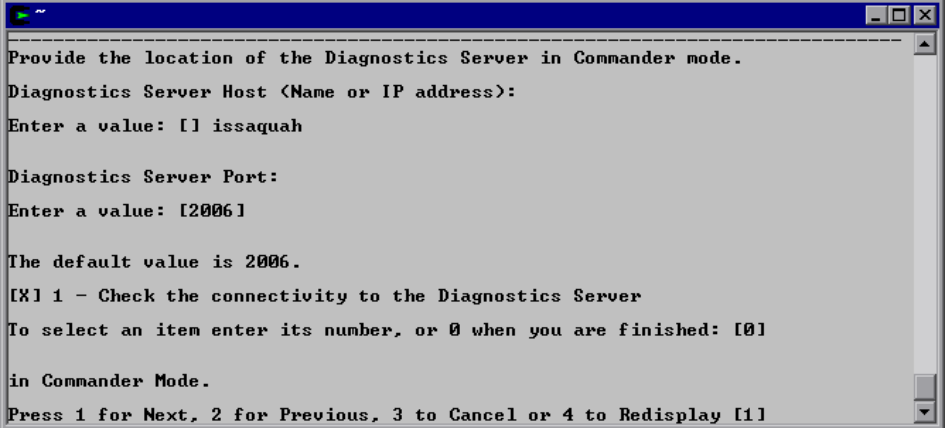
For instructions on requesting and uploading a valid license file, see Chapter 3, “Licensing Mercury Diagnostics.”

Installing the Diagnostics Server in Mediator mode

If you are installing the Diagnostics Server in Mediator mode, continue the installation as follows:

1 Provide the location of the Diagnostics Server in Commander mode.

Provide the information that enables the Diagnostics Server in Mediator mode to connect to the Diagnostics Server in Commander mode.



```

Provide the location of the Diagnostics Server in Commander mode.
Diagnostics Server Host (Name or IP address):
Enter a value: [1] issaquah

Diagnostics Server Port:
Enter a value: [2006]

The default value is 2006.
[X] 1 - Check the connectivity to the Diagnostics Server
To select an item enter its number, or 0 when you are finished: [0]

in Commander Mode.
Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1]

```

- a** Enter the host name for the Diagnostics Server in Commander mode.
- b** Enter the port for the Diagnostics Server in Commander mode.

The default port for the Diagnostics Server in Commander mode is **2006**. If you have changed the port since the Diagnostics Server was installed, you should specify that port number here instead of the default. For information on changing the Diagnostics Server port, see “Changing the Default Diagnostics Server Port” on page 318.

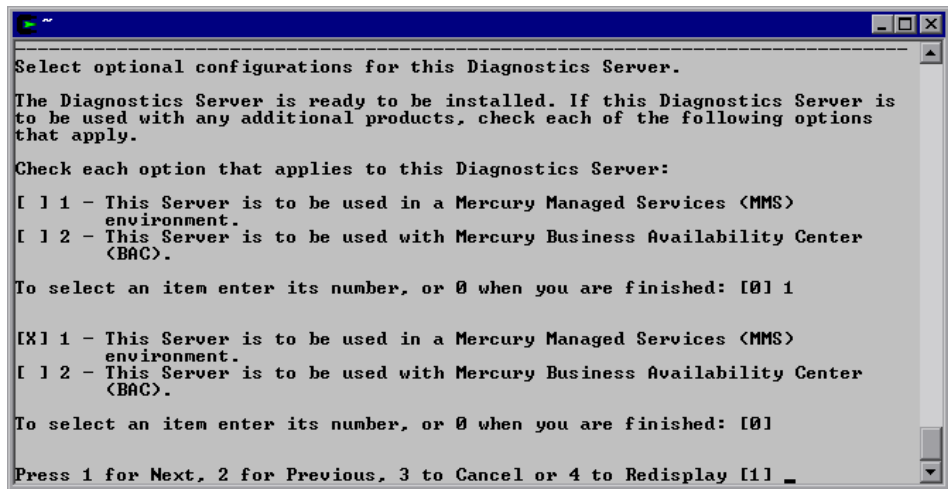
- c To allow the installer to check the connectivity to the host and port that you specified, select **Check the Connectivity to the Diagnostics Server**.

Select **Next** to continue with the installation.

If you instructed the installer to perform the test for connectivity, it tests the connectivity at this point. If there are negative results, it reports these before proceeding with the next installation step.

If you do not want to address these problems at this stage, clear the **Check the connectivity to the Diagnostics Server** option so that the installation can proceed.

2 Indicate whether the Diagnostics Server will be used in a Mercury Managed Services environment or with Mercury Business Availability Center.

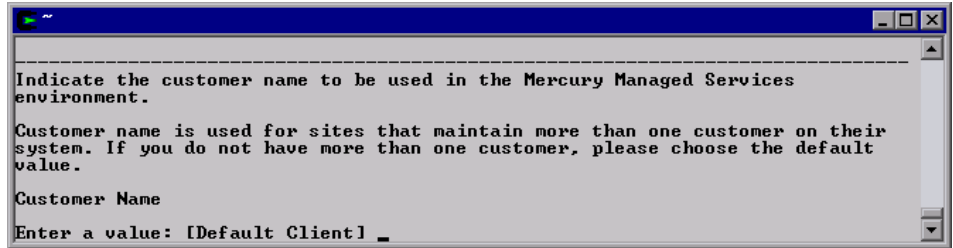


Select the options that apply to this Diagnostics Server, and select **Next** to continue with the installation.

Note: If you selected only the **Mercury Business Availability Center** option, skip to step 4.

3 If you indicated that the Diagnostics Server is to be used in a Mercury Managed Services environment, provide the customer name.

Enter a unique name for the customer in the Mercury Managed Services environment.



Select **Next** to continue with the installation.

4 Review the pre-installation summary.

The installation settings that you selected are displayed. Review the information to make sure that you are satisfied.

To change your settings, select **Previous** to return to the previous prompts.

To start the installation of the Diagnostics Server, select **Next**.

5 Close the installation wizard.

When the installation has completed, review the post-installation summary information to make sure that the installation completed successfully.

Select **Finish** to exit the installation.

Starting and Stopping the Diagnostics Server

Instructions for a Windows Machine

To start the Diagnostics Server on a Windows machine:

Select **Start > Programs > Mercury Diagnostics Server > Start Mercury Diagnostics Server**.

To stop the Diagnostics Server on a Windows machine:

Select **Start > Programs > Mercury Diagnostics Server > Stop Mercury Diagnostics Server**.

Instructions for a Solaris Machine

The *nanny* is a process that runs as a daemon to ensure that the Diagnostics Server is always running. The following procedures start and stop the Diagnostics Server using the nanny.

To start the Diagnostics Server on a Solaris machine:

- 1 Ensure that the **M_LROOT** environment variable is defined as the root directory of the Diagnostics Server.

For example, in *tcsh*, you could enter the following:

```
$> setenv M_LROOT <diagnostics_server_install_dir>/nanny/solaris
```

If the **M_LROOT** environment variable is not defined as the root directory, you will encounter the following error:

```
Warning : MDRV: cannot find lrun root directory . Please check your M_LROOT
Unable to format message id [-10791]
m_agent_daemon ( is down )
```

- 2 Change directories to **<diagnostics_server_install_dir>/nanny/solaris/bin**.
- 3 Run **m_daemon_setup** with the **-install** option, as in the following example:

```
$> ./m_daemon_setup -install
```

To stop the Diagnostics Server on a Solaris machine:

- 1** Change directories to `<diagnostics_server_install_dir>/nanny/solaris/bin`.
- 2** Run `m_daemon_setup` with the `-remove` option, as in the following example:

```
$> ./m_daemon_setup -remove
```

Instructions for a UNIX Machine

The following procedures start and stop the Diagnostics Server without using the nanny.

To start the Diagnostics Server on a UNIX machine:

Run `<diagnostics_server_install_dir>/bin/server.sh`.

To stop the Diagnostics Server on a UNIX machine:

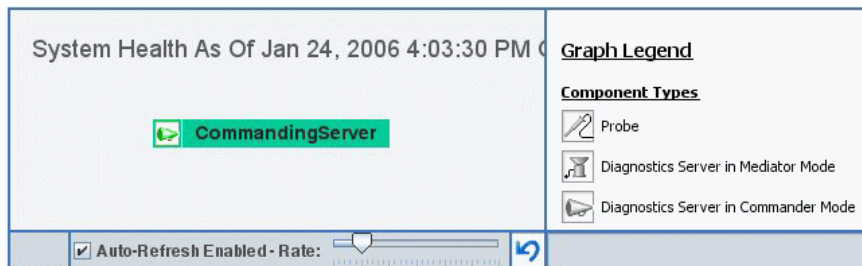
Terminate the process using a utility such as “kill.”

Verifying the Diagnostics Server Installation

To verify that the Diagnostics Server has been installed correctly, and that it has started properly, use the System Health Monitor. (For instructions on starting the System Health Monitor, see the *Mercury Diagnostics User's Guide*.)

According to the recommended installation sequence, after you install the Diagnostics Server you can use the System Health Monitor to verify that the Diagnostics Server was installed and started.

The Diagnostics Server in Commander mode should be displayed on the System Health Monitor as shown in the following example:



When a Diagnostics Server is deployed in Commander mode, it has both commanding and mediating responsibilities. The Diagnostics Server in Commander mode is represented in the System Health Monitor component map by an icon labelled **Commanding Server**.

When a Diagnostics Server is deployed in Mediator mode, it is represented by a single icon labelled **server-<name of host>**.

The System Health Monitor is part of the Diagnostics Server component. If you are unable to access the System Health Monitor after the Diagnostics Server has been installed, then either you are entering an incorrect URL or the Diagnostics Server did not start. For instructions on starting the Diagnostics Server, see “Starting and Stopping the Diagnostics Server” on page 46.

You can leave the System Health Monitor displayed in your browser to verify the progress of the component installations, and to identify and troubleshoot any problems that you encounter as you proceed through the rest of the component installations.

Licensing the Diagnostics Server in Commander Mode

After you have installed the Diagnostics Server in Commander mode you must upload a valid Diagnostics license file to the installation directory for the Diagnostics Server before you will be able to view the performance metrics in the views of the Diagnostics user interface.

For instructions on requesting a license file and uploading the file to the Diagnostics Server installation directory, see Chapter 3, “Licensing Mercury Diagnostics.”

Configuring the Diagnostics Server

The Diagnostics Server is installed with a default configuration that enables it to perform effectively in most situations. You may encounter situations where changing the configuration enables better Diagnostics Server performance, or allows it to work in unusual situations.

For information about configuring the Diagnostics Server, see Chapter 18, “Advanced Diagnostics Server Configuration.”

Determining the Version of the Diagnostics Server

When you request support, it is useful to know the version of the Diagnostics Server.



To determine the version of the Diagnostics Server:

Click the **Show Help** button from the Diagnostics tool bar and choose **About Mercury Diagnostics** from the menu.

The About Mercury Diagnostics dialog box opens, displaying the version of the Diagnostics Server.

Note: The version of the Diagnostics Server and the version of the Mercury LoadRunner Diagnostics AddIn must be compatible, so both versions should be upgraded at the same time. For more information, see “J2EE Probe Fails to Operate Properly” on page 564.

Uninstalling the Diagnostics Server

The following section contains instructions for uninstalling the Diagnostics Server from a Windows or Solaris machine.

Uninstalling the Diagnostics Server From a Windows Machine

- ▶ Uninstall the Diagnostics Server by selecting **Start > Programs > Mercury Diagnostics Server > Uninstall Mercury Diagnostics Server**.
- ▶ Alternatively, you can execute **uninstaller.exe** which is located in the **<diagnostics_server_install_dir>_uninst** directory.

During the uninstallation process, a message asks if you want to remove specific files. Do the following:

- ▶ If you want to completely uninstall the Diagnostics Server, as well as any property settings, click **Yes** or **Yes to All**.
- ▶ If you plan on reinstalling the Diagnostics Server, and want to keep the custom property settings of the Diagnostics Server you are uninstalling, then you need to back up the property files located in the **etc** directory to a new location.

If you have backed up these files, click **Yes** or **Yes to All**.

If you have not yet backed up these files, select **No** or **No to All**.

Uninstalling the Diagnostics Server From a Solaris Machine

You can uninstall the Diagnostics Server in console mode or graphical mode.

To uninstall the Diagnostics Server:

- 1** Stop the Diagnostics Server. For instructions, see “Starting and Stopping the Diagnostics Server” on page 46.
- 2** Change the directory to the root directory.
- 3** Enter the following at the UNIX command prompt:

- ▶ **In console mode:**

```
<diagnostics_server_install_dir>/jre/bin/java -jar
<diagnostics_server_install_dir>/_uninst/uninstall.jar -console
```

- ▶ **In graphical mode:**

```
<diagnostics_server_install_dir>/jre/bin/java -jar
<diagnostics_server_install_dir>/_uninst/uninstall.jar
```

Troubleshooting Diagnostics Server Issues

To troubleshoot Diagnostics Server problems, use the System Health Monitor. For more information, see “Using the System Health Monitor” on page 505.

3

Licensing Mercury Diagnostics

This chapter describes how to license Mercury Diagnostics.

This chapter describes:	On page:
About Mercury Diagnostics Licensing	53
Types of Licenses	54
Licensing the Diagnostics Server in Commander Mode	54

About Mercury Diagnostics Licensing

Diagnostics components can be installed before you have received the license. They can also be started so that they will begin to monitor your applications and process the performance metrics. However, the performance metrics that you can view the using the views of the Diagnostics user interface or the tabs in the Profilers is load-limited until you have provided a valid license file.

Mercury Diagnostics is licensed using a file that you upload to the Diagnostics Server in Commander mode. The license uses node-locking based on the MAC address for the host of the Diagnostics Server.

After you have installed the Diagnostics Server in Commander mode, you must ask your Mercury Customer Support representative to generate a license file for the Diagnostics deployment using the MAC address of the Diagnostics Server host which you provide to them.

When you receive the license file, you must store it in a location accessible by the host of the Diagnostics Server. You then upload the file using the License Upload area of the Diagnostics Server License Management page.

The Diagnostics Server in Mediator mode and Probes associated with the licensed Diagnostics Server in Commander mode are licensed based upon the license installed for the Diagnostics Server in Commander mode.

Types of Licenses

There are two types of licenses that can be issued for Diagnostics:

- A *perpetual* license is generated without an expiration date.
- A *custom* license is generated with a built in expiration date. The expiration date is set so that the license will be valid for a specific length of time. This type of license allows for a trial period before you purchase a perpetual license.

Licensing the Diagnostics Server in Commander Mode

You obtain your Diagnostics license from you Mercury Support contact, and then upload it to the Diagnostics Server installation directory.

To license your Diagnostics deployment:

- 1** Determine the MAC address for the host of the Diagnostics Server in Commander mode.
- 2** Request a license from your Mercury Support contact.
- 3** Upload the license file of the Diagnostics Server in Commander mode.
- 4** License the other Diagnostics components. When the Probes and Diagnostics Server in Mediator mode first connect with the Diagnostics Server in Commander mode they are licensed based upon the license installed on the Diagnostics Server in Commander mode.

Determining the MAC Address for the Diagnostics Server Host

The Diagnostics License is a node-locked license that is keyed off of the MAC address for the host of the Diagnostics Server in Commander mode.

To determine the MAC address for the Diagnostics Server Host:

- 1 Open the License Management page for the Diagnostics Server in Commander mode. From the Diagnostics Server administration menu select **Configure Diagnostics -> Licensing**.

The License Management page opens.

MERCURY™

License Management

System Information

Attribute	Value
MAC Address:	00:0D:56:70:50:6B, 00:0D:56:70:50:6A, 00:10:18:06:42:D8
<input type="button" value="Refresh"/>	

License Information

Attribute	Value
License:	Diagnostics Server License
Type:	Custom
Registered To:	Developer at Mercury
Issue Date:	Mar 17, 2006
Expiration:	Jul 1, 2006
Maximum Java Probes:	Unlimited
Maximum .NET Probes:	Unlimited
Maximum SAP Probes:	Unlimited
MAC Address:	Any

License Upload

Note: You may only upload files ending in ".lic". The uploaded file will be renamed to "DiagnosticsServer.lic".

License File:

Starting Mercury Diagnostics Server "server-helena", version 4.1.36.418

- 2 Note the MAC Address value displayed in the System Information section of the page. Click **Refresh** to make sure that the Diagnostics Server has discovered the current MAC Address information.

Be sure to let your Mercury Support contact know if the MAC Address value is **Unknown** when you request the license file. Do not look up the MAC Address using another method.

Note: You must use the MAC Address that is displayed in the System Information section of the License Management page when requesting a license. If the Diagnostics Server cannot determine a MAC address to display on the License Management page, it will not be able to find the MAC address when it attempts to validate a MAC address based license.

Requesting a License From Your Mercury Support Contact

Your Mercury Support contact can generate a license file for your Diagnostics Server host using the MAC Address value that you found on the License Management page.

To request a Diagnostics License:

- 1 Contact Mercury Support and provide them with the MAC Address that was displayed on the License Management page.
- 2 The Mercury Support contact generates a license file and sends it to you.

Uploading the License File

When you receive the license file that was generated for your Diagnostics deployment you must upload the file to the installation directory for the Diagnostics Server in Commander mode using the License Management page of the Server's Maintenance pages.

To upload the Diagnostics license:

- 1 Store the license file in a directory that can be accessed from the License Management page for the Diagnostics Server in Commander mode.

Note: The name of the file to be uploaded must end with the extension **.lic**.

- 2 In the License Upload section of the License Management page, type the path to the location where the license file was stored or click **Browse** to navigate to the license file location.

License Upload

Note: You may only upload files ending in ".lic". The uploaded file will be renamed to "DiagnosticsServer.lic".

License File:

- 3 Click **Upload** to apply the license file to the Diagnostics Server.

The file is renamed by the upload process and stored in the proper location in the Diagnostics Server install directory.

Note: Do not attempt to copy the license file directly to the Diagnostics Server installation directory. Always upload the file using the License Upload section of the License Management screen.

Licensing the Other Diagnostics Components

The Diagnostics Server in Mediator mode and the Diagnostics Probes do not have independent licenses. Their license is based upon the license of the Diagnostics Server in Commander mode. The first time they connect to a licensed Diagnostics Server in Commander mode, the Diagnostics Probes and Diagnostics Server in Mediator mode are automatically licensed.

When you install the J2EE or .NET probe, the Mercury Diagnostics Profiler is automatically installed as well. The Profiler is an independent diagnostics application that can be accessed either directly through the built in Profiler UI or through the Mercury Diagnostics UI.

The Mercury Diagnostics Profiler operates in an unlicensed mode with load restrictions until the probe is able to connect to a Diagnostics Server in Commander mode that has been properly licensed. In unlicensed mode, the Profiler is limited to capturing data from 5 concurrent threads.

Part III

Installation and Configuration of the Mercury Diagnostics Probes

4

Installing the Diagnostics Collector

This chapter provides instructions for installing a Diagnostics Collector on Windows and UNIX machines.

This chapter describes:	On page:
About Installing the Diagnostics Collector	62
Installing the Diagnostics Collector on a Windows Machine	62
Installing the Diagnostics Collector on a UNIX Machine	70
Silent Installation of the Diagnostics Collector	78
Configuring the Active System Property Files	79
Verifying the Diagnostics Collector Installation	82
Starting and Stopping the Diagnostics Collector	83
Determining the Version of the Diagnostics Collector	85
Uninstalling the Diagnostics Collector	85

About Installing the Diagnostics Collector

The Diagnostics Collector gathers data from external active systems. The Collector can be configured to collect performance data from two types of active systems:

- ▶ SAP R/3 system
- ▶ Oracle 10g Database

During the installation of the Collector, you can choose to monitor either one or both of these active systems. After the installation, you define instances of Oracle 10g and SAP R/3 systems to be monitored. Each one of these instances is represented as a probe (either Oracle Probe or SAP R/3 Probe) belonging to a probe group in the Mercury Diagnostics UI.

Installing the Diagnostics Collector on a Windows Machine

The following steps provide detailed instructions for installing the Collector on a Windows machine. These instructions also apply when you are installing the Collector on a UNIX machine using the graphical installer.

Note: The Collector can be installed on any machine. It does not necessarily have to be installed on the host machine of the SAP or Oracle application. For Collector host requirements, see “Requirements for the Diagnostics Collector Host” on page 11.

Launching the Installation

The installation can be launched directly from the installation CD, or from the Diagnostics Downloads page in Mercury Business Availability Center.

To launch the installation from the installation CD:

- 1** Insert the Mercury Diagnostics for Windows installation CD into the CD-ROM drive. If the installer does not launch automatically, double-click **setup.exe** in the root folder of the CD.

Note: To install the Collector from a different location, locate the executable file **CollectorSetupWin.exe** file in the **Diagnostics_Collector** directory on the CD, copy it to the new location, and then launch it.

The installer displays the Mercury Diagnostics main installation menu.

- 2** To start the installation for the Diagnostics Collector, click **Mercury Diagnostics Collector**.

Continue with “Running the Installation” on page 64.

For Mercury Business Availability Center Users

Note: To be able to download the Collector installer from the Diagnostics Downloads page in Mercury Business Availability Center, locate the **\Diagnostics_Collector\CollectorSetupWin.exe** on the Diagnostics CD, and copy it to the following folder of the Diagnostics Server installation directory: **<diag_server_install_dir>/html/opal/downloads**.

To start the installation from the Mercury Business Availability Center downloads page:

- 1** From the top menu in Mercury Business Availability Center, select **Admin > Diagnostics**, and click the **Downloads** tab.
- 2** On the Downloads page, click the appropriate link to launch the Collector installation for Windows.

Continue with “Running the Installation” on page 64.

Running the Installation

After you have launched the installation, the software license agreement opens.

To install the Collector:

- 1** **Accept the software license agreement.**

Read the agreement and select **I accept the terms of the license agreement**.

Click **Next** to continue.

- 2** **Specify the location where you want to install the Collector.**

In the **Installation Directory Name** box, type the name of the directory where you want to install the Collector. Alternatively, accept the default directory, **C:\MercuryDiagnostics\Collector**, or click **Browse** to navigate to another directory.

If there is a pre-existing installation of the **Collector** on the host machine, you can upgrade the existing **Collector** or install the **Collector** to a different installation directory.

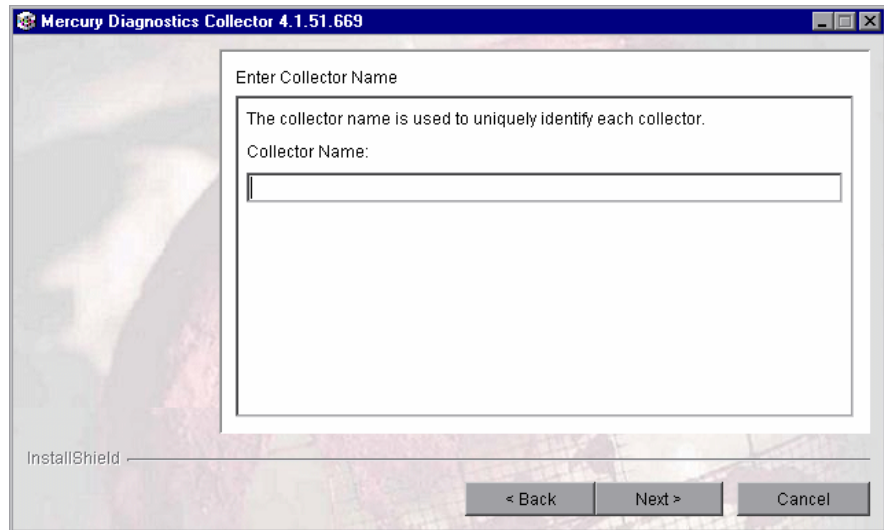
- To upgrade the existing **Collector** installation, ensure that the installation directory of the existing **Collector** has been specified in the **Installation Directory Name** box. For more information about upgrading Diagnostics components, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

When the installer runs, it will back up any property settings of the existing **Collector** installation to a directory called **etc.old**.

- ▶ To install the probe in an installation directory that is different from the existing **Collector's** directory, ensure that the installation directory of the existing **Collector** has not been specified in the **Installation Directory Name** box.

Click **Next** to continue.

3 Assign a unique name to the Collector.

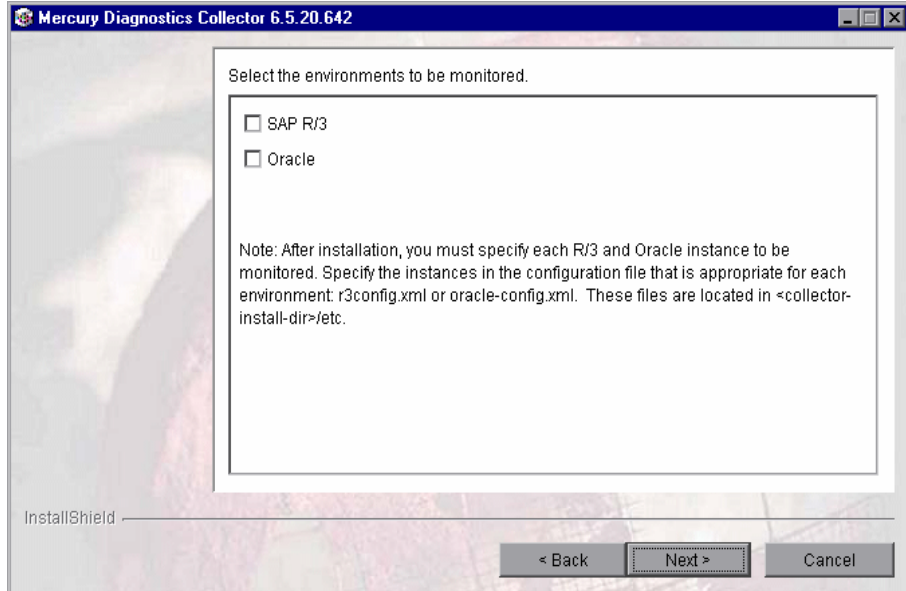


Assign a name to the Collector that will uniquely identify this specific Collector in the System Health Monitor. You use the System Health Monitor to verify the Collector installation and configuration. For more information, see “Verifying the Diagnostics Collector Installation” on page 82.

The following characters can be used in the name: - , _ , and all alphanumeric characters.

Click **Next** to continue.

4 Select the environment to monitor.



Select the options that apply to this Collector. You can select either one or both options.

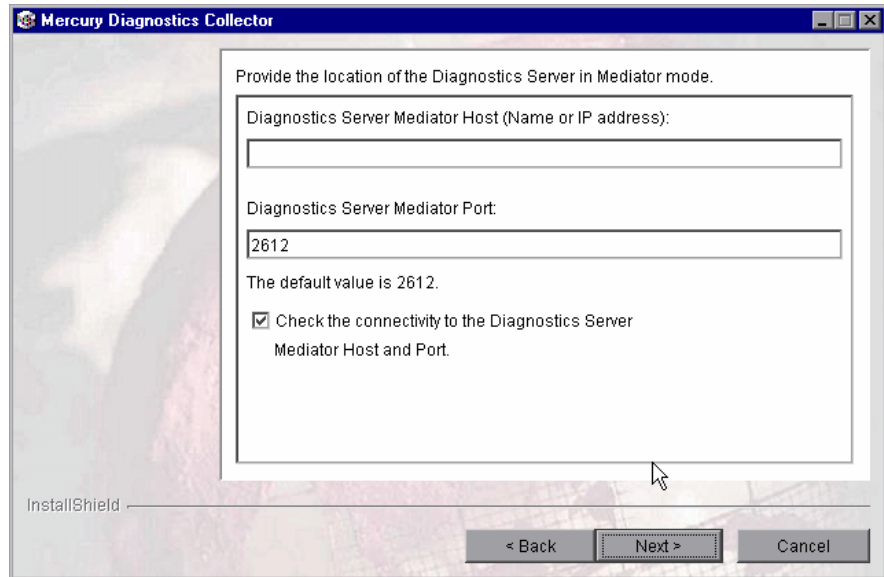
- To collect data in an SAP R/3 environment, select **SAP R/3**.
- To collect data on an Oracle 10g database server, select **Oracle**.

Important: After installation, you specify each of the SAP R/3 and Oracle instances to be monitored. These instances are manually defined in the XML files provided with the installation. For more information, see “Configuring the Active System Property Files” on page 79.

Click **Next** to continue.

5 Provide information about the Diagnostics Server in Mediator mode.

Provide the details that enables communication with the Diagnostics Server in Mediator mode.



If there is only one Diagnostics Server in the Diagnostics deployment where the Collector will run, enter the host name of the Diagnostics Server and its event port information.

If there is more than one Diagnostics Server in the deployment, enter the information for the Diagnostics Server in Mediator mode that is to receive the events from the Collector.

- a** In the **Diagnostics Server Mediator Host** box, type the host name or IP address of the host for the Diagnostics Server in Mediator mode.

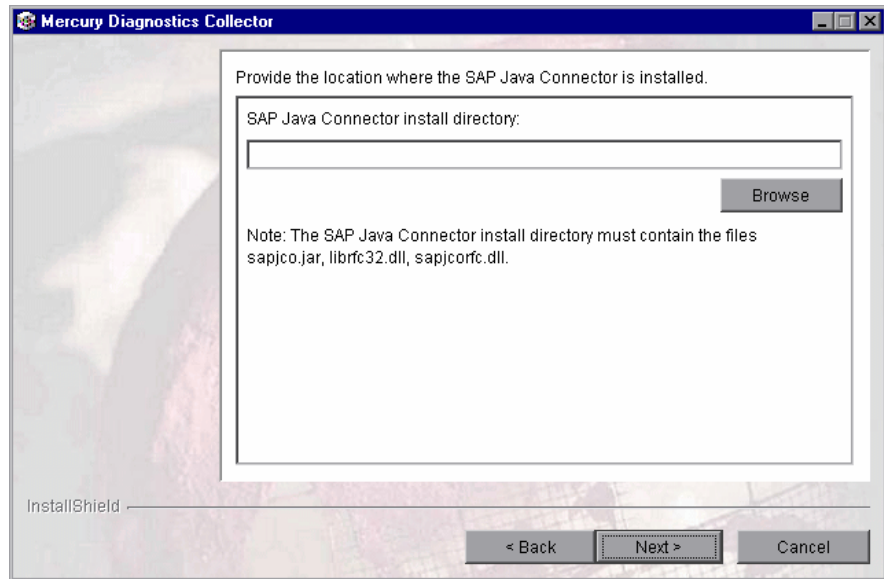
Note: You should specify the fully qualified host name. In a mixed OS environment, where UNIX is one of the systems, this is essential for proper network routing.

- b** In the **Diagnostics Server Mediator Port** box, type the port number where the Diagnostics Server is listening for Collector communication. The default port number is **2612**. If you have changed the port since the Diagnostics Server was installed, specify that port number instead of the default.
- c** If you want to make sure that the Diagnostics Server is running and accessible from the installation host, select **Check the connectivity to the Diagnostics Server Mediator Host and Port**.

Click **Next** to continue.

Note: If you selected **Check the connectivity to the Diagnostics Server Mediator Host and Port** and connectivity problems were encountered, the installer provides you with the results of the connectivity check. If you do not want to address these problems at this stage, clear the **Check the connectivity to the Diagnostics Server Mediator Host and Port** check box, proceed with the installation, and address the problem later.

6 If you selected SAP R3 in step 4, provide the location of the SAP Java Connector.



In the **SAP Java Connector install directory** box, enter the name of the directory where the SAP Java Connector is installed.

This directory must contain the following files:

- sapjco.jar
- librfc.dll
- sapjcorfc.dll

Note:

- If you do not know the SAP Java Connector directory name, contact your SAP representative.
 - If any of the files is missing from this directory, contact your SAP representative.
-

7 Review the pre-installation summary.

The installation settings that you selected are displayed in a read-only window. Review the information to make sure that you are satisfied.

To select different installation settings, click **Back**.

To begin installation, click **Next**.

8 Close the installation wizard.

When the installation has completed, a message is displayed, confirming that the Collector was successfully installed. Click **Finish** to exit the installation.

9 Configure the XML files for your active systems.

In step 4 you selected the active systems that will be monitored. For each of these active systems, you need to configure properties that enable the Collector host and the active system host to communicate.

For instructions on configuring the relative active system properties, see “Configuring the Active System Property Files” on page 79.

10 Verify that Collector was installed properly, and is running.

Using the System Health Monitor, you can verify that the Collector is running as it should be. For details see, “Verifying the Diagnostics Collector Installation” on page 82.

Installing the Diagnostics Collector on a UNIX Machine

The following instructions provide you with the steps necessary to install the Collector in Solaris and Linux environments, using either a graphical installation or a console mode installation.

The installation screens that are displayed in a graphical installation are the same as those documented for the Windows installation in “Installing the Diagnostics Collector on a Windows Machine” on page 62.

Launching the Installation

The installer can be launched directly from the installation CD, or from the Diagnostics Downloads page in Mercury Business Availability Center.

To launch the installation from the installation CD:

- 1 Copy the UNIX installer from the **Diagnostics_Collector** directory on the Diagnostics Collector installation CD to the machine where the Collector is to be installed.
- 2 Continue with “Running the Installation” on page 72.

For Mercury Business Availability Center users:

To start the installation from the Mercury Business Availability Center downloads page:

- 1 From the top menu in Mercury Business Availability Center, select **Admin > Diagnostics**, and click the **Downloads** tab.
- 2 On the Downloads page, click the link to the installer that is appropriate for your environment.
- 3 Save the installer on the machine where the Collector is to be installed.

Note: The Collector installer is available in Mercury Business Availability Center only if you provided the path to the Probe installers directory when you installed the Diagnostics Server in Commander mode. See “Installing the Diagnostics Server in Commander Mode” on page 27.

Continue with “Running the Installation” on page 72.

Running the Installation

After you have copied the installer to the machine where the Collector is to be installed, you are ready to run the installation.

Note: The following instructions assume an understanding of UNIX console screens and commands. For more information about UNIX screens and commands, see Appendix H, “Using UNIX Commands.”

To install the Collector:

1 Run the installer.

Where necessary, change the mode of the installer file to make it executable.

- ▶ To run the installer in the graphical mode, enter the following at the UNIX command prompt:

```
./<installer>
```

The installer displays the same screens that are displayed for the Windows installer.

Continue with “Installing the Diagnostics Collector on a Windows Machine” on page 62.

- ▶ To run the installer in console mode, enter the following at the UNIX command prompt:

```
./<installer> -console
```

The installer continues in console mode, and displays the software license agreement.

Continue with step 2.

2 Accept the software license agreement.

Read the license agreement and accept the terms.

Select **Next** to continue with the installation.

3 Specify the location where you want to install the Collector.

At the **Directory Name** prompt, accept the default installation location shown in brackets, or type the path to a different installation location.

If there is a pre-existing installation of the **Collector** on the host machine, you can upgrade the existing **Collector** or install the **Collector** to a different installation directory.

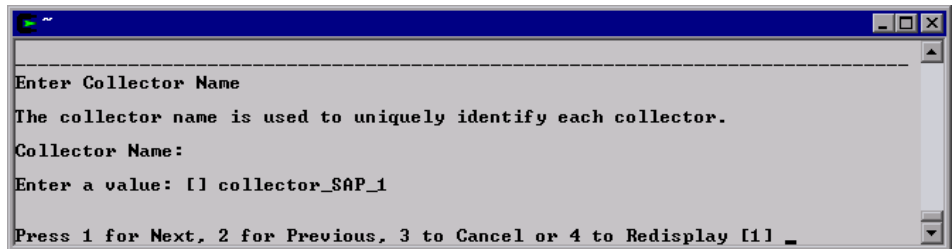
- To upgrade the existing **Collector** installation, specify the installation directory of the existing **Collector** at the **Directory Name** prompt. For more information about upgrading Diagnostics components, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

When the installer runs, it will back up any property settings of the existing **Collector** installation to a directory called **etc.old**.

- To install the **Collector** in an installation directory that is different from the existing **Collector**’s directory, specify an installation directory other than that of the existing **Collector**.

Select **Next** to continue with the installation.

4 Assign a unique name to the Collector.

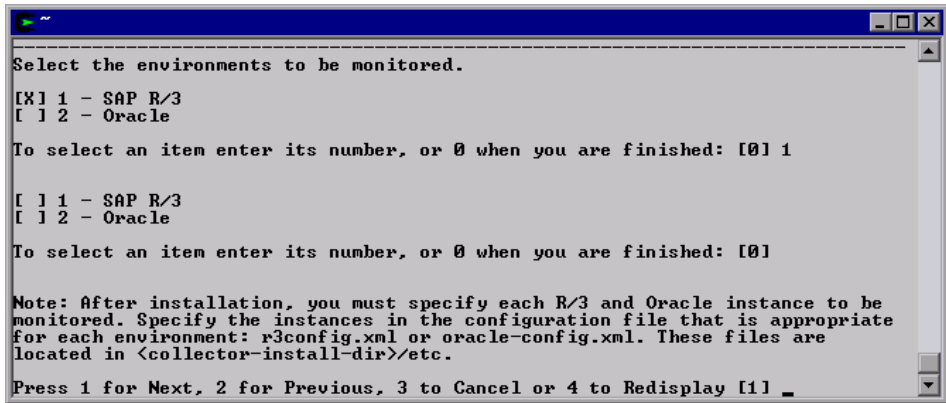


Assign a name to the Collector that will uniquely identify this specific Collector in the System Health Monitor. You use the System Health Monitor to verify the Collector installation and configuration. For more information, see “Verifying the Diagnostics Collector Installation” on page 82.

The following characters can be used in the name: - , _ , and all alphanumeric characters.

Select **Next** to continue.

5 Specify the active systems that the Collector will monitor.



Select the active systems that apply to this Collector. You can select either one or both options.

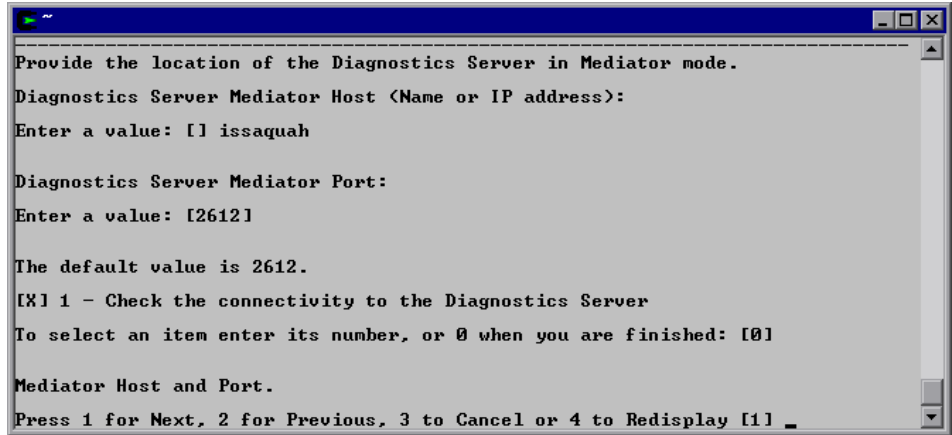
- To collect data in an SAP R/3 environment, select **SAP R/3**.
- To collect data on an Oracle 10g database server, select **Oracle**.

Important: After installation, you specify each of the SAP R/3 and Oracle instances to be monitored. These instances are manually defined in the XML files provided with the installation. For more information, see “Configuring the Active System Property Files” on page 79

Select **Next** to continue with the installation.

6 Provide the details for the Diagnostics Server in Mediator mode.

Provide the information that enables communication with the Diagnostics Server in Mediator mode.



If there is only one Diagnostics Server in the Diagnostics deployment where the Collector will run, enter the host name of the Diagnostics Server and its event port information.

If there is more than one Diagnostics Server in the deployment, enter the information for the Diagnostics Server in Mediator mode that is to receive the events from the Collector.

- a** Enter the host name or IP address of the host of the Diagnostics Server in Mediator mode.

Note: You should specify the fully qualified host name. In a mixed OS environment, where UNIX is one of the systems, this is essential for proper network routing.

- b** Enter the number of the port where the Diagnostics Server is listening for Collector communication. The default port is **2612**. If you have changed the port since the Diagnostics Server was installed, specify that port number instead of the default.

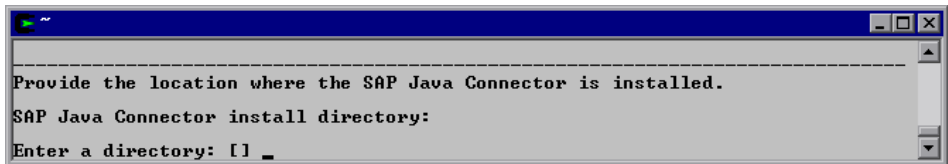
- If you want to make sure that the Diagnostics Server is running and accessible from the installation host, select **Check the connectivity to the Diagnostics Server**.

Select **Next** to continue with the installation.

If you instructed the installer to perform the test for connectivity, it tests the connectivity at this point. If there are negative results, it reports these before proceeding with the next installation step.

If you do not want to address these problems at this stage, clear the **Check the connectivity to the Diagnostics Server** option so that the installation can proceed.

7 Provide the location of the SAP Java Connector.



At the prompt, type the name of the directory where the SAP Java Connector is installed.

This directory must contain the following files:

- sapjco.jar
- librfc.dll
- sapjcorfc.dll

Note:

- ▶ If you do not know the SAP Java Connector directory name, contact your SAP representative.
 - ▶ If any of the files is missing from this directory, contact your SAP representative.
-

Select **Next** to continue with the installation.

8 Review the pre-installation summary.

The installation settings that you selected are displayed. Review the information to make sure that you are satisfied.

To change your settings, select **Previous** to return to the previous prompts.

To start the installation of the Collector, select **Next**.

9 Close the installation wizard.

When the installation has completed, review the post-installation summary information to make sure that the installation completed successfully.

Select **Finish** to exit the installation.

10 Configure the XML files for your active systems.

In step 5 you selected the active systems that will be monitored. For each of these active systems, you need to configure properties that enable the Collector host and the active system host to communicate.

For instructions on configuring the active system properties, see “Configuring the Active System Property Files” on page 79.

11 Verify the Collector installation.

Using the System Health Monitor, you can verify that the Collector is running as it should be. For details, see “Verifying the Diagnostics Collector Installation” on page 82.

Silent Installation of the Diagnostics Collector

A *silent installation* is an installation that is performed automatically, without the need for user interaction. In place of user input, the silent installation accepts input from a response file for each step of the installation.

For example, a system administrator who needs to deploy a component on multiple machines can create a response file that contains all the prerequisite configuration information, and then perform a silent installation on multiple machines. This eliminates the need to provide any manual input during the installation procedure.

Before you perform silent installations on multiple machines, you need to generate a response file that will provide input during the installation procedure. This response file can be used in all silent installations that require the same input during installation.

To generate a response file:

- Perform a regular installation with the following command-line option:

```
<installer> -options-record <responseFileName>
```

This creates a response file that includes all the information submitted during the installation.

To perform a silent installation:

- Perform a silent installation using the relevant response file.

You perform the silent installation with the **-silent** command-line option as follows:

```
<installer> -silent -options <responseFileName>
```

Configuring the Active System Property Files

When you install the Collector, you are asked to indicate the active systems that the Collector will monitor. After installation, you define instances of the active systems to be monitored. These instances are manually defined in the XML files provided with the installation. An instance definition in the XML file behaves like a probe on the instance of the active system.

Configuring SAP R3 Probes

SAP R3 system deployment can include one or more SAP R3 application instances. These instances together form an SAP R3 system.

Depending on user permissions, access to the system or application instances on the system may be direct or may require connection through the SAP Message Server. For each SAP R3 Probe that you define, you need to know what connection option is used.

You define and configure SAP R3 Probes in the `<collector_install_dir>\etc\r3config.xml` file. The layout, elements, and attributes of the xml file are described in the `<collector_install_dir>\etc\r3config.xsd`.

To configure an SAP R3 Probe:

- 1** Open `<collector_install_dir>\etc\r3config.xml`.
- 2** If you are defining an SAP R3 Probe where access to the SAP R3 instance is through the SAP Message Server, locate the section of code preceded by the following comment:

```
<!--
  Template to be used with the message server connection option.
-->
```

If you are defining an SAP R3 Probe where access to the SAP R3 instance is direct, locate the section of code preceded by the following comment:

```
<!--
  Template to be used with the direct connection option.
-->
```

- 3 Make a copy of the comment together with the template code below the comment, and paste it at the end of the file.
- 4 Comment out the original template code by typing <!-- in an empty line above the template code, and --> in an empty line thereafter.
- 5 In the copied code at the end of the file, alter the value of each property, as described in the following table and then save the file:

Property	Description	Value
r3system name	A logical name for the probe group under which this SAP Probe appears in the Diagnostics UI.	User-defined.
systemId	The ID of the SAP R3 system. Comprises 3 characters only.	Format: [XXX] Obtainable from the SAP system administrator.
client	The client name for the SAP R3 system.	Obtainable from the SAP system administrator.
user	The name of the user connecting to the SAP R3 system.	Obtainable from the SAP system administrator.
password	The password of the user connecting to the SAP R3 system.	Obtainable from the SAP system administrator.
messageServerHost (Message Server connection only)	The name of the SAP Message Server host machine.	Obtainable from the SAP system administrator.
r3Name (Message Server connection only)	Comprises 3 characters only.	Format: [XXX] Obtainable from the SAP system administrator.
group (Message Server connection only)		Obtainable from the SAP system administrator.

Configuring Oracle Probes

You define and configure the Oracle Probes in `<collector_install_dir>\etc\oracle-config.xml`. The layout, elements and attributes of the xml file are described in the `<collector_install_dir>\etc\oracle-config.xsd`.

To configure an Oracle Probe:

- 1 Open `<collector_install_dir>\etc\oracle-config.xml`.
- 2 Copy the template code, and paste it at the end of the file.
- 3 Comment out the template code by typing `<!--` in an empty line above the template code, and `-->` in an empty line thereafter.
- 4 In the copied code, alter the value of each property, as described in the following table and then save the file:

Properties	Description	Value
hostName	The name of the Oracle database server host machine.	Obtainable from the Oracle administrator.
portNumber	The number of the port where the Oracle database server listens for requests.	Default value: 1521
instanceName	The name given to the Oracle instance during installation of the Oracle database server.	Default value: Orcl Obtainable from the Oracle administrator.
userId	The ID of the user connecting to the Oracle database server. Note: The user needs at least CREATE SESSION and SELECT ANY DICTIONARY to collect performance metrics.	Obtainable from the Oracle administrator.
password	The password of the user connecting to the Oracle database server.	Obtainable from the Oracle administrator.

Properties	Description	Value
probeName	The logical name to represent this Oracle instance in the Diagnostics UI. This name must be unique.	User-defined. If this value is not defined, the same value given for instanceName will be used here.
probeGroupName	The logical name of the Probe group under which this Probe appears in the Diagnostics UI. This can be an existing Probe group, or you can define a new one.	User-defined. For example: Existing: Default New: Oracle

Verifying the Diagnostics Collector Installation

The Collector starts running automatically when the installation is complete. You can verify the Collector installation using the System Health Monitor. For detailed instructions about how to use the System Health Monitor, see Appendix C, “Using the System Health Monitor.”

If you have been following the recommended installation sequence, after installing the Collector you can verify the following:

- ▶ The Diagnostics Server in Commander mode was successfully installed.
- ▶ Where relevant, additional Diagnostics Servers in Mediator mode were successfully installed and are communicating with the Diagnostics Server in Commander mode.
- ▶ The Collector was successfully installed and has established connectivity with the relevant Diagnostics Server.

Depending on your deployment, the Collector is shown on the System Health Monitor as a child of either the Diagnostics Server in Commander mode or a Diagnostics Server in Mediator mode. The Collector is represented by an icon labelled <name of Collector>. The name of the collector is assigned during the installation procedure.



When the Collector has been stopped, the Collector node in the System Health monitor is grey. When the Collector has been started the node is green.

Starting and Stopping the Diagnostics Collector

Instructions for a Windows Machine

To start the Collector on a Windows machine:

Select **Start > Programs > Mercury Diagnostics Collector > Start Mercury Diagnostics Collector**.

To stop the Collector on a Windows machine:

Select **Start > Programs > Mercury Diagnostics Collector > Stop Mercury Diagnostics Collector**.

Instructions for a Solaris Machine

The *nanny* is a process that runs as a daemon to ensure that the Collector is always running. The following procedures start and stop the Collector using the *nanny*.

To start the Collector on a Solaris machine:

- 1 Ensure that the **M_LROOT** environment variable is defined as the root directory of the Collector.

For example, in *tssh*, you could enter the following:

```
$> setenv M_LROOT <collector_install_dir>/nanny/solaris
```

If the **M_LROOT** environment variable is not defined as the root directory, you will encounter the following error:

```
Warning : MDRV: cannot find lrun root directory . Please check your M_LROOT  
Unable to format message id [-10791]  
m_agent_daemon ( is down )
```

- 2 Change directories to **<collector_install_dir>/nanny/solaris/bin**.
- 3 Run **m_daemon_setup** with the **-install** option, as in the following example:

```
$> ./m_daemon_setup -install
```

To stop the Collector on a Solaris machine:

- 1 Change directories to **<collector_install_dir>/nanny/solaris/bin**.
- 2 Run **m_daemon_setup** with the **-remove** option, as in the following example:

```
$> ./m_daemon_setup -remove
```

Instructions for a UNIX Machine

The following procedures start and stop the Collector without using the nanny.

To start the Collector on a UNIX machine:

- ▶ Run `<collector_install_dir>/bin/collector.sh`.

To stop the Collector on a UNIX machine:

- ▶ Terminate the process using a utility such as “kill.”

Determining the Version of the Diagnostics Collector

When you request support, it is useful to know the version of the Diagnostics Collector.

The version number of the Collector can be found:

- ▶ in the `<collector_install_dir>\version.txt` file
- ▶ among the details of the Collector on the System Health Monitor

Uninstalling the Diagnostics Collector

To uninstall the Collector:

- ▶ On a Windows machine, run **uninstaller.exe** which is located in the `<collector_install_dir>_uninst` directory.
- ▶ On a UNIX machine, run **uninstall*** which is located in the `<collector_install_dir>_uninst` directory.

5

Installing the Mercury Diagnostics Probe for .NET

This chapter describes how to install a Mercury Diagnostics Probe for .NET.

This chapter describes:	On page:
About the Mercury Diagnostics Probe for .NET	87
Installing the .NET Probe	88
Verifying the .NET Probe Installation	100
Configuring the .NET Probe	102
Determining the Version of the .NET Probe	102
Uninstalling the .NET Probe	102

About the Mercury Diagnostics Probe for .NET

The Mercury Diagnostics Probe for .NET (.NET Probe) provides a low-overhead capture solution that works with Mercury's Application Delivery and Application Management products. The .NET Probe captures events from a .NET application and sends the event metrics to the Diagnostics Server.

The .NET Probe uses runtime instrumentation to capture method latency information from specified applications. By default, the .NET Probe captures methods from the ASP.NET and ADO tiers. The probe can be configured to capture custom business logic methods by adding instrumentation points to your application's **points** file.

An *instrumented method* is any method defined in any of the points files, for which the probe was configured. *Initialization* refers to the registration of the probe with the Diagnostics Server. *.NET Probe initialization* occurs only when an instrumented method is encountered for the first time. For ASP.NET applications, the probe is started the first time that a page in the application is requested after the Web publishing service has been started.

Installing the .NET Probe

The following section provides detailed instructions for installing the .NET Probe.

About the Installation

You install the .NET Probe on the host machine of the application that you wish to monitor.

For information about the recommended system configurations for hosting the .NET Probe, see “Requirements for the Mercury Diagnostics Probe for .NET Host” on page 10.

The default configuration of the .NET Probe has been set to monitor your application effectively, with very little impact on the performance of your application. For instructions on advanced .NET Probe configuration, see Chapter 19, “Advanced .NET Probe Configuration.”

Starting the Installation

Important: .NET Framework 1.1 or later needs to be installed on your machine before you run the .NET Probe installation.

You start the installer from the product CD or from the Diagnostics Downloads page in Mercury Business Availability Center.

To start the installation from the product CD:

- 1** Insert the Mercury Diagnostics installation CD for Windows into a CD-ROM drive. If the installer does not start automatically, double-click **setup.exe** in the root folder of the CD-ROM.

The installer displays the Mercury Diagnostics main installation menu.

- 2** Click **Mercury Diagnostics Probe for .NET** to initiate the installer for the .NET Probe.

To install the .NET Probe from a location other than the CD:

- 1** Locate the file **Mercury .NET Probe.msi** in the **Diagnostics _Probes** directory on the CD, copy it to the new location, and then run it.
- 2** Continue with “Running the Installation” on page 90.

To start the installation from the downloads page (for Mercury Business Availability Center users):

- 1** Select **Admin > Diagnostics** from the top menu in Mercury Business Availability Center, and click the **Downloads** tab.
- 2** On the Downloads page, click the appropriate link to download the .NET Probe installer for Windows.

Note: The probe installers are available in Mercury Business Availability Center only if you provided the path to the probe installers directory during the installation of the Diagnostics Server in Commander mode.

Continue with “Running the Installation” on page 90.

Running the Installation

After you have launched the installer, you are ready to begin the main installation procedure.

To install the .NET Probe on a Windows machine:

- 1 Accept the software license agreement.

Read the agreement and select **I accept the terms of the license agreement**.

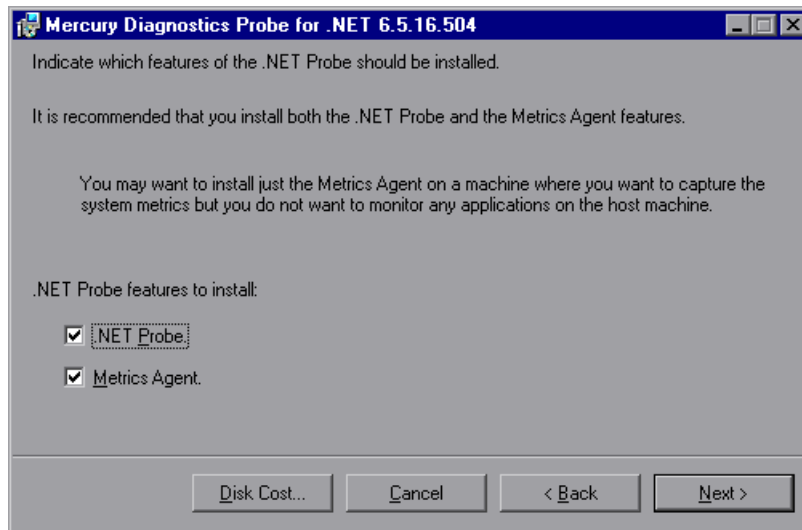
Click **Next** to proceed.

- 2 Provide the location where you want the probe installed.

Accept the default directory or select a different location either by typing in the path to the installation directory into the **Installation Directory Name** box, or by clicking **Browse** to navigate to the installation directory.

Click **Next** to continue.

- 3 Select the .NET Probe features that you want to install.

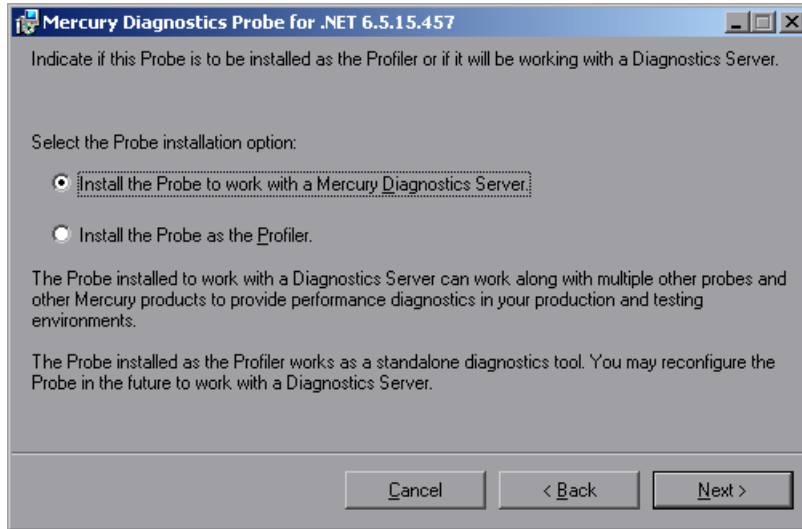


It is recommended that you install both the **.NET Probe** and **Metrics Agent** features.

To capture system metrics on the host machine without monitoring any applications, select **Metrics Agent** only.

If you want to check the amount of available disk space on the drives of the host, click **Disk Cost**. Use this functionality to make sure that there is enough room for the probe installation.

- 4 Select whether you want to install the probe as the Profiler only, or with a Mercury Diagnostics Server.



Make the selection that is appropriate for the environment where you will be using the probe:

- To use the probe with other probes and other Mercury products to provide performance metrics in your production or testing environment, select **Install the Probe to work with a Mercury Diagnostics Server**.
- To use the probe as a stand-alone diagnostics tool for a development environment, select **Install the Probe as the Profiler**.

Click **Next** to continue.

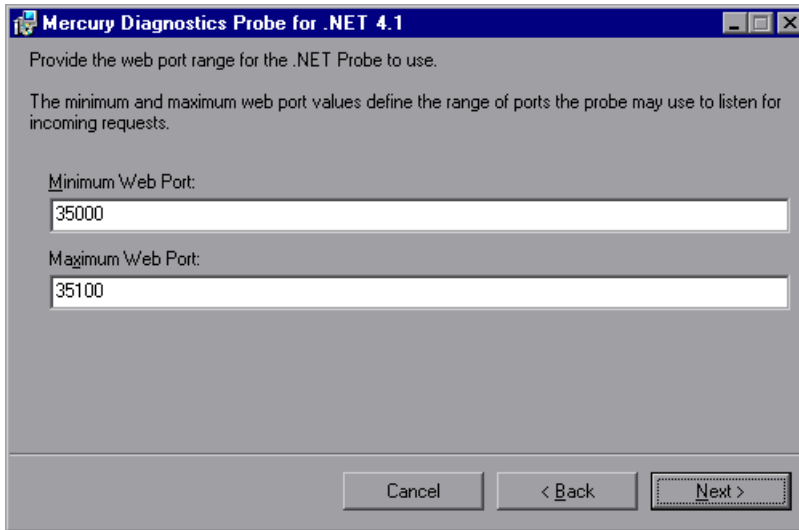
Next Step: At this stage, the installation procedure differs, depending on the environment in which you are installing the probe.

- ▶ If you are installing the probe as the Profiler only, continue with “Installing the Probe as a Profiler Only” on page 92.
 - ▶ If you are installing the probe to work with a Mercury Diagnostics Server, continue with “Installing the Probe to Work with a Mercury Diagnostics Server” on page 94.
-

Installing the Probe as a Profiler Only

If you are installing the probe to work as a Profiler only, continue with the following procedure.

- 1 Provide the Web port range for the .NET Probe to use.



Mercury Diagnostics Probe for .NET 4.1

Provide the web port range for the .NET Probe to use.

The minimum and maximum web port values define the range of ports the probe may use to listen for incoming requests.

Minimum Web Port:
35000

Magimum Web Port:
35100

Cancel < Back Next >

- ▶ **Web Port Min.** Type the lowest port number, in a range of ports on the probe host, that you want to assign to the probe.

- **Web Port Max.** Type the highest port number, in a range of ports on the probe host, that you want to assign to the probe.

Note: The default range is from 35000 to 35100 (inclusive).

The upper and lower limits of the Web Port Range are defined by the **Web Port Min** and **Web Port Max** fields. The Web Port Range contains the ports that the probe can use.

When a probe is started, it attempts to find an unused port from within this range; starting from the lowest port number in the range and working its way up to the highest. Ports within the range may already be in use if another probe or application has previously claimed them.

The minimum size for the port range is equal to the maximum number of probes that will be concurrently running on the probe's host.

Considerations when setting the Web Port Range:

- If the probes are working with ASP.NET applications, it is recommended that you double the number of ports to account for ASP.NET's appdomain recycling.
- If you have a firewall between the probe and a component that will be communicating with the probe, you must open the firewall for the ports within the range. For this reason you may want to adjust the range to be just big enough.

Click **Next** to continue.

- 2** The pre-installation summary screen opens. Click **Back** to make any changes. Click **Install** to start the .NET Probe installation.
- 3** When the .NET Probe installation completes, instructions for configuring the probe are displayed in the installer window.

The .NET Probe configuration is discussed in greater detail in Chapter 19, "Advanced .NET Probe Configuration."

Click **Finish** to exit the installer.

- 4 After you exit the installer, you must restart either the IIS or the Web publishing service before you can use the .NET Probe with ASP.NET applications.

Installing the Probe to Work with a Mercury Diagnostics Server

If you are installing the probe to work with a Mercury Diagnostics Server, continue with the following procedure.

- 1 Enter the probe name and probe group name.

Mercury Diagnostics Probe for .NET 6.5.20.642

Enter the Probe Name and Probe Group Name.

The Probe Name uniquely identifies each probe. The default is the name of the application which loads the probe.

Probe Name (Leave blank to accept default based on application name):

A Probe Group is a logical collection of probes that are monitored by the same Diagnostics Server. The default value is "Default".

Probe Group Name:

Default

Cancel < Back Next >

- **Probe Name.** The name that identifies the probe within Mercury Diagnostics. If you leave this field blank, the .NET Probe will auto-generate a probe name based on the application's domain name.

Important: It is recommended that you leave **Probe Name** blank and allow the probe to auto-generate the probe name. Read the following information carefully if you decide to enter your own probe name.

Considerations when entering a probe name:

- ▶ Valid characters that can appear in the probe name are: letters, digits, dashes, underscores, and periods.
- ▶ Assign a probe name that will help you recognize the application that the probe is monitoring, and the type of probe it is.

For example, the probe name for .NET Probe installed to monitor the application named PetWorld can be:

PetWorld_Dotnet_Probe

- ▶ When you specify a probe name, all of the probes on the host are forced to use the same probe name. If you want to override the default names, use the following substitution macros to enhance the name at run time:

\$(MACHINENAME): Machine's host name

\$(APPDOMAIN): Application's domain name

\$(PID): Application's process ID

The default probe name auto-generated by the probe when the probe name field is left blank is equivalent to specifying “\$(APPDOMAIN).NET”.

To obtain behavior compatible with version 3.3.x of the probe, specify “\$(MACHINENAME).\$(APPDOMAIN).NET”.

- **Probe Group:** Probe groups are logical groupings of probes that report to the same Diagnostics Server. The performance metrics for a probe group are tracked, and can be displayed on many of the Diagnostics views.

You may want to assign all of the probes for a particular enterprise application to a single probe group so that you can monitor the performance of the group as well as the individual probes.

The default value for the probe group name is Default.

Note: The probe group name is case-sensitive.

Click **Next** to continue.

- 2 Provide the information needed to enable the .NET Probe to communicate with the Diagnostics Server in Mediator mode.

Mercury Diagnostics Probe for .NET 6.5.20.642

Provide the location of the Diagnostics Server in Mediator mode.

Diagnostics Server Mediator Host (Name or IP address):

Diagnostics Server Mediator Port (Default is 2612):

Test

Cancel < Back Next >

- a** In the **Diagnostics Server Mediator Host** box, type the host name or IP address of the host for the Diagnostics Server in Mediator mode.

You should specify the fully qualified host name, not just the simple host name. In a mixed OS environment, where UNIX is one of the systems, this is essential for proper network routing.

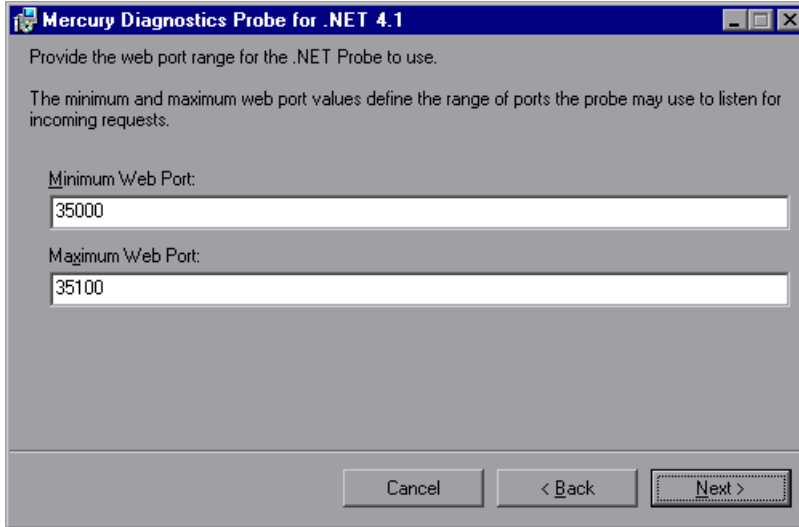
- b** In the **Diagnostics Server Mediator Port** box, type the port number where the Diagnostics Server in Mediator mode is listening for probe communication. The default port number is 2612. If you have changed the port since the Diagnostics Server was installed, you should specify that port number here instead of the default.

- c** If you want to perform a connectivity check to make sure that the Diagnostics Server is running and accessible from the installation host, click **Test**.

The connectivity check lets you know right away if you have made an error in the information that you provided about the Diagnostics Server in Mediator mode, or if there is a connection problem between the Diagnostics Server's host and the probe's host. If the connection to the Diagnostics Server in Mediator mode host cannot be resolved, an error message is displayed.

- d** Click **Next** to proceed.

3 Provide the Web port range for the .NET Probe to use.



- **Web Port Min.** Type the lowest port number, in a range of ports on the probe host, that you want to assign to the probe.
- **Web Port Max.** Type the highest port number, in a range of ports on the probe host, that you want to assign to the probe.

Note: The default range is from 35000 to 35100 (inclusive).

The upper and lower limits of the Web Port Range are defined by the **Web Port Min** and **Web Port Max** fields. The Web Port Range contains the ports that the probe can use.

When a probe is started, it attempts to find an unused port from within this range; starting from the lowest port number in the range and working its way up to the highest. Ports within the range may already be in use if another probe or application has previously claimed them.

The minimum size for the port range is equal to the maximum number of probes that will be concurrently running on the probe's host.

Considerations when setting the Web Port Range:

- ▶ If the probes are working with ASP.NET applications, it is recommended that you double the number of ports to account for ASP.NET's appdomain recycling.
- ▶ If you have a firewall between the probe and a component that will be communicating with the probe, you must open the firewall for the ports within the range. For this reason you may want to adjust the range to be just big enough.

Click **Next** to continue.

- 4** The pre-installation summary screen opens. Click **Back** to make any changes. Click **Install** to start the .NET Probe installation.
- 5** When the .NET Probe installation completes, instructions for configuring the probe are displayed in the installer window.

The .NET Probe configuration is discussed in greater detail in Chapter 19, "Advanced .NET Probe Configuration."

Click **Finish** to exit the installer.

- 6** After you exit the installer, you must restart either the IIS or the Web publishing service before you can use the .NET Probe with ASP.NET applications.
- 7** You can verify the installation as described in "Verifying the .NET Probe Installation" on page 100.

Note: The default configuration of the .NET Probe has been set to monitor your application effectively, with very little impact on the performance of your application. For more information, see "Configuring the .NET Probe" on page 102.

Verifying the .NET Probe Installation

Important: This section applies only if you installed the probe to work with a Mercury Diagnostics Server.

Use the System Health Monitor to verify the installation of the .NET Probe. For instructions on how to use the System Health Monitor, see Appendix C, “Using the System Health Monitor.”

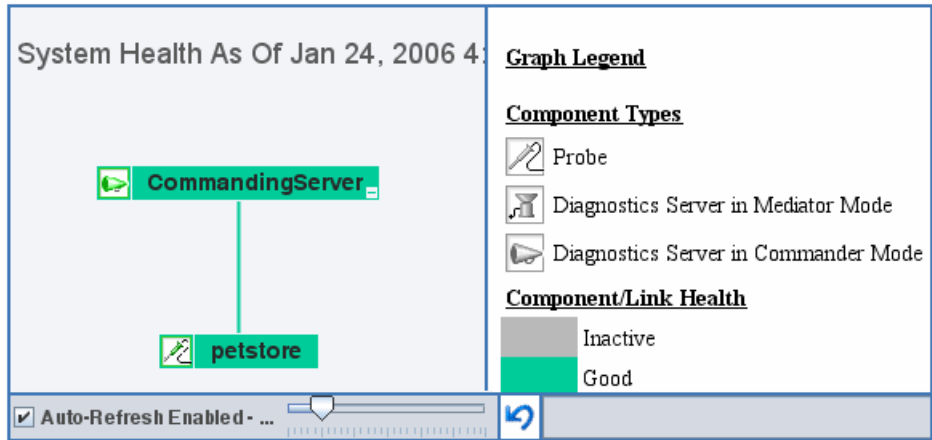
If you have been following the recommended installation sequence, after you have installed the .NET Probe and restarted the instrumented application, you will be able to verify the following:

- ▶ The Diagnostics Server in Commander mode was successfully installed.
- ▶ Where relevant, additional Diagnostics Servers in Mediator mode were successfully installed and are communicating with the Diagnostics Server in Commander mode.
- ▶ The .NET Probe was successfully installed and is communicating with the Diagnostics Server.

Note: The .NET Probe will not be displayed in System Health when first installed because the probe does not register with the Diagnostics Server until it is started. The probe is started and is registered with the Diagnostics Server when the instrumented application is run. For ASP.NET applications, this happens the first time that a page is requested for the instrumented application.

The default logging level for the .NET Probe has been set to **info** so that if the probe is not displayed in System Health when you believe that it should be, you can check the log file to help troubleshoot the problem. If you do not see a log file you can check the Windows Event Viewer.

Depending on your deployment, the new .NET Probe is shown on the System Health Monitor to be reporting to either the Diagnostics Server in Commander mode or the Diagnostics Server in Mediator mode.



Configuring the .NET Probe

The default configuration of the .NET Probe allows you to begin capturing performance metrics for your applications. Moreover, the probe's configuration can be customized to suit the configuration of your environment and the performance issues that you would like to diagnose.

The installer configures your ASP.NET applications and the .NET Probe so that they can work together to capture the basic workload of your applications. It is possible that one or more of your ASP.NET applications has been deployed in a manner that prevents the installer from detecting it, or you may want to enhance the standard instrumentation to capture the performance metrics for the custom classes in your application.

The .NET Probe is installed with a tool that helps you discover the methods and classes in an application. For information about the Reflector tool, see “Customizing the Instrumentation for ASP.NET Applications” on page 341.

For instructions on advanced .NET Probe configuration, see Chapter 19, “Advanced .NET Probe Configuration.”

Determining the Version of the .NET Probe

When you request support, it is useful to know the version of the Diagnostics components that you have installed.

To determine the version of the .NET Probe:

- Right-click the file `<probe_install_dir>\bin\Mercury.Profiler.dll`, and display the component version information by selecting **Properties** from the menu.

Alternatively, you can determine the version of the .NET Probe using the System Health Monitor. For more information, see Appendix C, “Using the System Health Monitor.”

Uninstalling the .NET Probe

To uninstall the .NET Probe, use the **Add/Remove Programs** utility in Windows.

6

Installing the Mercury Diagnostics Probe for J2EE

This chapter provides instructions for installing a Mercury Diagnostics Probe for J2EE (J2EE Probe) on Windows machines, UNIX machines, and z/OS mainframe machines. Instructions have also been provided for using a generic installer to install the J2EE Probe on other platforms.

This chapter describes:	On page:
About Installing and Configuring the J2EE Probe	104
Installing the J2EE Probe on a Windows Machine	105
Installing the J2EE Probe on a UNIX Machine	122
Installing the J2EE Probe on a z/OS Mainframe	137
Installing the J2EE Probe Using the Generic Installer	139
Silent Installation of the J2EE Probe	141

About Installing and Configuring the J2EE Probe

Before you can use the J2EE Probe to monitor your application, you need to perform the following operations:

1 Install the J2EE Probe.

The J2EE Probe is installed on the machine hosting the application that you wish to monitor. For more information about installing the J2EE Probe, see one of the following sections, depending on your operating system and method of installation:

- ▶ “Installing the J2EE Probe on a Windows Machine” on page 105
- ▶ “Installing the J2EE Probe on a UNIX Machine” on page 122
- ▶ “Installing the J2EE Probe on a z/OS Mainframe” on page 137
- ▶ “Installing the J2EE Probe Using the Generic Installer” on page 139
- ▶ “Silent Installation of the J2EE Probe” on page 141

2 Configure the J2EE Probe.

The probe is automatically configured during the installation process. In certain situations however, you may be required to change these default settings or manually configure the probe. For more information, see “Configuring the Probe Manually” on page 144.

3 Configure the application server.

In order to allow the probe to monitor your application, you need to instrument the JRE and manually modify the application startup script. For more information, see “Configuring the Application Server” on page 147.

Note: For information about the recommended system configurations for hosting the J2EE Probe, see “Requirements for the Mercury Diagnostics Probe for J2EE Host” on page 8.

Installing the J2EE Probe on a Windows Machine

This section provides detailed instructions for installing the J2EE Probe on a Windows machine. These instructions also apply when you are installing the probe on a UNIX machine using the graphical installer.

This section includes:

Launching the Installer

The installer can be launched from the product CD or from the Diagnostics Downloads page in Mercury Business Availability Center.

To launch the installer from the product CD:

- 1 Insert the Mercury Diagnostics installation CD for Windows into a CD-ROM drive. If the installer does not launch automatically, double-click **setup.exe** in the root folder of the CD.

The installer displays the Mercury Diagnostics main installation menu.

- 2 Click **Mercury Diagnostics Probe for J2EE** to initiate the installer for the J2EE Probe.

To install the probe from a location other than the product CD:

- 1 Locate the executable file **J2EEProbeSetup_win.exe** file in the **Diagnostics_Probes** directory on the CD, copy it to the new location, and then run it.
- 2 Continue with “Running the Installation” on page 106.

To launch the installer from the downloads page, for Mercury Business Availability Center users:

- 1** Select **Admin > Diagnostics** from the top menu in Mercury Business Availability Center, and click the **Downloads** tab.
- 2** On the Downloads page, click the appropriate link to download the J2EE Probe installer for Windows.

Note: The probe installers are available in Mercury Business Availability Center only if you provided the path to the probe installers directory during the installation of the Diagnostics Server in Commander mode

Continue with “Running the Installation” on page 106.

Running the Installation

After you have launched the installer, the software license agreement opens and you are ready to run the installation.

To install the J2EE Probe on a Windows machine:

- 1** **Accept the software license agreement.**

Read the agreement and select **I accept the terms of the license agreement.**

Click **Next** to proceed.

2 Specify the location where you want to install the probe.

Accept the default directory or select a different location either by typing the path to the installation directory into the **Installation Directory Name** box, or by clicking **Browse** to navigate to the installation directory.

If there is a pre-existing installation of the probe on the host machine, you can upgrade the existing probe or install the probe to a different installation directory.

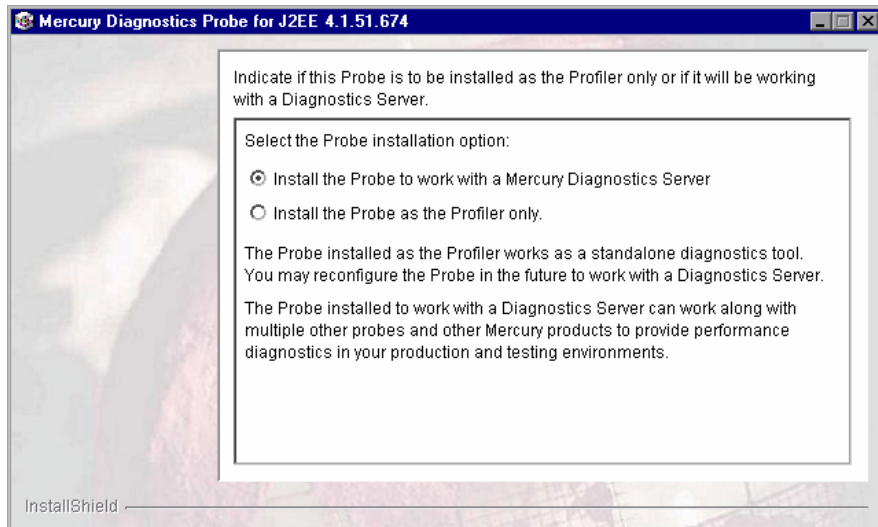
- ▶ To upgrade the existing probe installation, ensure that the installation directory of the existing probe has been specified in the **Installation Directory Name** box. For more information about upgrading Diagnostics components, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

When the installer runs, it will back up any property settings of the existing J2EE Probe installation to a directory called **etc.old**.

- ▶ To install the probe in an installation directory that is different from the existing probe’s directory, ensure that the installation directory of the existing probe has not been specified in the **Installation Directory Name** box.

Click **Next** to proceed.

3 Select whether you want to install the probe with a Mercury Diagnostics Server, or as the Profiler only.



- ▶ If you want to use the probe with other probes and other Mercury products to provide performance diagnostics in your production or testing environment, select **Install the Probe to work with a Mercury Diagnostics Server**.
- ▶ If you want to use the probe as a stand-alone diagnostics tool for a development environment, select **Install the Probe as the Profiler only**.

Note: When the probe is installed to work with a Mercury Diagnostics Server, the configuration is set such that the J2EE Diagnostics Profiler is available as well. You can drill down from the Mercury Diagnostics views into the tabs of the J2EE Diagnostics Profiler or you can access the Profiler in stand-alone mode.

When the probe is installed as a Profiler only, you can reconfigure the probe in the future to work with the Diagnostics Server.

Click **Next** to proceed.

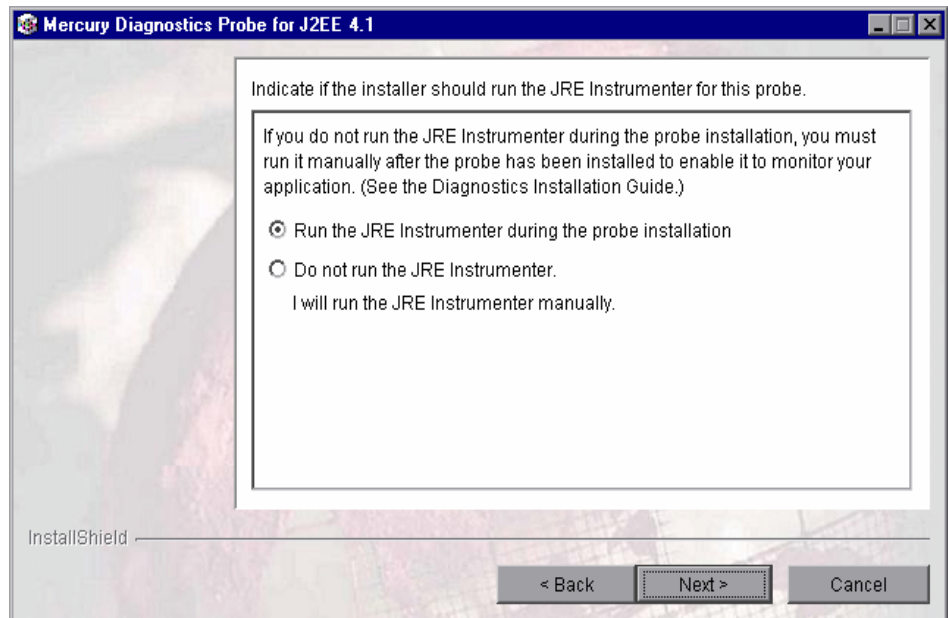
Next Step: At this stage, the installation procedure differs, depending on the environment in which you are installing the probe.

- ▶ If you are installing the probe as a Profiler only, continue with “Installing the Probe as a Profiler Only” on page 109.
 - ▶ If you are installing the probe to work with a Mercury Diagnostics Server, continue with “Installing the Probe to Work with a Mercury Diagnostics Server” on page 113.
-

Installing the Probe as a Profiler Only

If you are installing the probe to work as a Profiler only, continue with the following procedure.

- 1** Indicate whether or not you want to run the JRE Instrumenter during the installation of the probe.



You can run the instrumenter now, or you can skip this step by selecting **Do not run the JRE Instrumenter**. If you choose to skip this step now, you must run the JRE Instrumenter manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.

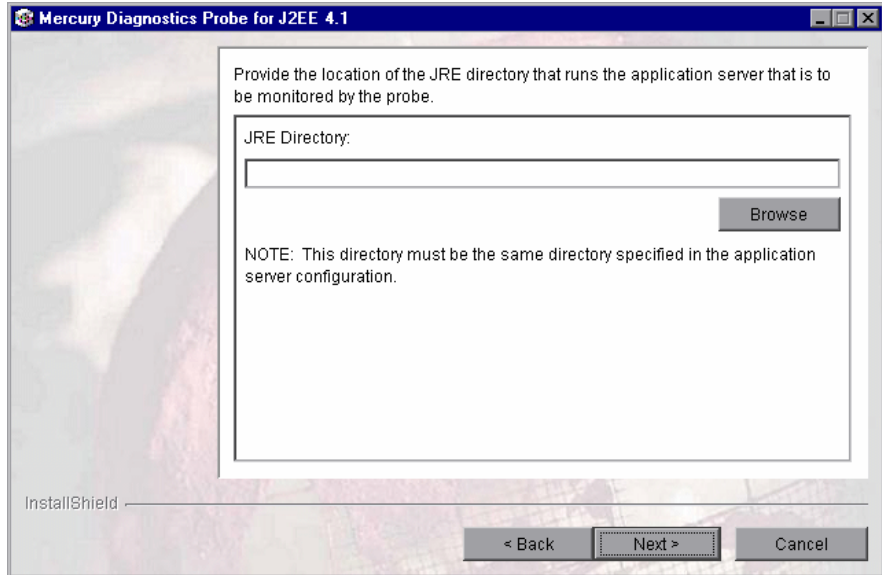
Note: If you are installing this probe to work with WebSphere in an Integrated Development Environment (IDE), select **Do not run the JRE Instrumenter**. You will run the JRE Instrumenter manually. See “Running the JRE Instrumenter for WebSphere IDE” on page 183.

Click **Next** to proceed.

Next Step: If you chose to run the JRE instrumenter now, continue below. If you chose not to run the JRE instrumenter now, skip to step 3.

2 Enter the JRE directory.

If you indicated that you want to run the JRE Instrumenter during the installation of the probe, enter the JRE directory.



In the **JRE Directory** box, type the path to the JRE bin folder where the JRE used by the monitored application server can be found. For example, if you have installed WebLogic 8.1 (using Sun JDK) in your **D:\bea** directory, the **java.exe** file can be found in the bin folder of the JRE directory, **D:\bea\jdk142_05\JRE**.

Click **Next** to proceed.

3 Confirm the pre-installation summary information.

The pre-installation summary screen opens. Review the summary information. Click **Back** to make any changes. Click **Next** to proceed.

The installation process begins, and a progress bar indicates how the installation is proceeding.

4 Close the installation wizard.

After installation, a status message notifies you that the installation was successfully completed.

Click **Finish**.

5 Configure the Application Server

Before you can use the probe to monitor your application, you need to instrument the JRE and manually modify the application startup script.

- a** If you chose to skip running the JRE Instrumenter during this installation, you need to run it manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.
- b** Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”

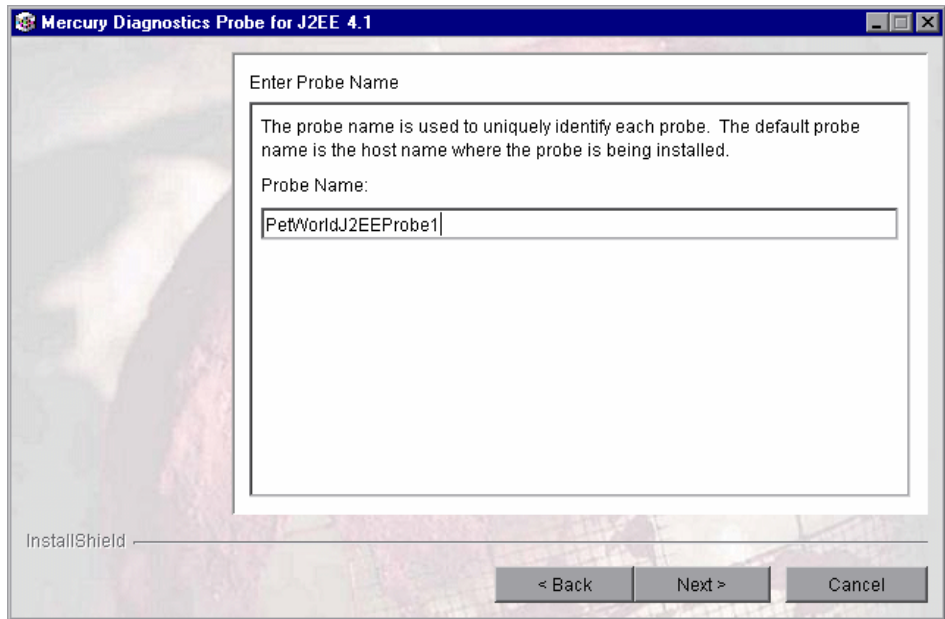
Note: The probe is automatically configured during the installation process. In certain situations however, you may be required to change these default settings or to manually configure the probe. For more information, see “Configuring the Probe Manually” on page 144.

For situations that require more advanced configurations of the probe and application server, see Chapter 21, “Advanced J2EE Probe and Application Server Configuration.”

Installing the Probe to Work with a Mercury Diagnostics Server

If you are installing the probe to work with a Mercury Diagnostics Server, continue with the following procedure.

1 Assign a name for the probe.



Type a unique name that will be used to identify the probe within Mercury Diagnostics. The following characters can be used in the name: - , _ , and all alphanumeric characters.

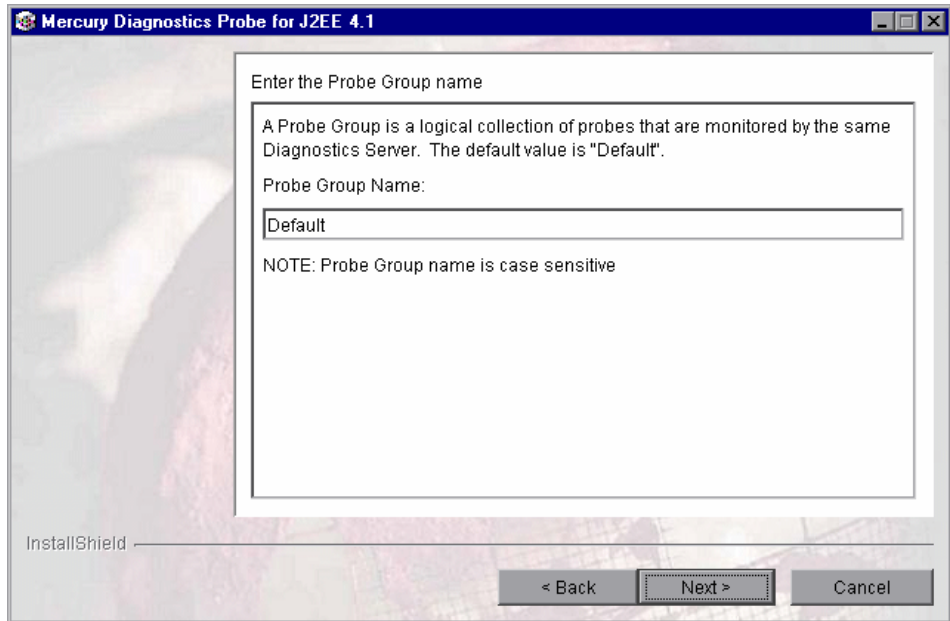
When assigning a name to a probe, choose a name that will help you to recognize the application that the probe is monitoring, and the type of the probe (in this case, a J2EE Probe).

For example, the probe name for the first J2EE Probe installed, that will be monitoring an application named `PetWorld`, could be:

`PetWorldJ2EEProbe1`

Click **Next** to proceed.

2 Assign a name for the probe group.



Probe groups are logical collections of probes that are monitored by the same Diagnostics Server. The performance metrics for a probe group are tracked and can be displayed on many of the Diagnostics views.

For example, you may want to assign all of the probes for a particular enterprise application to a single probe group so that you can monitor the performance of the group as well as the individual probes.

Type a name for the probe group, or accept the default.

Note: The probe group name is case-sensitive.

Click **Next** to proceed.

3 Enter the details for the Diagnostics Server in Mediator mode.

Enter the necessary information to enable communication with the Diagnostics Server in Mediator mode.

Mercury Diagnostics Probe for J2EE 4.1

Provide the location of the Diagnostics Server in Mediator mode.

Diagnostics Server Mediator Host (Name or IP address):
pen

Diagnostics Server Mediator Port:
2612

The default value is 2612.

Check the connectivity to the Diagnostics Server Mediator Host and Port.

InstallShield

< Back Next > Cancel

Note: If there is only one Diagnostics Server in the Diagnostics deployment where the probe will run, enter the Diagnostics Server's host name and event port information here.

When there is more than one Diagnostics Server in your deployment, enter the information for the Diagnostics Server in Mediator mode that is to receive the events from the probe.

- a** In the **Diagnostics Server Mediator Host** box, type the host name or IP address of the host for the Diagnostics Server in Mediator mode.

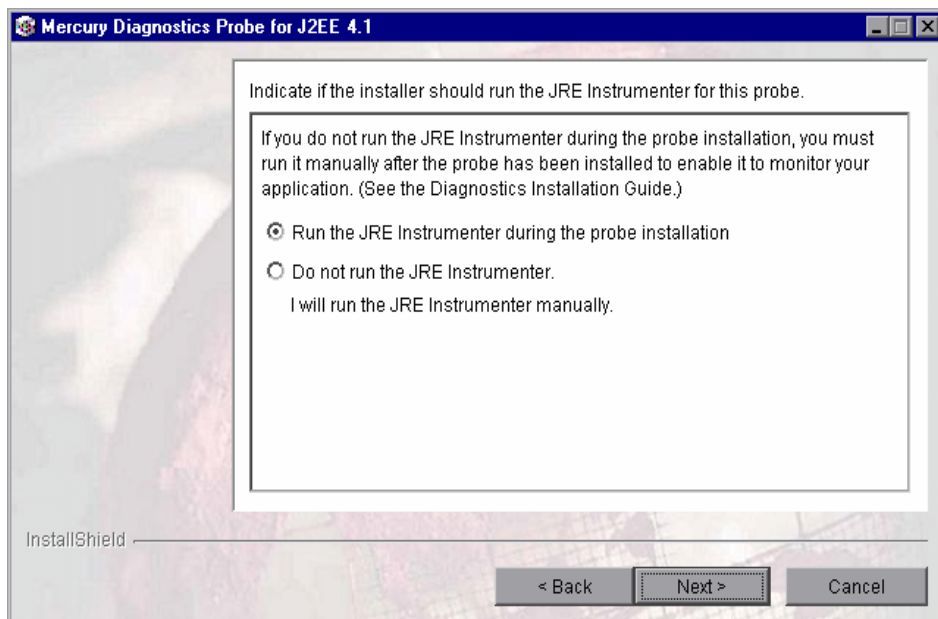
You should specify the fully qualified host name, not just the simple host name. In a mixed OS environment, where UNIX is one of the systems, this is essential for proper network routing.

- b** In the **Diagnostics Server Mediator Port** box, type the port number where the Diagnostics Server in Mediator mode is listening for probe communication. The default port number is 2612. If you have changed the port since the Diagnostics Server was installed, you should specify that port number here instead of the default.
- c** If you want to make sure that the specified Diagnostics Server is running and accessible from the installation host, select **Check the connectivity to the Diagnostics Server Mediator Host and Port**.

Click **Next** to proceed.

If you selected **Check the connectivity to the Diagnostics Server Mediator Host and Port** and connectivity problems were encountered, the installer provides you with the results of the connectivity check. If you do not want to address these problems at this stage, clear the check box, proceed with the installation, and address the problem later.

4 Indicate whether or not you want to run the JRE Instrumenter during the installation of the probe.



You can run the instrumenter now, or you can skip this step by selecting **Do not run the JRE Instrumenter**. If you choose to skip this step now, you must run the JRE Instrumenter manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.

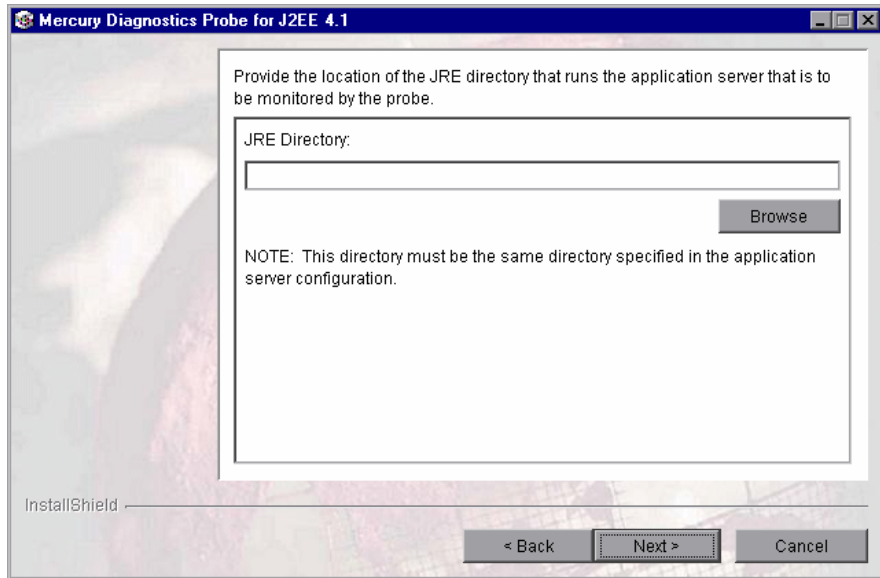
Note: If you are installing this probe to work with WebSphere in an Integrated Development Environment (IDE), select **Do not run the JRE Instrumenter**. You will run the JRE Instrumenter manually. See “Running the JRE Instrumenter for WebSphere IDE” on page 183.

Click **Next** to proceed.

Next Step: If you chose to run the JRE instrumenter now, continue below. If you chose not to run the JRE instrumenter now, skip to step 6.

5 Enter the JRE directory.

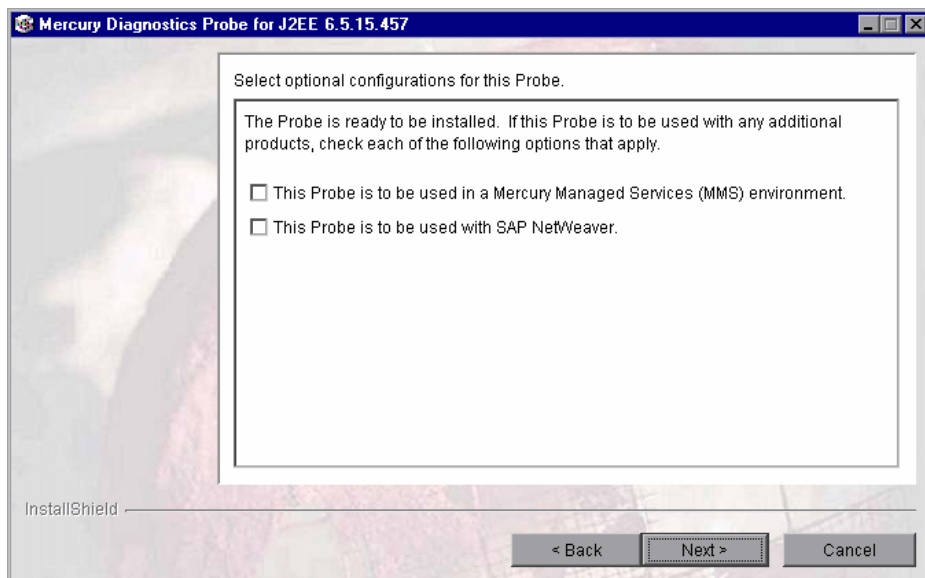
If you indicated that you want to run the JRE Instrumenter during the installation of the probe, enter the JRE directory.



In the **JRE Directory** box, type the path to the JRE bin folder where the JRE used by the monitored application server can be found. For example, if you have installed WebLogic 8.1 (using Sun JDK) in your **D:\bea** directory, the **java.exe** file can be found in the bin folder of the JRE directory, **D:\bea\jdk142_05\JRE**.

Click **Next** to proceed.

6 Select the configuration options if the probe is to be used with additional products.

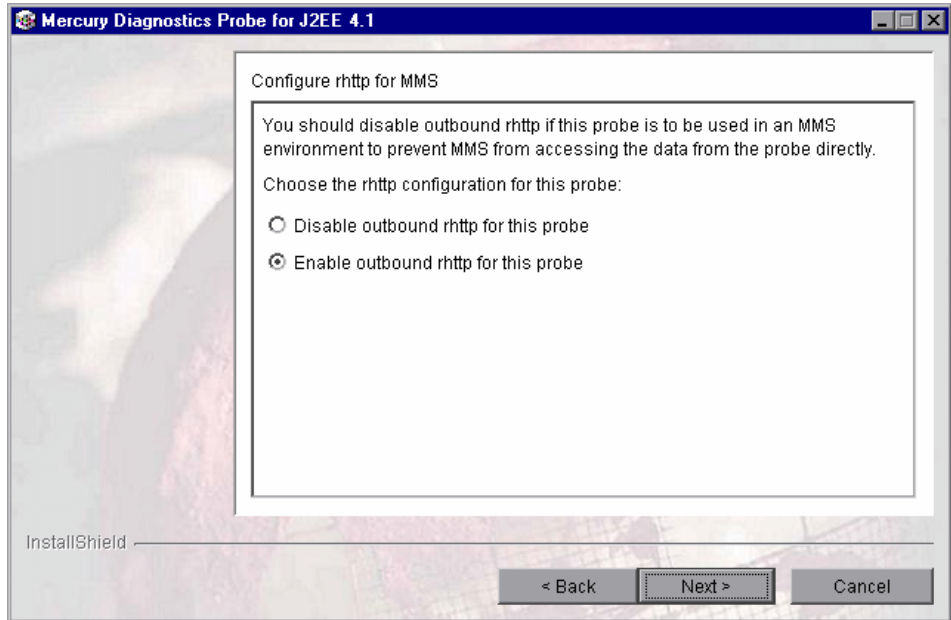


Indicate whether you will be using the probe within a Mercury Managed Services (MMS) environment, and whether you will be using the probe with a SAP NetWeaver Web application server.

Click **Next** to proceed.

7 Configure the Mercury Managed Services environment.

If you indicated that the probe is to be used in a Mercury Managed Services environment (see step 6), you are given the option to disable outbound rhttp (reverse http).



The Diagnostics Server uses rhttp to query metrics from the J2EE Probe. You should disable the probe's outgoing rhttp to prevent Mercury Managed Services from accessing the data directly from the probe.

Select the appropriate option and click **Next**.

8 Confirm the pre-installation summary information.

The pre-installation summary screen opens. Review the summary information. Click **Back** to make any changes.

Click **Next** to proceed.

The installation process begins, and a progress bar indicates how the installation is proceeding.

9 Close the installation wizard.

After installation, a status message notifies you that the installation was successfully completed.

Click **Finish** to close the installation wizard.

10 Configure the Application Server

Before you can use the probe to monitor your application, you need to instrument the JRE and manually modify the application startup script.

- a** If you chose to skip running the JRE Instrumenter during this installation, you need to run it manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.
- b** Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”

Note: The probe is automatically configured during the installation process. In certain situations however, you may be required to change these default settings or to manually configure the probe. For more information, see “Configuring the Probe Manually” on page 144.

For situations that require more advanced configurations of the probe and application server, see Chapter 21, “Advanced J2EE Probe and Application Server Configuration.”

11 Verify the probe installation.

If you are installing this probe to work with a Mercury Diagnostics Server, you can verify the probe installation as described in “Verifying the J2EE Probe Installation” on page 148.

Installing the J2EE Probe on a UNIX Machine

J2EE Probe installers have been provided for several UNIX platforms. The following instructions provide you with the steps necessary to install the J2EE Probe in most UNIX environments using either a graphical mode installation or a console mode installation.

The installer screens that are displayed in a graphical mode installation are the same as those documented for the Windows installer in “Installing the J2EE Probe on a Windows Machine” on page 105.

In some instances, you may not be able to use the regular UNIX installers. In these cases, you should use the Generic installer as described in “Installing the J2EE Probe Using the Generic Installer” on page 139.

The instructions and screen shots that follow are for a probe installation on a Solaris machine. These same instructions should apply for the other certified UNIX platforms.

Launching the Installer

The Installer can be launched from the product CD or from the Diagnostics Downloads page in Mercury Business Availability Center.

To launch the installer from the product CD:

- 1 Copy the installer from the **Mercury Diagnostics installation CD for <UNIX version>\Diagnostics_Probes** directory to the machine where the probe is to be installed.
- 2 Continue with “Running the Installation” on page 123.

To launch the installer from the downloads page (for Mercury Business Availability Center users):

- 1 Select **Admin > Diagnostics** from the top menu in Mercury Business Availability Center, and click the **Downloads** tab.
- 2 On the Downloads page, click the link to the installer that is appropriate for your environment and save it to the machine where the probe is to be installed.

Note: The probe installers are available in Mercury Business Availability Center only if you provided the path to the probe installers directory during the installation of the Diagnostics Server in Commander mode.

Continue with “Running the Installation” on page 123.

Running the Installation

After you have copied the installer to the machine where the probe is to be installed, you are ready to run the installation.

Note: The following instructions assume an understanding of UNIX console screens and commands. For more information about UNIX screens and commands, see Appendix H, “Using UNIX Commands.”

To install the J2EE Probe on a Unix machine:

1 Run the installer.

Where necessary, change the mode of the installer file to make it executable.

- To run the installer in console mode, enter the following at the UNIX command prompt:

```
./<installer> -console
```

The installer displays the installation prompts in console mode as shown in the following step.

- ▶ To run the installer in the graphical mode enter the following at the UNIX command prompt:

```
./<installer>
```

The installer displays the same screens that are displayed for the Windows installer, as shown in “Installing the J2EE Probe on a Windows Machine” on page 105.

2 Accept the software agreement.

The software license agreement is displayed.

Read the agreement. As you read, you can press **ENTER** to move to the next page of text, or type **q** to jump to the end of the license agreement.

Accept the terms of the agreement, and select **Next** to continue with the installation.

3 Specify the location where you want to install the probe.

At the **Directory Name** prompt, accept the default installation location shown in brackets, or enter the path to a different location.

If there is a pre-existing installation of the probe on the host machine, you can upgrade the existing probe or install the probe to a different installation directory.

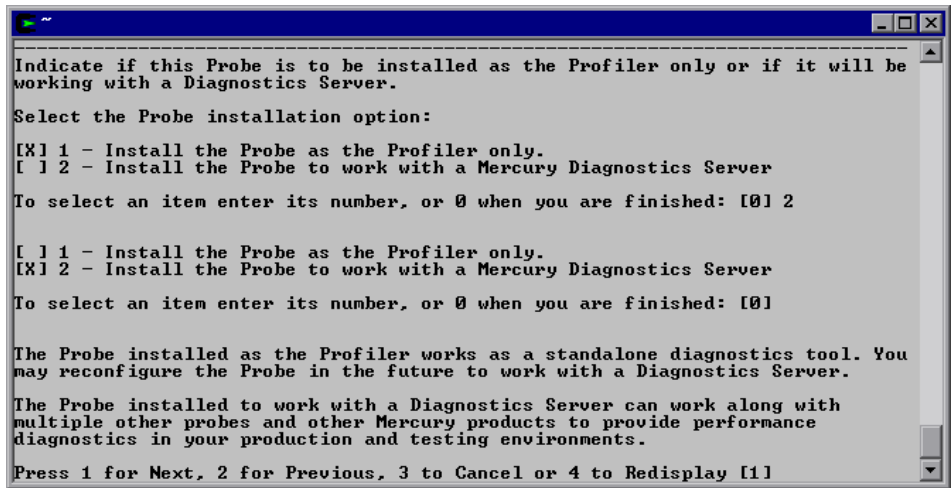
- ▶ To upgrade the existing probe installation, specify the installation directory of the existing probe at the **Directory Name** prompt. For more information about upgrading Diagnostics components, see Appendix F, “Upgrading Diagnostics and Other Mercury Products.”

When the installer runs, it backs up any property settings for the existing J2EE Probe installation to a directory called **etc.old**.

- ▶ To install the probe in an installation directory that is different from the existing probe’s directory, specify an installation directory other than that of the existing probe.

Select **Next** to continue with the installation.

4 Specify whether you want to install the probe as the Profiler only, or with a Mercury Diagnostics Server.



The probe can be configured to work in one of the following ways:

- To use the probe with other probes and other Mercury products to provide performance diagnostics in your production or testing environment, select **Install the Probe to work with a Mercury Diagnostics Server**.
- To use the probe as a stand-alone diagnostics tool for a development environment, select **Install the Probe as the Profiler only**.

Select **Next** to continue with the installation.

Note: When the probe is installed to work with a Mercury Diagnostics Server, it is configured so that the J2EE Diagnostics Profiler is available as well. You can drill down from the Mercury Diagnostics views into the tabs of the J2EE Diagnostics Profiler or you can access the Profiler in stand-alone mode.

A probe that was installed in Profiler-only mode can be reconfigured to work with the Diagnostics Server if this becomes necessary.

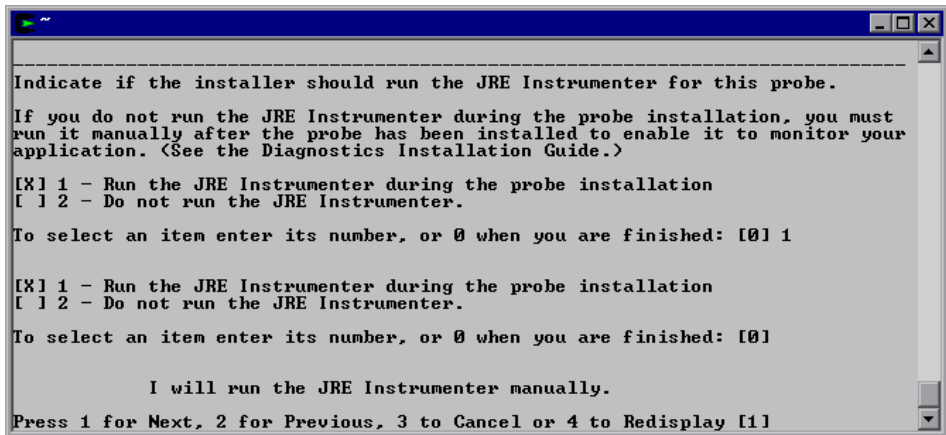
Next Step: At this stage, the installation procedure branches, depending on the option that you selected for installing the probe.

- ▶ If you are installing the probe as a Profiler only, continue with “Installing the Probe as a Profiler Only” on page 126.
 - ▶ If you are installing the probe to work with a Mercury Diagnostics Server, continue with “Installing the Probe to Work with a Mercury Diagnostics Server” on page 129.
-

Installing the Probe as a Profiler Only

If you are installing the probe to work as a Profiler only, continue with the following procedure.

- 1** Indicate whether or not you want to run the JRE Instrumenter during the installation of the probe.



- ▶ To run the instrumenter now, select **Run the JRE Instrumenter during the probe installation**.

- To skip this step, select **Do not run the JRE Instrumenter**. If you choose to skip this step now, you must run the JRE Instrumenter manually before you can use the probe. For instructions, see Chapter 8, “Running the JRE Instrumenter.”

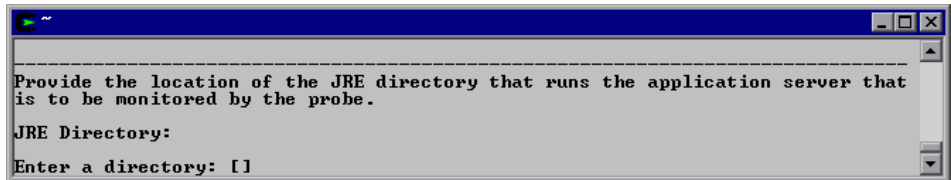
Note: If you are installing this probe to work with WebSphere in an Integrated Development Environment (IDE), skip the JRE Instrumenter. You will run the JRE Instrumenter manually. See “Running the JRE Instrumenter for WebSphere IDE” on page 183.

Select **Next** to continue with the installation.

Next Step: At this stage, the installation procedure branches, depending on the option that you selected for running the JRE Instrumenter.

- If the installer is running the JRE Instrumenter, the installation process continues as documented in the next step.
 - Otherwise, the installation process skips the next step, and proceeds with the step following the next.
-

2 Enter the JRE directory.



Type the path to the JRE bin folder where the JRE used by the monitored application server can be found. For example, if you have installed WebLogic 8.1 (using Sun JDK) in your **/opt/boa** directory, the JRE directory is **/opt/boa/jdk142_05/JRE**.

Select **Next** to continue with the installation.

3 Confirm the pre-installation summary information.

The installation settings that you selected are displayed. Review the information to make sure that you are satisfied.

To change your settings, select **Previous** to return to the previous prompts.

Select **Next** to start the installation.

4 Close the installation wizard.

When the installation has completed, review the post-installation summary information to make sure that the installation completed successfully.

Select **Finish** to exit the installation.

5 Configure the application server.

Before you can use the probe to monitor your application, you need to instrument the JRE and manually modify the application startup script.

- a** If you chose to skip running the JRE Instrumenter during this installation, you need to run it manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.
- b** Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”

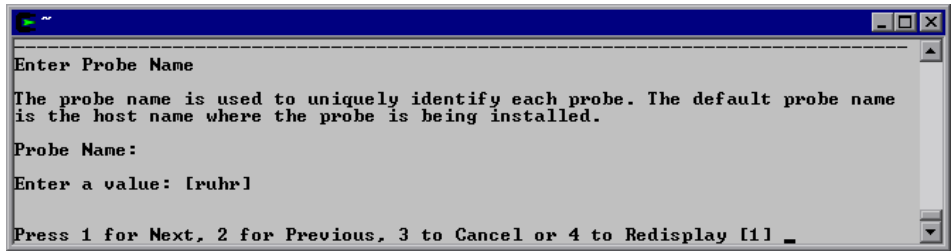
Note: The probe is automatically configured during the installation process. In certain situations however, you may be required to change these default settings or to manually configure the probe. For more information, see “Configuring the Probe Manually” on page 144.

For situations that require more advanced configurations of the probe and application server, see Chapter 21, “Advanced J2EE Probe and Application Server Configuration.”

Installing the Probe to Work with a Mercury Diagnostics Server

If you are installing the probe to work with a Mercury Diagnostics Server, continue with the following procedure.

1 Assign a name for the probe.



Enter a unique name that is to be used to identify the probe within Mercury Diagnostics. The following characters can be used in the name: - , _ , and all alphanumeric characters.

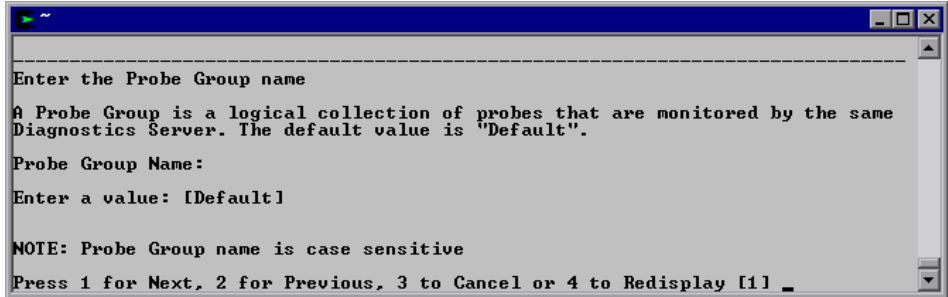
When assigning a name to a probe, choose a name that will help you recognize the application that the probe is monitoring, and the type of probe (in this case, a J2EE Probe).

For example, the probe name for the first J2EE Probe installed, that will be monitoring an application named PetWorld, could be:

```
PetWorldJ2EEProbe1
```

Select **Next** to continue with the installation.

2 Assign a name for the probe group.



Enter a name for the probe group for this probe.

Note: The probe group is case-sensitive.

Probe groups are logical groupings of probes that report to the same Diagnostics Server. The performance metrics for a probe group are tracked and can be displayed on many of the Diagnostics views.

For example, you may want to assign all of the probes for a particular enterprise application to a probe group so that you can monitor both the performance of the group and the performance of the individual probes.

Select **Next** to continue with the installation.

3 Enter the details for the Diagnostics Server in Mediator mode.

Enter the necessary information to enable communication with the Diagnostics Server in Mediator mode.

```

Provide the location of the Diagnostics Server in Mediator mode.
Diagnostics Server Mediator Host <Name or IP address>:
Enter a value: [lissaquah]

Diagnostics Server Mediator Port:
Enter a value: [2612]

The default value is 2612.
[X] 1 - Check the connectivity to the Diagnostics Server
To select an item enter its number, or 0 when you are finished: [0]

Mediator Host and Port.
Press 1 for Next, 2 for Previous, 3 to Cancel or 4 to Redisplay [1]

```

Note: If there is only one Diagnostics Server in the Diagnostics deployment where the probe will run, enter the Diagnostics Server's host name and event port information here.

When there is more than one Diagnostics Server in your deployment, enter the information for the Diagnostics Server in Mediator mode that is to receive the events from the probe.

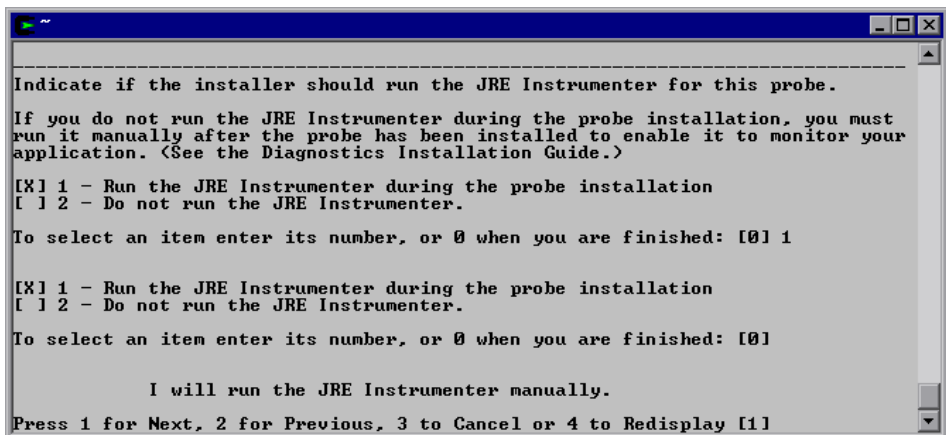
- a** Enter the host name or IP address of the host of the Diagnostics Server.
You should specify the fully qualified host name; not just the simple host name. In a mixed OS environment, where UNIX is one of the systems, this is essential for proper network routing.
- b** Enter the port number of the Diagnostics Server.
The default port for the Diagnostics Server is **2612**. If you have changed the port since the Diagnostics Server was installed, you should specify that port number here instead of the default.

- If you want to make sure that the specified Diagnostics Server is running and accessible from the installation host, select **Check the connectivity to the Diagnostics Server**.

Select **Next** to continue with the installation.

If you selected **Check the connectivity to the Diagnostics Server Mediator Host and Port** and connectivity problems were encountered, the installer provides you with the results of the connectivity check. If you do not want to address these problems at this stage, clear this option, proceed with the installation, and address the problem later.

4 Indicate if you want to run the JRE Instrumenter.



- To run the instrumenter now, select **Run the JRE Instrumenter during the probe installation**.
- To skip this step, select **Do not run the JRE Instrumenter**. If you choose to skip this step now, you must run the JRE Instrumenter manually before you can use the probe. For instructions, see Chapter 8, “Running the JRE Instrumenter.”

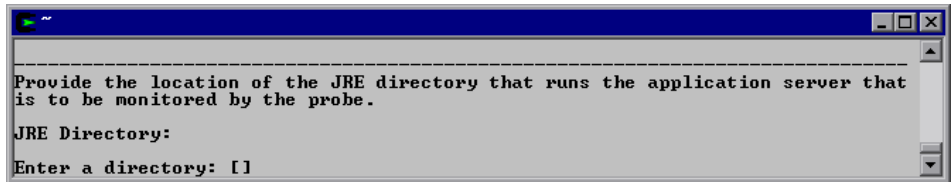
Note: If you are installing this probe to work with WebSphere in an Integrated Development Environment (IDE), skip the JRE Instrumenter. You will run the JRE Instrumenter manually. See “Running the JRE Instrumenter for WebSphere IDE” on page 183.

Select **Next** to continue with the installation.

Next Step: At this stage, the installation procedure branches, depending on the option that you selected for running the JRE Instrumenter.

- ▶ If the installer is running the JRE Instrumenter, the installation process continues as documented in the next step.
 - ▶ Otherwise, the installation process skips the next step, and proceeds with the step following the next.
-

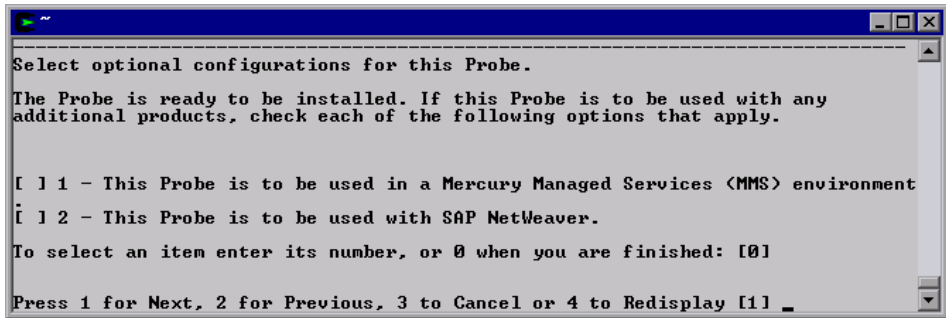
5 Enter the JRE directory.



Type the path to the JRE bin folder where the JRE used by the monitored application server can be found. For example, if you have installed WebLogic 8.1 (using Sun JDK) in your `/opt/BEA` directory, the JRE directory is `/opt/BEA/jdk142_05/JRE`.

Select **Next** to continue with the installation.

6 Select the configuration options if the probe is to be used with additional products.



Indicate whether you will be using the probe with other products.

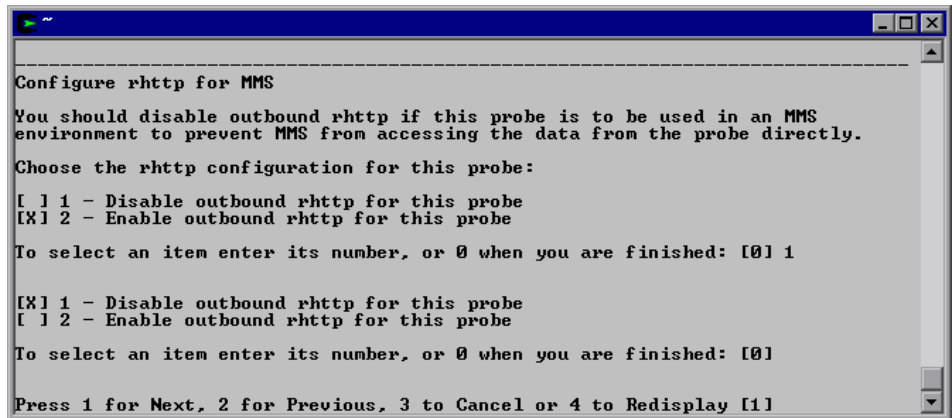
- ▶ To use the probe within a Mercury Managed Services (MMS) environment, select **This Probe is to be used in a Mercury Managed Services (MMS) environment.**
- ▶ To use the probe with a SAP NetWeaver Web application server, select **This Probe is to be used with SAP NetWeaver.**

Select **Next** to continue with the installation.

7 Configure the Mercury Managed Services environment.

If you indicated that the probe is to be used in a Mercury Managed Services environment (see step 6 above), it is recommended that you disable outbound rhttp (reverse http).

The Diagnostics Server uses rhttp to query metrics from the J2EE Probe. You should disable the probe's outgoing rhttp to prevent Mercury Managed Services from accessing the data directly from the probe.



- To disable outbound rhttp, select **Disable outbound rhttp for this probe**.
- To enable outbound rhttp, select **Enable outbound rhttp for this probe**.

Select **Next** to continue with the installation.

8 Confirm the pre-installation summary information.

The installation settings that you selected are displayed. Review the information to make sure that you are satisfied.

To change your settings, select **Previous** to return to the previous prompts.

To start the installation of the J2EE Probe, select **Next**. A progress bar and status message is displayed.

9 Close the installation wizard.

When the installation has completed, review the post-installation summary information to make sure that the installation completed successfully.

Select **Finish** to exit the installation.

10 Configure the application server.

Before you can use the probe to monitor your application, you need to instrument the JRE and manually modify the application startup script.

- a** If you chose to skip running the JRE Instrumenter during this installation, you need to run it manually before you can use the probe. For more information, see “Running the JRE Instrumenter” on page 155.
- b** Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”

Note: The probe is automatically configured during the installation process. In certain situations however, you may be required to change these default settings or to manually configure the probe. For more information, see “Configuring the Probe Manually” on page 144.

For situations that require more advanced configurations of the probe and application server, see Chapter 21, “Advanced J2EE Probe and Application Server Configuration.”

11 Verify the probe installation.

If you are installing this probe to work with a Mercury Diagnostics Server, you can verify the probe installation as described in “Verifying the J2EE Probe Installation” on page 148.

Installing the J2EE Probe on a z/OS Mainframe

This section provides instructions for installing the J2EE Probe from the .tar file that is included on the product CD.

Important Considerations when Installing the J2EE Probe on z/OS

Consider the following before you install a probe in a z/OS environment:

Editing Property Files on a z/OS Mainframe

When installed in a z/OS environment, the probe expects the Diagnostics property files to be in EBCDIC format. Use an EBCDIC editor to update the property files and make sure to store the updates in the same format.

Viewing the System Logs in z/OS

You can view the system log by accessing the primary operator's console in SDSF.

Capturing Metrics on the z/OS Mainframe

Load test metrics are not captured for z/OS. The probe can be configured to capture a limited number of system level metrics.

For more information on capturing system metrics in z/OS, see “Enabling z/OS System Metrics Capture” on page 468.

Installing the J2EE Probe on z/OS from the Product CD

A tar file containing the probe files has been included on the product CD.

To install the J2EE Probe on a z/OS mainframe:

- 1 Upload **ProbelInstall_zOS.tar.gz** from the **Diagnostics_Probes** folder on the Mercury Diagnostics installation CD (or directly from the probe installation cd) to the directory on the z/OS machine where you want to unzip the installer.
- 2 Unzip **ProbelInstall_zOS.tar.gz** using `gzip` as shown in the following example:

```
gzip -d ProbelInstall_zOS.tar.gz
```

This command creates the unzipped file, **ProbelInstall_zOS.tar**.

- 3 To unpack the tar file, run the `tar` command as shown in the following example:

```
tar -xvf ProbelInstall_zOS.tar
```

This command creates the unpacked directory, **ProbelInstall**.

- 4 Prepare to run **jreinstrumenter.sh** by setting the permissions on the script so that it can be run (`o+x`).
- 5 Run the JRE Instrumenter. For z/OS, use the following command:

```
<path_to_java> -jar <path_to_jre_instrumenter> -b <jvm_directory>
```

where:

- `<path_to_java>` is the path to the java executable.
 - `<path_to_jre_instrumenter>` is `<probe_install_dir>/lib/jreinstrumenter.jar`
 - `<jvm_directory>` is the path to the JRE for the java installation that your application is using.
- 6 Configure the probe properties to enable the probe to work with the other Diagnostics components as described in “Configuring the Probe Manually” on page 144.

- 7 Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”
- 8 Verify the probe installation as described in “Verifying the J2EE Probe Installation” on page 148.
- 9 If you want to collect z/OS system metrics, see “Enabling z/OS System Metrics Capture” on page 468.

Installing J2EE Probes on Multiple z/OS Machines

If you will be installing probes on more than one z/OS machine, you may want to create a pax archive of the probe implementation on the first machine and then use the pax archive to install the probe onto the other machines. Contact your system administrator for more information.

Installing the J2EE Probe Using the Generic Installer

Important! This section only applies if you are going to install the probe to run with a Diagnostics Server.

The installers for the J2EE Probe have been built to support installing the probe on all of the platforms for which the component has been certified. However, the probe may work on other platforms that have not yet been certified. A generic installer has been provided on the product CD to allow you to install the probe on these uncertified platforms.

To get the probe to work on the platforms that are not supported by the regular installer, you must run the generic installer, and manually configure the probe so that it will be able to communicate with the other Diagnostics components and be able to monitor the processing of your application.

To install and configure the J2EE Probe on an uncertified platform:

- 1** Locate **ProbelInstall.tar.gz** from the **Diagnostics_Probes** folder on the Mercury Diagnostics installation CD (or directly from the probe installation cd).
- 2** Unzip **ProbelInstall.tar.gz** using `gzip` as shown in the following example:

```
gzip -d ProbelInstall.tar.gz
```

When this command completes, the unzipped file is called **ProbelInstall.tar**.

- 3** To unpack the tar file, execute the tar command as shown in the following example:

```
tar -xvf ProbelInstall.tar
```

This command creates the unpacked **ProbelInstall** directory.

- 4** Configure the probe to enable the probe to work the other Diagnostics components as described in “Configuring the Probe Manually” on page 144.
- 5** Configure the Application Server to allow the probe to monitor your application.
 - a** Run the JRE Instrumenter as described in “Running the JRE Instrumenter” on page 155.
 - b** Once you have installed the probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the probe is started together with the monitored application. For detailed instructions, see Chapter 9, “Configuring the Application Servers.”
- 6** Verify the probe installation as described in “Verifying the J2EE Probe Installation” on page 148.

Silent Installation of the J2EE Probe

A *silent installation* is an installation that is performed automatically, without the need for user interaction. In place of user input, the silent installation accepts input from a response file for each step of the installation.

For example, a system administrator who needs to deploy a component on multiple machines can create a response file that contains all the prerequisite configuration information, and then perform a silent installation on multiple machines. This eliminates the need to provide any manual input during the installation procedure.

Before you perform silent installations on multiple machines, you need to generate a response file that will provide input during the installation procedure. This response file can be used in all silent installations that require the same input during installation.

To generate a response file:

- Perform a regular installation with the following command-line option:

```
<installer> -options-record <responseFileName>
```

This creates a response file that includes all the information submitted during the installation.

To perform a silent installation:

- Perform a silent installation using the relevant response file.

You perform the silent installation with the **-silent** command-line option as follows:

```
<installer> -silent -options <responseFileName>
```


7

Post-Installation Configuration of the Mercury Diagnostics Probe for J2EE

This chapter provides instructions for configuring and working with the performing the Mercury Diagnostics Probe for J2EE (J2EE Probe) after it has been installed. You should review these instructions to make sure that you have configured the Probe to effectively and efficiently monitor your application's performance.

This chapter describes:	On page:
Configuring the Probe Manually	144
Configuring the Application Server	147
Configuring the Probe for Advanced Situations	148
Deploying BEA JRockit Mission Control	148
Verifying the J2EE Probe Installation	148
Determining the Version of the J2EE Probe	150
Upgrading to a Newer Version of the J2EE Probe	151
Uninstalling the J2EE Probe	154

Configuring the Probe Manually

The probe is automatically configured during the installation process. The default configuration of the J2EE Probe has been set to monitor your application effectively, with very little impact on the performance of your application. In the following situations however, you may be required to change these default settings or to configure the probe manually.

- ▶ The Windows and UNIX installers gather the information needed to automatically configure the probe. If you skipped some of the optional steps in the installer, or if you need to change some of the information that you provided to the installer after the probe has been installed, you may need to perform some or all of the probe configuration steps manually.
- ▶ After you have installed the probe using the generic installation process or the z/OS installation process, you must configure the probe properties so that the probe can work with the other Diagnostics Components.

Note: For information about more advanced Probe configuration instructions, see Chapter 21, “Advanced J2EE Probe and Application Server Configuration.”

To configure the J2EE Probe manually:

1 Configure the probe properties:

a In `<probe_install_dir>/etc/probe.properties`:

- Set the **active.products** property to indicate the products that the probe will be working with. See the comments in the property file for the valid values.
- Set the **id** property to indicate the probe id for the probe. Type the unique name that is to be used to identify the probe within Mercury Diagnostics. The following characters can be used in the name: - , _ , and all alphanumeric characters.
- Set the **group** property to assign the probe to a probe group. Make sure that the probe group is exactly the same as that of the other probes with which you want the probe grouped.

b In `<probe_install_dir>/etc/dynamic.properties`:

- Set the **mediator.host.name** property to indicate the host name for the Diagnostics Server in Mediator mode.
- Set the **mediator.port.number** property to indicate the port number where the Diagnostics Server Mediator listens for probe communication. The default port number is 2612. If you have changed the port since the Diagnostics Server was installed, you should specify that new port number here instead of the default.

c In `<probe_install_dir>/etc/dispatcher.properties`:

Set the **registrar.url** property value to the URL for the Diagnostics Server in Mediator mode to indicate the host name and port for the Diagnostics Server in Mediator mode.

```
registrar.url=http://<mediator>:2006/commander/registrar/
```

Note: Make sure that you update the **registrar.url** property in the property file that is appropriate for the configuration of your network. The property **registrar.url** exists in two different property files in `<probe_install_dir>\etc`.

The **registrar.url** property in **dispatcher.properties** is used to provide the URL for the registrar when there is not a proxy.

The **registrar.url** property in **webserver.properties** is used to provide the URL for the registrar when there is a proxy server between the probe and the registrar.

- 2** Ensure that the permission bits on the `<probe_install_dir>/log` directory are set so that the user who starts the application server has write (o+w) permissions.

Important: Before you can use the J2EE Probe to monitor your application, you need to instrument the JRE and manually modify the application startup script. For more information, see “Configuring the Application Server” on page 147.

Configuring the Application Server

In order to allow the Probe to monitor your application, you need to instrument the JRE and manually modify the application server startup script.

Instrumenting the JRE

The instrumentation process defines for the Probe, the classes and methods that are to be monitored in your application. It also controls the layers that will be used to report the performance metrics that were gathered.

You use the JRE Instrumenter to run the instrumentation. The JRE Instrumenter automatically runs at the end of the Probe installation to prepare your application for instrumentation unless you indicate otherwise.

If you elect to skip running the JRE Instrumenter during the Probe installation, you must run it manually before you can use the Probe to monitor your application. For more information, see Chapter 8, “Running the JRE Instrumenter.”

Modifying the Application Startup Script

Once you have installed the Probe and instrumented the JRE, you need to manually modify the startup script for your application server so that the Probe is started together with the monitored application.

For detailed instructions about how to modify your application server's startup script, see Chapter 9, “Configuring the Application Servers.”

Note: The process for configuring the J2EE Probe and your application servers when there are multiple JVMs on a single machine is described in “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.

Configuring the Probe for Advanced Situations

You should review the topics in Chapter 21, “Advanced J2EE Probe and Application Server Configuration” to determine if there are Probe configuration settings that will enable the Probe to work more efficiently in your environment.

Deploying BEA JRockit Mission Control

The Mercury Diagnostics license allows you to use BEA JRockit Mission Control. If the application that you are monitoring is using BEA JRockit JVM and if the J2EE Probe has been configured to work with a Diagnostics Server, then you will be able to access a license for BEA JRockit Mission Control and use Mission Control to monitor your application’s performance.

For more information see “Using BEA JRockit Mission Control” on page 414 for instructions on how to access and deploy the license for BEA JRockit Mission Control.

Verifying the J2EE Probe Installation

You can verify the Probe installation using the System Health Monitor.

For detailed instructions about how to use the System Health Monitor, see the *Mercury Diagnostics User’s Guide*.

Important: This section only applies if you installed the Probe to run with a Diagnostics Server.

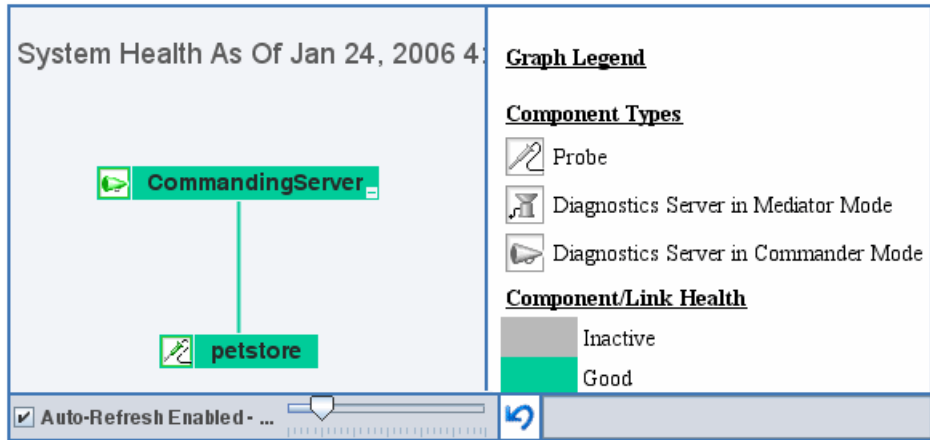
Note: The Probe does not register with the Diagnostics Server until it is started. The Probe is started when the instrumented application server is started. Therefore, you cannot verify the installation of the Probe using the System Health Monitor until you have configured the Probe and Application Server as described in this chapter.

If you have been following the recommended installation sequence, after you have installed the J2EE Probe you are able to verify the following:

- ▶ The Diagnostics Server in Commander mode was successfully installed.
 - ▶ Where relevant, additional Diagnostics Servers in Mediator mode were successfully installed and are communicating with the Diagnostics Server in Commander mode.
 - ▶ The J2EE Probe was successfully installed and has established connectivity with the Diagnostics Server.
-

Note: The J2EE Probe is not displayed in System Health when first installed because the Probe does not register with the Diagnostics Server until it is started. The Probe is started and is registered with the Diagnostics Server when you start the monitored application server.

Depending on your deployment, the new J2EE Probe is shown on the System Health Monitor as a child of either the Diagnostics Server in Commander mode or the Diagnostics Server in Mediator mode.



The J2EE Probe is colored grey in the System Health Monitor when the monitored application is closed. Once the application has been started the Probe is colored green.

Determining the Version of the J2EE Probe

When you request support, it is useful to know the version of the Diagnostics component that you have a question about.

You can find the version of the Probe in one of the following ways:

- ▶ Locate the version file `<Probe_install_dir>\version.txt`. The file contains the 4-digit version number, as well as the build number.
- ▶ The version number is available in the Probe log file (`<Probe_install_dir>/log/<probe_id>/probe.log`)
- ▶ The version number is available among the details of the Probe on System Health Monitor.

Upgrading to a Newer Version of the J2EE Probe

When you install the Probe, the installer detects if a Probe has already been installed in the installation directory that you specified. The installer upgrades the Probe installation to the current version and creates a backup of the property settings in the `<probe_install_dir>/etc` directory. The backup file is called `<probe_install_dir>/etc.old`.

Note: If you do not want the installer to upgrade the installation for an existing probe, you should specify a different installation directory when prompted by the installer.

The following instructions are instructions for upgrading from an older version of the Mercury Diagnostics Probe for J2EE

Note: The current version of the J2EE Probe has been designed to work with the current versions of the Diagnostics components that are part of the Mercury LoadRunner 8.1, Mercury Performance Center 8.1, and Mercury Business Availability Center 6.1 or later installations. The Probe may not work with earlier versions of these products. Likewise, previous versions of the Probe may not work with the current versions of these Mercury products.

To upgrade the J2EE Probe to a newer version of the Probe:

- 1** Shutdown the application servers that are being monitored by the Probes that you want to upgrade.
- 2** Create a backup copy of the startup script for the application server.

- 3 If you want to reuse the configuration of your current Probe with the new Probe, create a backup copy of the folder **<Probe_install_dir>/etc** called **old_etc_backup**.

Note: This optional step could help you save time by reusing the current settings for Probe configuration such as instrumentation changes, buffering changes, port numbers, Diagnostics Server locations, probe.id, and product mode.

- 4 Uninstall the Probe. For instructions, see “Uninstalling the J2EE Probe” on page 154.
- 5 Install the new version of the J2EE Probe as instructed in Chapter 6, “Installing the Mercury Diagnostics Probe for J2EE.”

Note: When you install the J2EE Probe, do not select the option to have the installer configure your application server, and do not select the option to allow the installer to instrument the JRE. When the Probe installation is complete, proceed with the steps below.

- 6 If you made a backup copy of the old **<Probe_install_dir>/etc** folder so that you could reuse the old Probe’s configuration in step 3 above, you should follow the backup information to configure the new Probe.
 - a Create a backup of the newly installed folder **<Probe_install_dir>/etc** called **new_etc_backup** so that you have a copy of the files as they were created by the installer.
 - b Rename the **<Probe_install_dir>/etc/auto_detect.points** file that was just installed to **new_auto_detect.points**.
 - c Copy the backup copy of **auto_detect.points** into **<Probe_install_dir>/etc**.

- d** The old **auto_detect.points** file will work with the new Probe. If you want to take advantage of the enhancements in the new version of the Probe, copy the following sections from **new_auto_detect.points** file:
- All sections that begin “BEA-”, including the preceding comments
 - All sections that begin “SAP_”, including the preceding comments
 - The “Synchronization” section, including the preceding comments
 - The “RMI” section, including the preceding comments
 - args_by_class
 - Any enhancements/optimizations that you made
- e** Copy the rest of the files, excluding **modules.properties** and **webserver.properties**, from **old_etc_backup**. Replace the files in the current **<Probe_install_dir>/etc/**.

Note: Do not replace **modules.properties** and **webserver.properties**. You must use the new version of these files.

- f** In **capture.properties**, add the following property:

```
ac.register.sink=true
```

- g** In **dispatcher.properties**, add the following properties:

```
dispatcher.properties.file.name=dispatcher.properties
registrar.server_ping.time=90s
ac.autostart=true
```

- h** In **inst.properties**, revise the value of the following property:

```
classes.to.exclude=!aik\.security\.*;!c8e\.*;!org\.jboss\.net\.protocol\.file\
Handler;!org\.jboss\.net\.protocol\.file\.FileURLConnection;!.*ByCGLIB.*
```

- i** In **inst.properties**, add the following property:

```
rmi=com.mercury.opal.capture.inst.RMIServerInstrumenter,com.mercury.o
pal.capture.inst.RMIProxyInstrumenter
```

- 7 Configure the J2EE Probe and application server following the instructions in this chapter.

Note: If a firewall separates your Probe and the Diagnostics Servers from the other Diagnostics components, make sure that you have configured the firewall to allow communications across the firewall using the port numbers in the range that you specify. For more information, see “Configuring Diagnostics Components to Work with a Firewall” on page 435.

To avoid having to reconfigure your firewall, you may want to override the default port numbers for the new Probe so that they will match the port numbers that the older Probe used.

Uninstalling the J2EE Probe

To uninstall the J2EE probe:

- ▶ On a Windows machine, run **uninstaller.exe** which is located in the `<Probe_install_dir>_uninst` directory.
- ▶ On a UNIX machine, run **uninstall*** which is located in the `<Probe_install_dir>_uninst` directory.

8

Running the JRE Instrumenter

This chapter explains how to run the JRE Instrumenter to enable the Mercury Diagnostics Probe for J2EE to monitor your application.

This chapter describes:	On page:
About the JRE Instrumenter	155
Running the JRE Instrumenter	157

About the JRE Instrumenter

When you install a J2EE Probe, you must run the JRE Instrumenter to allow the Probe to gather the performance metrics for the application that it is to monitor. The JRE Instrumenter instruments the ClassLoader class for the JVM that your application is using and places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes directory. It also provides you with the JVM parameter that must be used when the application server is started so that your application server will use the instrumented class loader.

The JRE Instrumenter runs automatically during the Probe installation unless you elect not to run it at that time. If you elect to skip the JRE Instrumenter during the installation of the Probe, you must run it manually as described in this chapter.

You must run the JRE Instrumenter again if the JDK (java.exe executable) used by your application server changes so that the Probe will be able to continue to monitor the processing.

Note: If a Probe is being used to monitor multiple JVMs, the JRE Instrumenter must be run once for each JVM so that the Probe can be prepared to instrument the applications that are running on each JVM. For details, see “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.

JRE Instrumenter Processing

The JRE Instrumenter performs the following functions:

- ▶ Identifies JVMs that are available to be instrumented.
- ▶ Searches for additional JVMs in directories that you specify.
- ▶ Instruments the JVMs that you specify and provides the parameter that you must add to the startup script for the JVM to point to the location of the instrumented ClassLoader class.

The Instrumenter puts the instrumented ClassLoader in two different places depending on how it is executed.

- ▶ When the Instrumenter is run from the Probe installer, the Instrumenter places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes/boot directory.
- ▶ When the Instrumenter is run using the graphical interface in a Windows or UNIX environment, the Instrumenter places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes/<JVM_vendor>/<JVM_version> directory.
- ▶ When the Instrumenter is run in a UNIX environment in console mode, the Instrumenter places the instrumented ClassLoader in either a folder under the <probe_install_dir> /classes/boot directory or the <probe_install_dir> /classes/<JVM_vendor>/<JVM_version> directory depending on the processing option specified. For more information on the UNIX processing options see “Instrumenting a Listed JVM” on page 165.

Running the JRE Instrumenter

Instructions for running the JRE Instrumenter in a Windows environment and in a UNIX environment in console mode are provided below.

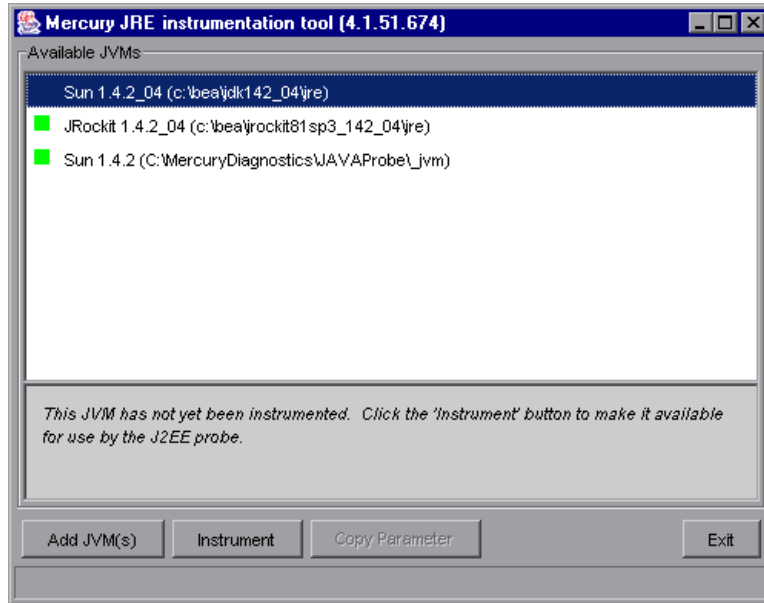
Note: If you chose to not run the JRE Instrumenter during the Probe installation, you must run it manually to enable the Probe to gather the performance metrics for your application.

Running the JRE Instrumenter on a Windows Machine

When the JRE Instrumenter is run in a Windows environment, the Instrumenter displays the dialogs of its graphical user interface. The same dialogs are displayed when running the installer on a UNIX machine when the Instrumenter is not running in console mode.

Starting the JRE Instrumenter on a Windows Machine

Open `<probe_install_dir>\bin` to locate the JRE Instrumenter executable. Run the command `jreinstrumenter.cmd`. When the Instrumenter starts, it displays the Mercury JRE Instrumentation Tool dialog.



The Instrumenter lists the JVMs that were discovered by the Instrumenter and are available for instrumentation. The JVMs that have already been instrumented are listed with a green square preceding the name of the JVM.

From the Mercury JRE Instrumentation Tool dialog you can perform the following tasks:

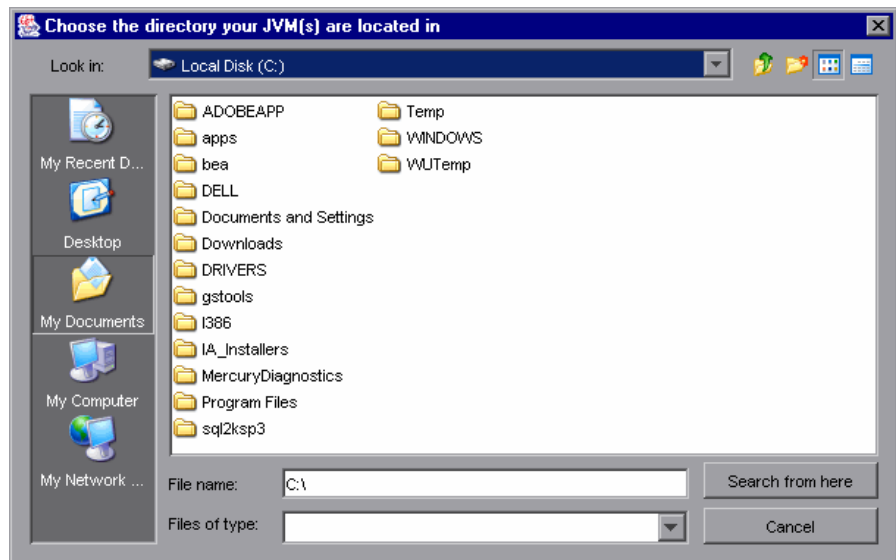
- ▶ If the JVM that you want to instrument is not listed in the Available JVMs list in the dialog, you can add JVMs to the list as described in “Adding JVMs to the Available JVMs List” on page 159.
- ▶ If the JVM that you want to instrument is listed but has not yet been instrumented, you can instrument the JVM as described in “Instrumenting a Selected JVM” on page 162.

- ▶ If the JVM that you want to instrument is listed and has already be instrumented, you can copy the JVM parameter that must be inserted into the start script for the JVM to activate the Probe's monitoring as described in "Including the JVM Parameter in the Application Server's Startup Script" on page 162.
- ▶ If you have finished using the JRE Instrumenter you can click **Exit** to close the Mercury JRE Instrumentation Tool.

Adding JVMs to the Available JVMs List

- 1 In the Mercury JRE Instrumentation Tool dialog, click **Add JVM(s)** to search for other JVMs and add them to the Available JVMs list.

The Instrumenter displays the Choose the Directory dialog box.

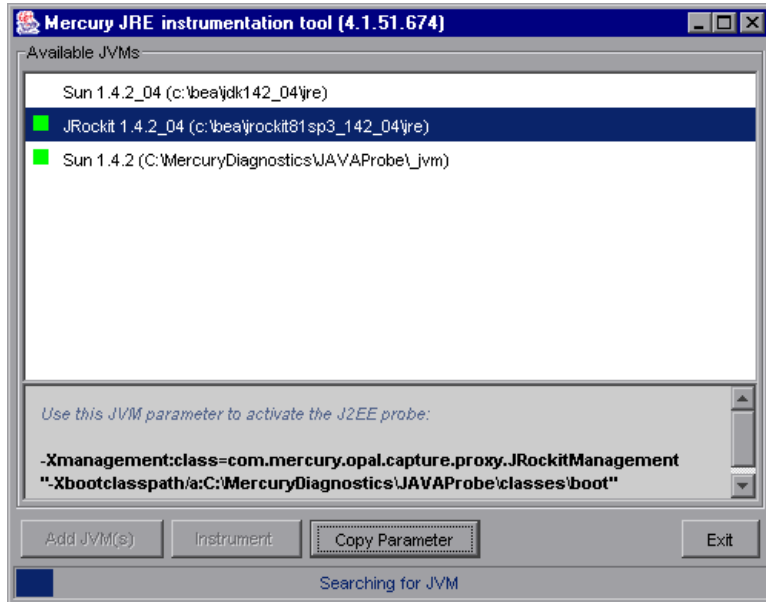


- 2 Navigate to the directory location where you would like the Instrumenter to begin searching for JVMs using the standard Windows Explorer type navigation controls.
- 3 Select the file where you would like to begin the search so that its name appears in the **File Name** box.

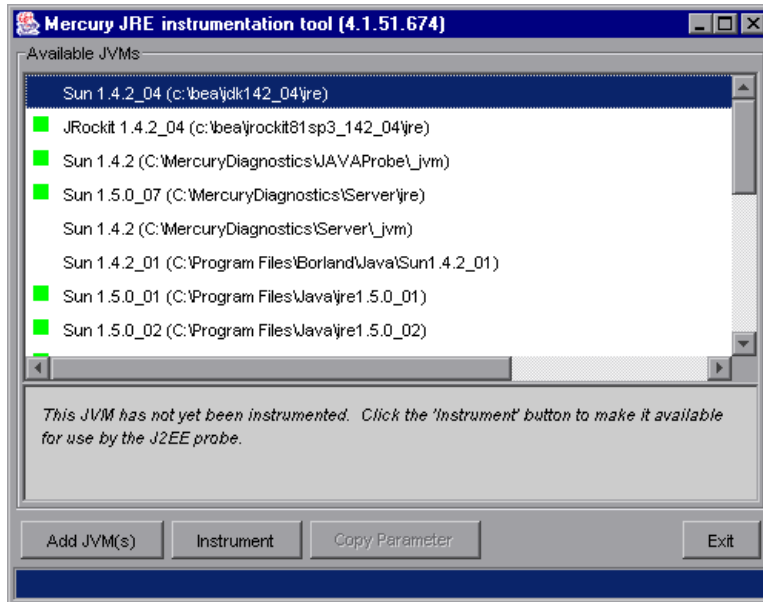
4 Click **Search from here** to start searching for JVMs.

The Instrumenter closes the dialog box and displays the Mercury JRE Instrumentation Tool dialog box once more. The command buttons on the dialog are disabled while the Instrumenter searches for JVMs. A progress bar at the bottom of the dialog indicates that the Instrumenter is searching and shows how far along it is in the search process.

As the tool locates JVMs, it lists them in the **Available JVMs** list.

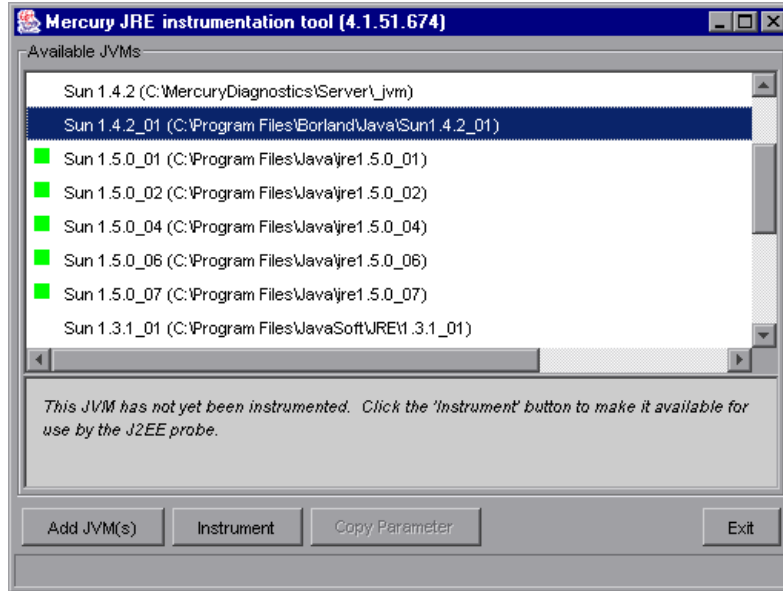


When the Instrumenter has completed the search, it enables the command buttons on the dialog has shown below. If the selected row is a JVM that has already been instrumented, the Instrument button is disabled.



Instrumenting a Selected JVM

Select a JVM that has not been instrumented from the **Available JVMs** list and click **Instrument**.



The JRE Instrumenter instruments the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes directory. It also displays the JVM parameter that must be used when the application server is started in the box below the Available JVMs list.

Including the JVM Parameter in the Application Server's Startup Script

When the JRE Instrumenter instruments a JVM it also creates the JVM parameter that you must include in the startup script for the application server in order to cause your application to use the instrumented class loader. When you select an instrumented JVM from the Available JVMs list the JVM parameter is displayed in the box below the list.

To copy the JVM parameter displayed in this box to the clipboard, click **Copy Parameter**. The JVM parameter is copied to the clipboard so that you can paste it into the location that allows it to be picked up when your application server starts.

Running the JRE Instrumenter on a UNIX Machine

The following instructions provide you with the steps necessary to run the JRE Instrumenter using either a graphical mode installation or a console mode installation.

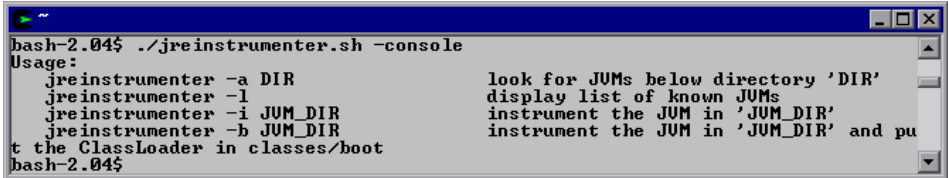
The JRE Instrumenter screens that are displayed in a graphical mode are the same as those documented for Windows installer in “Running the JRE Instrumenter on a Windows Machine” on page 157.

Starting the JRE Instrumenter on a UNIX Machine

Open `<probe_install_dir>\bin` to locate the JRE Instrumenter executable. Run the following command:

```
./jreinstrumenter.sh -console
```

When the Instrumenter starts, it displays a list of the processing options that are available:



```
bash-2.04$ ./jreinstrumenter.sh -console
Usage:
  jreinstrumenter -a DIR          look for JUMs below directory 'DIR'
  jreinstrumenter -l             display list of known JUMs
  jreinstrumenter -i JUM_DIR     instrument the JUM in 'JUM_DIR'
  jreinstrumenter -b JUM_DIR     instrument the JUM in 'JUM_DIR' and pu
t the ClassLoader in classes/boot
bash-2.04$
```

You can redisplay the list of options by specifying the `-x` option when you run the `jreinstrumenter.sh` command:

```
./jreinstrumenter.sh -x
```

The following table directs you to the documentation for each of the processing options:

Instrumenter Function	Documentation Section
<code>jinstrumenter -l</code>	“Displaying the List of Available JVMs” on page 164
<code>jinstrumenter -a DIR</code>	“Adding JVMs to the Available JVMs List” on page 164
<code>jinstrumenter -i JVM_DIR</code> <code>jinstrumenter -b JVM_DIR</code>	“Instrumenting a Listed JVM” on page 165

Displaying the List of Available JVMs

To display a list of the JVMs that are known to the JRE Instrumenter enter the following command:

```
./jinstrumenter -l
```

The Instrumenter lists the JVMs that it is aware of in rows containing the JVM vendor, JVM version, and the location where the JVM is located.

Adding JVMs to the Available JVMs List

To search for JVMs within a specific directory and add any JVMs that are found to the list of the JVMs that are known to the JRE Instrumenter enter the following command:

```
./jinstrumenter -a DIR
```

Replace DIR with the path to the location where you would like the Instrumenter to begin searching.

The Instrumenter searches the directories from the location specified including the directories and subdirectories. When it has completed its search, it displays the updated list of Available JVMs.

Instrumenting a Listed JVM

To instrument a JVM listed in the Available JVMs list, use one of the following two commands:

- Explicit path to ClassLoader

```
./jreinstrumenter -i JVM_DIR
```

Replace JVM_DIR with the path to the location of the JVM as specified in the Available JVM list.

This command instructs the JRE Instrumenter to instrument the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes/<JVM_vendor>/<JVM_version> directory.

This is the command that you should use; especially if you want to instrument multiple JVM to be monitored by a single Probe.

- Generic path to ClassLoader

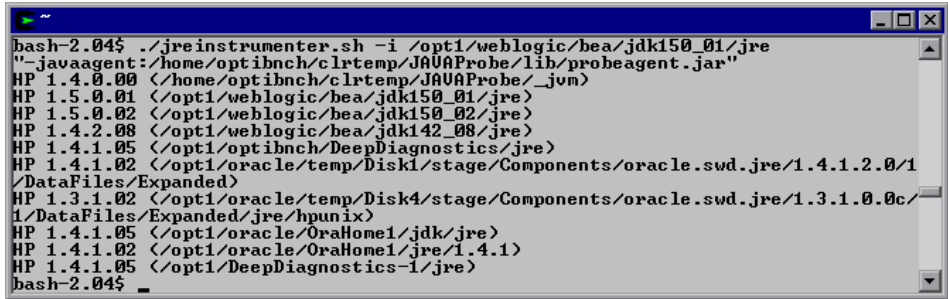
```
./jreinstrumenter -b JVM_DIR
```

Replace JVM_DIR with the path to the location of the JVM as specified in the Available JVM list.

This command instructs the JRE Instrumenter to instrument the ClassLoader class for the selected JVM and places the instrumented ClassLoader in a folder under the <probe_install_dir> /classes/boot directory.

You should only use this command if you are monitoring a single JVM with the Probe and there is some reason that you do not want to use the more explicit path generated when you use the -i command option.

When the Instrumenter has finished instrumenting the JVM, it displays the JVM parameter that must be used to activate the instrumentation and enable the Probe to monitor the performance of your application. Following the JVM parameter, the Instrumenter lists the Available JVM list again as shown in the following example:



```
bash-2.04$ ./jreinstrumenter.sh -i /opt1/weblogic/bean/jdk150_01/jre
"-javaagent:/home/optibnch/clrtemp/JAUAProbe/lib/probeagent.jar"
HP 1.4.0.00 </home/optibnch/clrtemp/JAUAProbe/_jvm>
HP 1.5.0.01 </opt1/weblogic/bean/jdk150_01/jre>
HP 1.5.0.02 </opt1/weblogic/bean/jdk150_02/jre>
HP 1.4.2.08 </opt1/weblogic/bean/jdk142_08/jre>
HP 1.4.1.05 </opt1/optibnch/DeepDiagnostics/jre>
HP 1.4.1.02 </opt1/oracle/temp/Disk1/stage/Components/oracle.swd.jre/1.4.1.2.0/1
/DataFiles/Expanded>
HP 1.3.1.02 </opt1/oracle/temp/Disk4/stage/Components/oracle.swd.jre/1.3.1.0.0c/
1/DataFiles/Expanded/jre/hpunix>
HP 1.4.1.05 </opt1/oracle/OraHome1/jdk/jre>
HP 1.4.1.02 </opt1/oracle/OraHome1/jre/1.4.1>
HP 1.4.1.05 </opt1/DeepDiagnostics-1/jre>
bash-2.04$ _
```

Including the JVM Parameter in the Application Server's Startup Script

When the JRE Instrumenter instruments a JVM it also creates the JVM parameter that you must include in the startup script for the application server in order to cause your application to use the instrumented class loader. When the Instrumenter has finished instrumenting the JVM, it displays the JVM parameter.

Copy the JVM parameter to the clipboard and paste it into the location that allows it to be picked up when your application server starts.

9

Configuring the Application Servers

The chapter provides instructions on how to configure your application server to allow the probe to monitor your application.

This chapter describes:	On page:
About Configuring the Application Server	168
Configuring WebSphere Application Servers	169
Configuring WebLogic Application Servers	184
Configuring the Oracle9i Application Server	193
Configuring the JBoss Application Server	196
Configuring the SAP NetWeaver Application Server	199
Configuring a Generic Application Server	201
Adjusting the Heap Size for the J2EE Probe in the Application Startup Script	203

About Configuring the Application Server

Once you have executed the JRE Instrumenter for the J2EE Probe, you must modify the startup script for your application so that the probe that is to monitor the application will be started when the application is started.

You can configure the application servers by updating the application server startup scripts manually. The following sections provide instructions for updating the certified application servers manually.

It is possible that your site administrator has site-specific methods for making configuration modifications. In this case, the generic procedure described in “Configuring a Generic Application Server” on page 201 should provide the information that the site administrator needs to implement the required configuration changes.

Note: If there are no instructions for your specific type of application server in the sections that follow, follow the procedure in “Configuring a Generic Application Server” on page 201.

The process for configuring the J2EE Probe and your application servers when there are multiple JVMs on a single machine is described in “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.

Note: In this chapter “<probe_install_dir>” is used to indicate the directory where the J2EE Probe was installed.

When modifying the -Xbootclasspath parameter, be sure to use quotes if there are spaces in the path that you specify.

Configuring WebSphere Application Servers

Supported versions	Operating systems
3.5, 4.x, 5.x, 6.x	AIX 4.3 Windows 2000/2003 Solaris 8

WebSphere servers are controlled using the WebSphere Application Server Administrative Console. The Console has control over the JVM command line and allows you to add classpath elements, define runtime variables (-D variables), and configure the bootclasspath for WebSphere 3.5, 4.x, 5.x, and 6.x versions. You use the Administrative Console to add the Xbootclasspath property and any additional arguments that are needed for the JVM command line.

Note that the appearance of the Console may differ for different versions of WebSphere. The instructions that follow show different screens for each of the certified WebSphere Application Server versions. The way that changes are implemented in each version of WebSphere may vary slightly. As a result, the examples shown in this section may not correspond exactly to your WebSphere version. The examples should provide the information that you need to enter the required parameters in the appropriate location in the Console.

Note: WebSphere applies the changes that are made on each tab only when you click **Apply** on the tab. Your changes will not be applied until you click **Apply**.

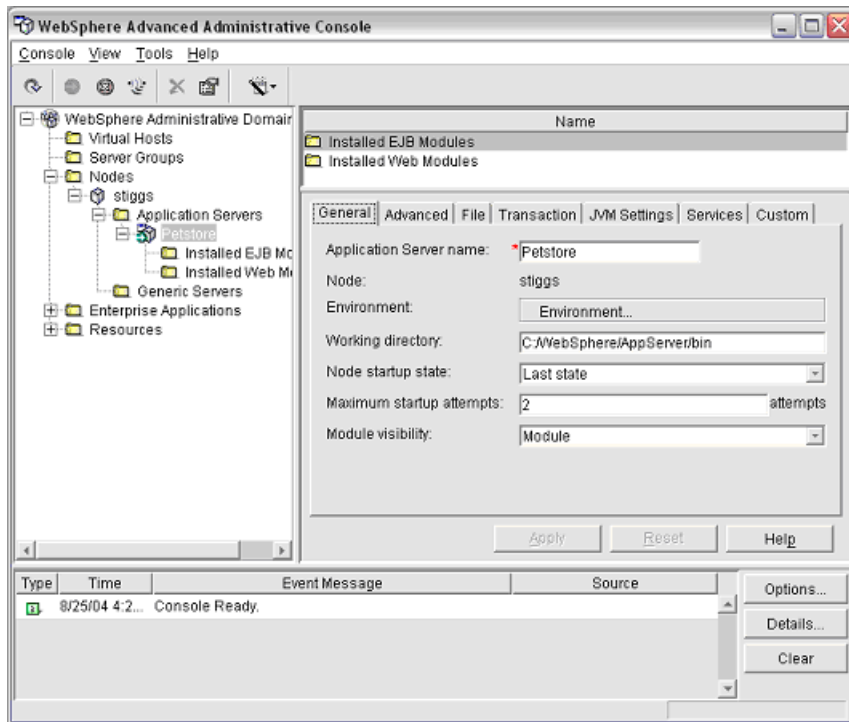
WebSphere 4.0

To configure a WebSphere 4.0 application server:

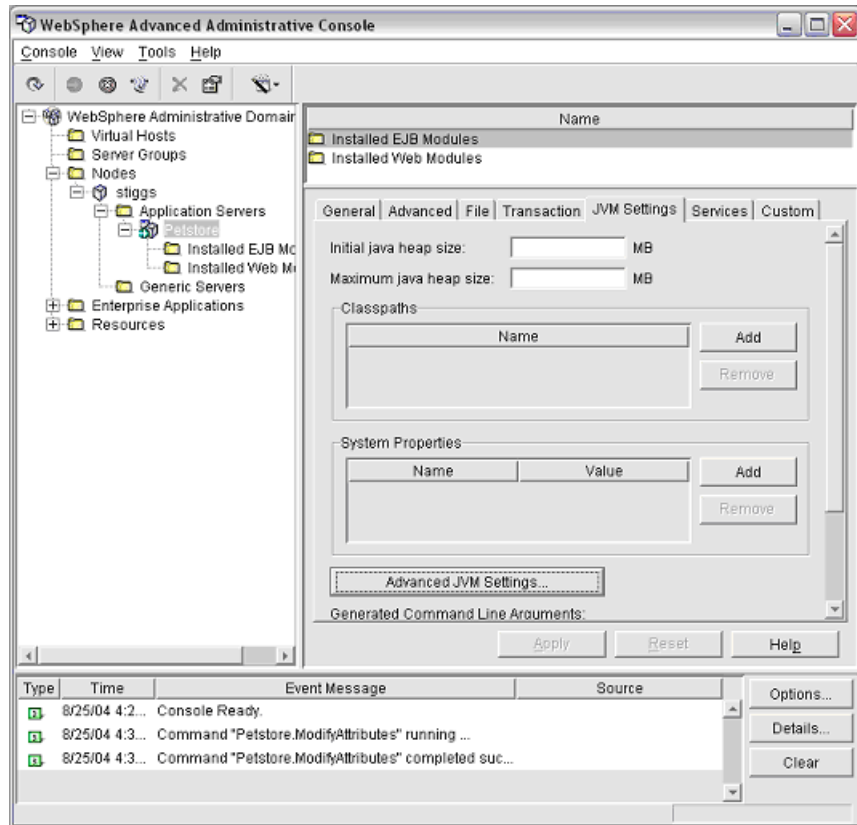
- 1 Use your Web browser to access the Web page for the WebSphere Administrative Console for the application server instance for which the probe was installed:

```
<WebSphere_Install_Dir>\AppServer\bin\adminclient.sh(bat)
```

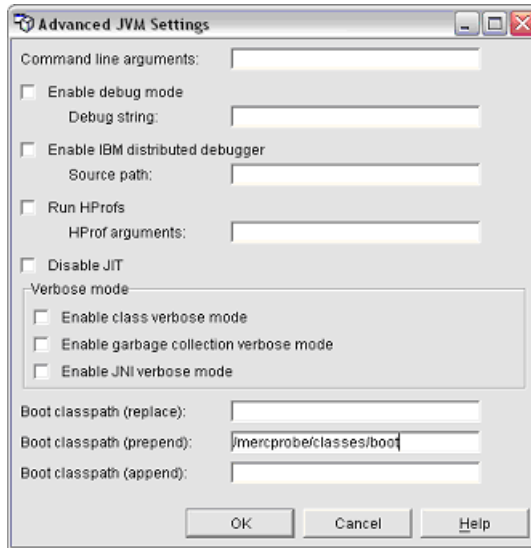
- 2 From the tree view under the **Nodes** parent, select the application server for the application to be monitored. The General tab for the selected application server is displayed to the right of the tree view.



- 3 Click the **JVM Settings** tab. The JVM Settings information is displayed as shown in the following example:



- 4 Click **Advanced JVM Settings**. The Advanced JVM Settings dialog box opens.



- 5 In the **Boot classpath (prepend)** box, type the boot classpath property as follows:

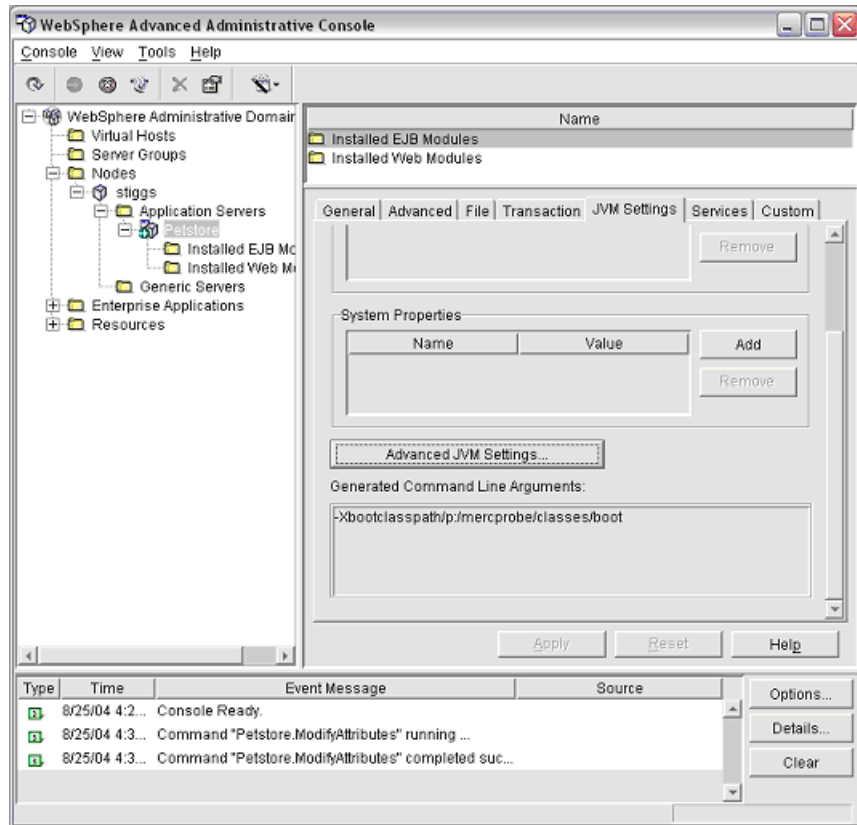
```
<probe_install_dir>\classes\IBM1.3.1_04;<probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

Note: If your JVM uses a JIT option, such as `-hotspot` or `-classic`, make sure that the `-Xbootclasspath` option is entered following the JIT option in the Java Command Line arguments on the Console.

Click **OK** to close the Advanced JVM Settings dialog box.

- 6 In **Generated Command Line Arguments** area of the JVM Settings tab, verify that the Xbootclasspath property was created properly.



The property should be set as follows:

```
"-Xbootclasspath/p:<probe_install_dir>\classes\IBM\1.3.1_04;<probe_install_dir>\classes\boot"
```

Note: When modifying the `-Xbootclasspath` parameter. Be sure to use quotes if there are spaces in the path that you specify.

- 7 Scroll down to display the command buttons, and click **Apply** to save your changes which will take effect when you restart the application server.
- 8 Restart the WebSphere application server from the Console menu. You do not need to restart the host for your application server.
- 9 To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>\log\<probe.id>\probe.log` file. If there are no entries in the file, either you did not run the JRE Instrumenter or you did not enter the `Xbootclasspath` correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

You can also refer to the Websphere `stdout.log` and `stderr.log` files for troubleshooting any problems arising from this configuration.

WebSphere 5.x and 6.x

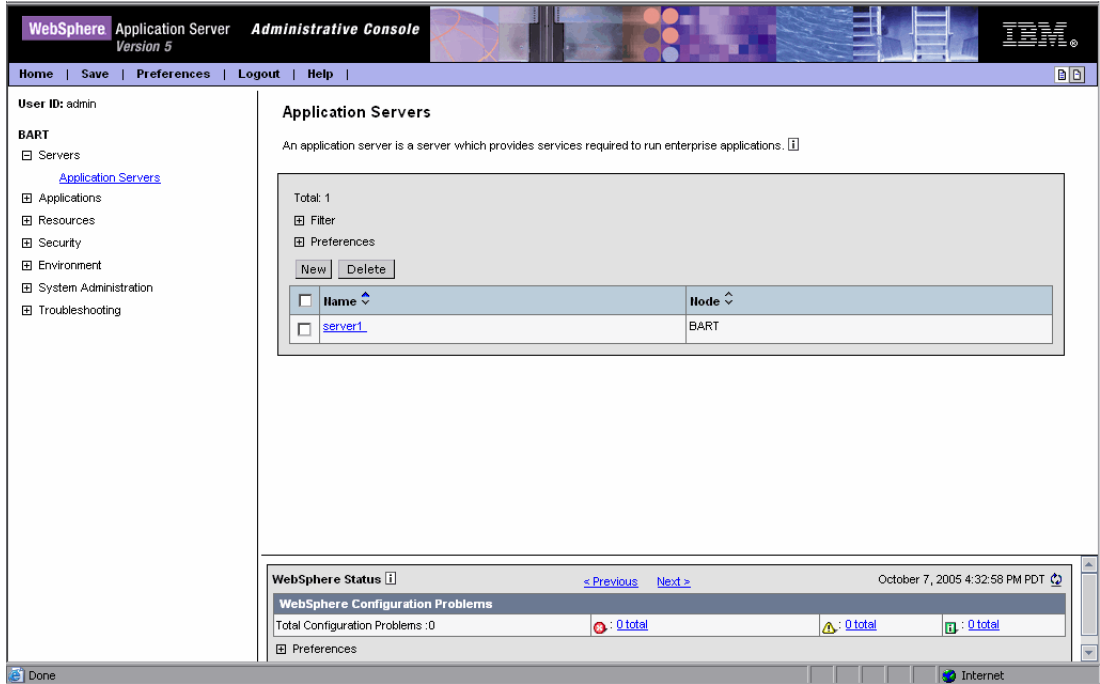
To configure a WebSphere 5.x and 6.x application server:

- 1 Use your Web browser to access the WebSphere Application Server Administrative Console for the application server instance for which the probe was installed:

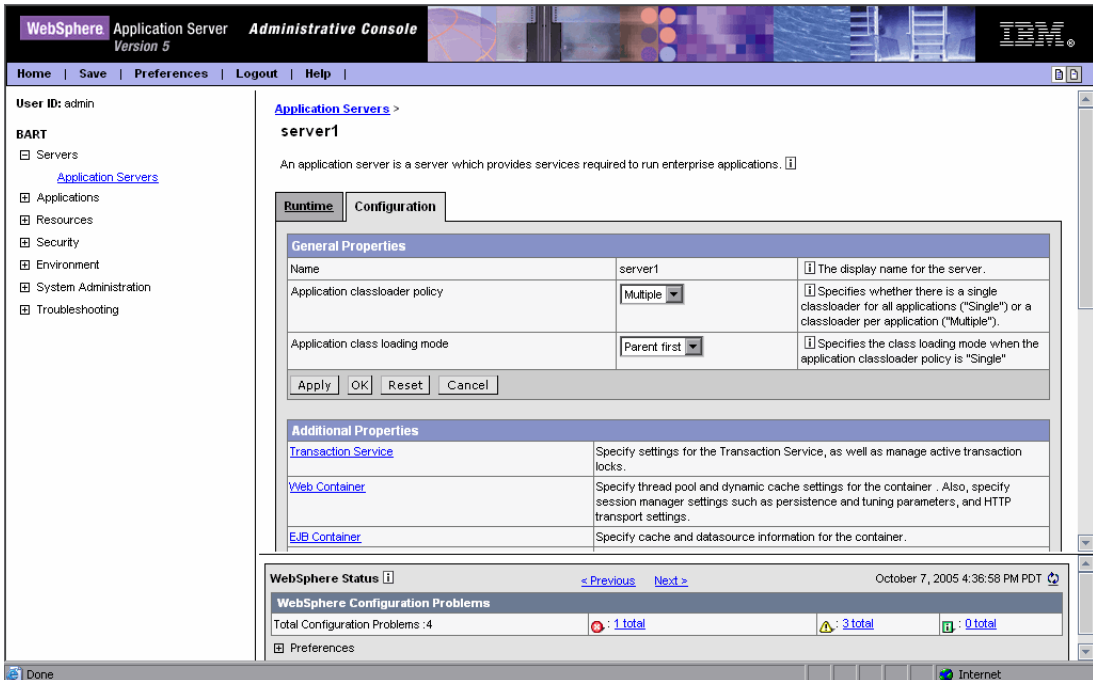
```
http://<App_Server_Host>:9090/admin
```

Replace `<App_Server_Host>` with the machine name for the application server host.

The Websphere Application Server Administrative Console opens.



- 2 In the left panel, select **Servers > Application Servers**.
 - 3 From the list of application servers in the right panel, select the name of the server that you want to configure so that it will be monitored by the probe.
- The Configuration tab for the selected application server is displayed.



4 Scroll down in the Configuration tab, and in the **General Properties** column, look for the **Process Definition** property.

The screenshot shows the WebSphere Administrative Console interface. The top navigation bar includes 'Home', 'Save', 'Preferences', 'Logout', and 'Help'. The left sidebar shows a tree view with 'Application Servers' selected. The main content area displays a list of configuration properties for the selected server. The 'Process Definition' property is highlighted with a red oval. Below the list is a 'WebSphere Status' section showing configuration problems.

Property Name	Description
EJB Container	Specify cache and datasource information for the container.
Dynamic Cache Service	Specify settings for the Dynamic Cache service of this server.
Logging and Tracing	Specify Logging and Trace settings for this server.
Message Listener Service	Configuration for the Message Listener Service. This service provides the Message Driven Bean (MDB) listening process, whereby MDBs are deployed against ListenerPorts that define the JMS destination to listen upon. These Listener Ports are defined within this service along with settings for its Thread Pool.
ORB Service	Specify settings for the Object Request Broker Service.
Custom Properties	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.
Administration Services	Specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.
Diagnostic Trace Service	View and modify the properties of the diagnostic trace service.
Debugging Service	Specify settings for the debugging service, to be used in conjunction with a workspace debugging client application.
IBM Service Logs	Configure the IBM service log, also known as the activity log.
Custom Services	Define custom service classes that will run within this server and their configuration properties.
Server Components	Additional runtime components which are configurable.
Process Definition	A process definition defines the command line information necessary to start/initialize a process.
Performance Monitoring Service	specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.

WebSphere Status | October 7, 2005 4:38:58 PM PDT

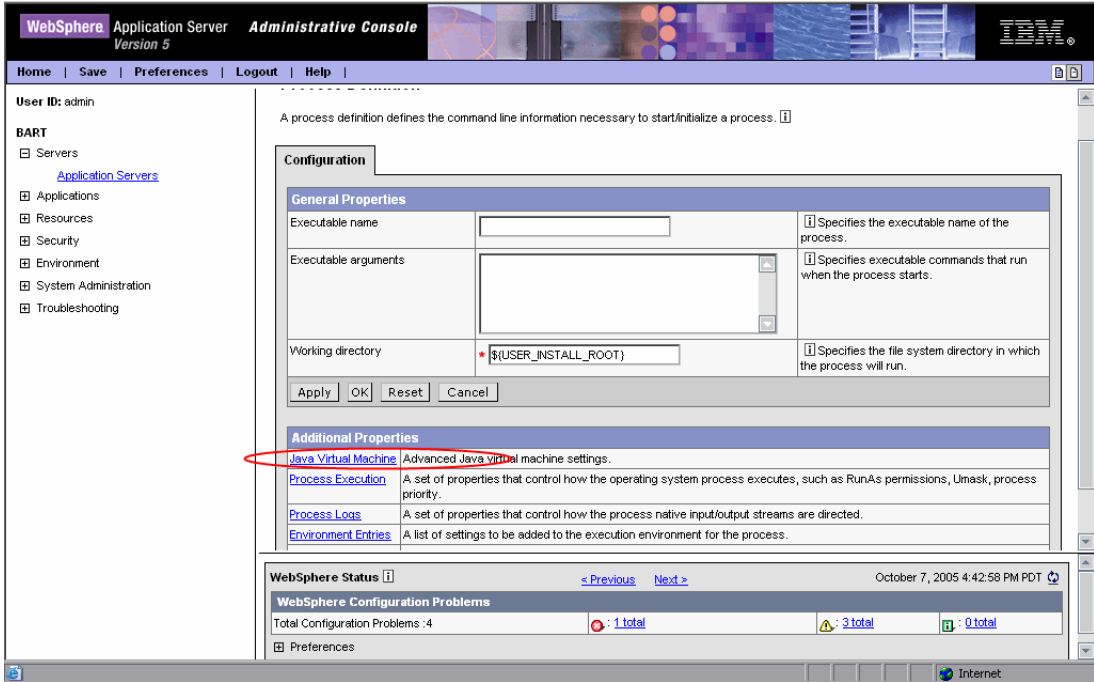
WebSphere Configuration Problems

Total Configuration Problems: 4	1 total	3 total	0 total
---------------------------------	---------	---------	---------

Done | Internet

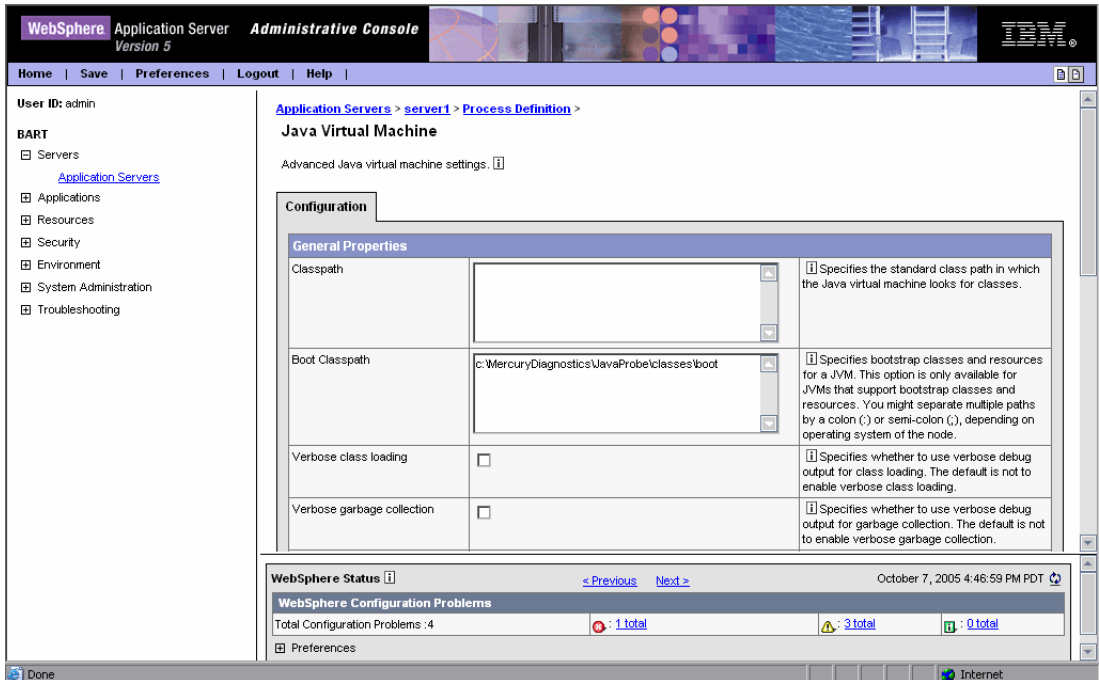
5 Click **Process Definition**.

6 Scroll down in the right panel, and look for **Java Virtual Machine**.



7 Click **Java Virtual Machine**.

8 The Configuration tab for the Java Virtual Machine is displayed.

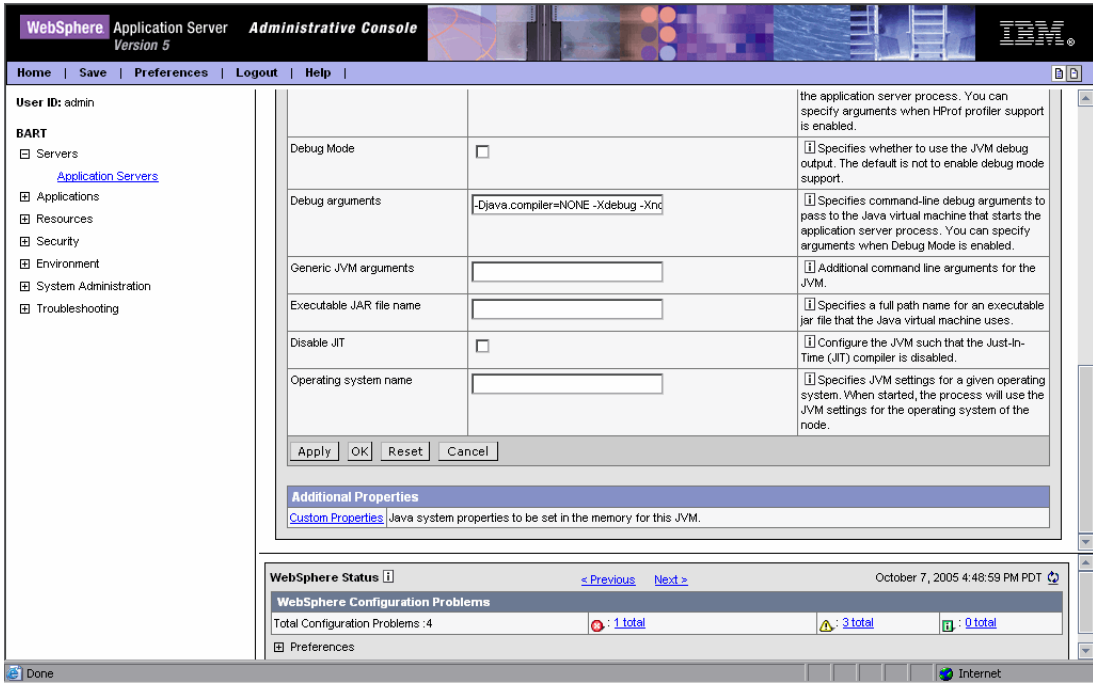


9 In the **Boot Classpath** box, type the path to the boot directory for the probe as follows:

```
<probe_install_dir>\classes\IBM1.4.2_06;<probe_install_dir>\classes\boot
```

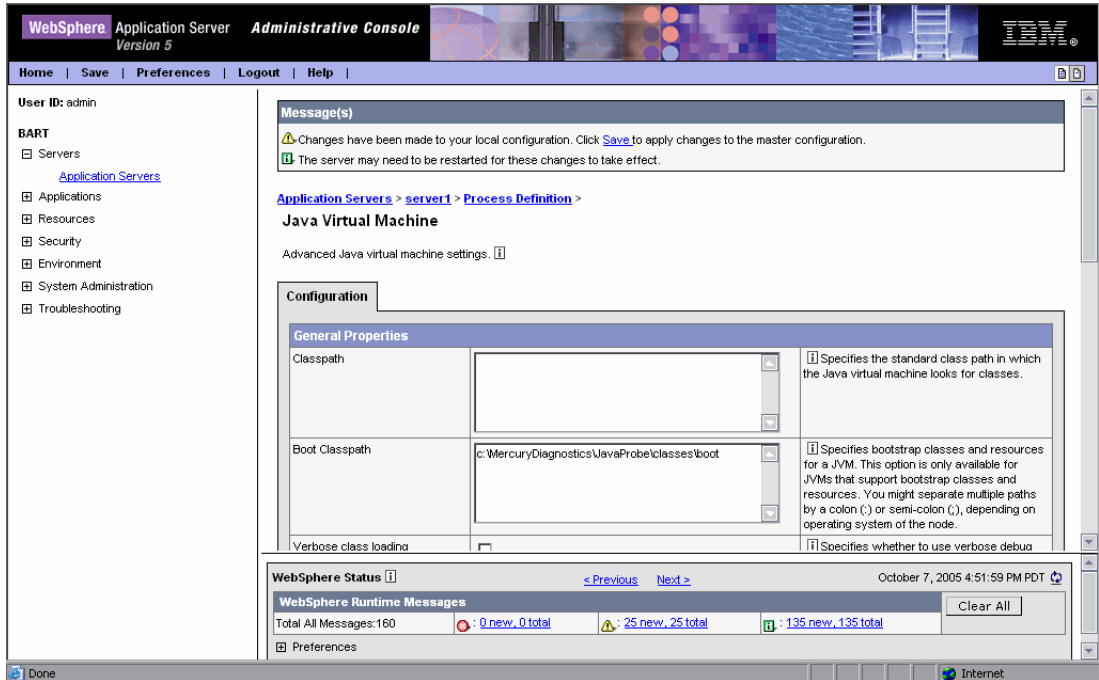
where **<probe_install_dir>** is the path to the location where the probe was installed.

10 Scroll to the bottom of the Configuration tab until the command buttons are visible.



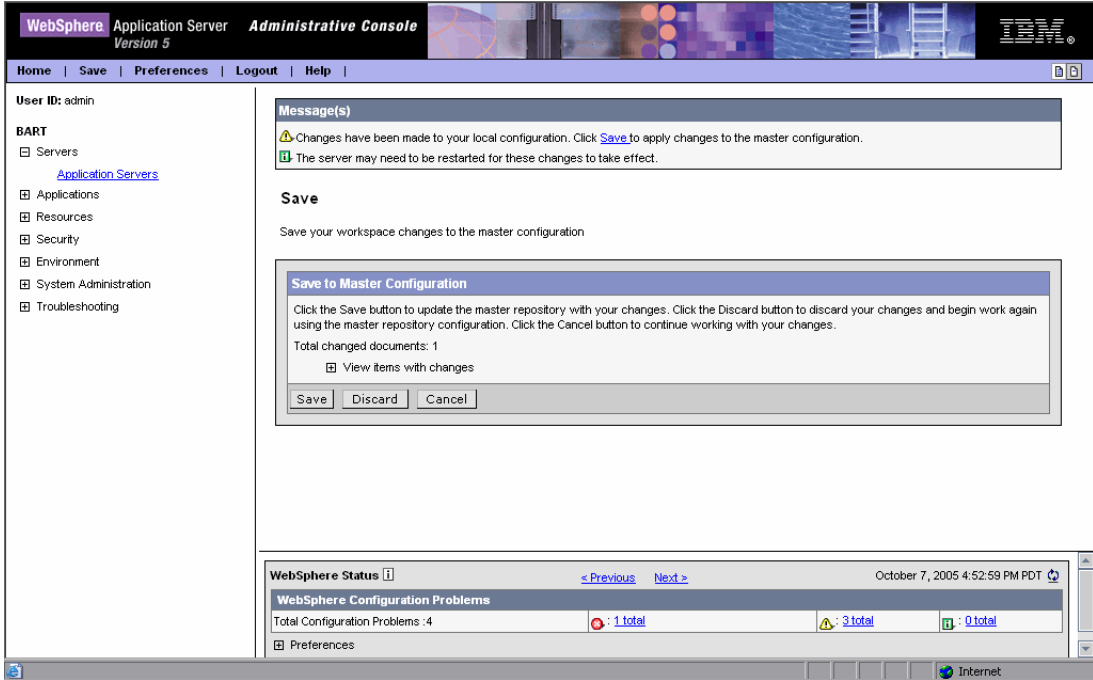
Click **Apply**.

11 A message confirms that your changes have been applied.



Click **Save** to apply the changes to the master configuration.

12 In the **Save to Master Configuration** area, click **Save**.



13 Restart the WebSphere application server. You do not need to restart the host for your application server.

14 To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>\log\<probe_id>.log` file. If there are no entries in the file, this indicates that either you did not run the JRE Instrumenter or you did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

Running the JRE Instrumenter for WebSphere IDE

If you are using WebSphere IDE, you must run the JRE Instrumenter manually in order to make sure that the correct JAVA executable for the WSAD IDE has been instrumented.

The WSAD IDE has 10 different java.exe executables to choose from. You must make sure to instrument the one that is used to run the IDE.

To instrument the correct java.exe:

- 1 Determine the version of WebSphere that you are using.
- 2 Determine the location of the appropriate java.exe. See the table below.
- 3 Run the JRE Instrumenter as described in “Running the JRE Instrumenter” on page 155.

Version	Executable
WAS 5.0	IDE INSTALL\runtimes\base_v5\java\bin\java.exe
WAS 5.1	IDE INSTALL\runtimes\base_v51\java\bin\java.exe
WAS 4.x	IDE INSTALL\runtimes\aes_v4\java\bin

Note: Before you can capture data, you must modify the Xbootclasspath for the application’s JVM startup parameters per your IDE’s instructions.

If you are NOT running your application inside the IDE, follow the instructions below to configure the JVM parameters using the WebSphere Administrative Console.

Configuring WebLogic Application Servers

Supported versions	Operating systems
6.1, 7.0, 8.1, 9.0	Windows 2000 SP6 Solaris 8

WebLogic application servers are configured by adding the **Xbootclasspath** property to the script that is used to start the application server. WebLogic is started by running shell scripts in a UNIX environment, or command scripts in a Windows environment. Because the startup scripts that WebLogic provides are frequently customized by a site administrator, it is not possible to provide detailed configuration instructions that apply to all situations. Instead, the following sections provide instructions for each of the certified versions of the WebLogic application server for a generic implementation. Your site administrator should be able to use these instructions to show you how to make these changes in your customized environment.

Note: Make sure you understand the structure of the startup scripts, how the property values are set, and how to use environment variables before you make any configuration changes for the probe. Always create a backup copy of any file that you are going to update prior to making the changes.

WebLogic 6.1

To configure a WebLogic 6.1 application server:

- 1 Locate the startup script used to start WebLogic for your domain. This file is typically located in a path similar to this example:

```
D:\bea\wlserver6.1\config\\start<Dom_Name>.cmd
```

Replace **<Dom_Name>** with the name of the script that starts your application.

For example, if your domain name is `petstore`, the path looks like this:

```
D:\bea\wlserver6.1\config\petstore\startPetStore.cmd
```

- 2 Create a backup copy of the startup script prior to making any changes to the script.
- 3 Use your editor to open the startup script.
- 4 Add the **Xbootclasspath** parameter to the Java command line that starts the application server. The parameter must be placed at the beginning of the Java parameters following any JIT options such as **-hotspot** or **-classic**.

Following is an example of the **Xbootclasspath** parameter:

```
<probe_install_dir>\classes\Sun\1.3.1_04;<probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is with the path to the directory where the probe was installed.

Following is an example of a WebLogic startup script before adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="C:\bea" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

Following is an example of a WebLogic startup script after adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m
-Xbootclasspath/p:"C:\Program Files\Mercury
Interactive\common\JavaProbe\classes\boot"
-classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="C:\bea" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\wlsrserver6.1/lib/weblogic.policy" weblogic.Server
```

- 5 Save the changes to the startup script.
- 6 Restart the WebLogic application server. You do not need to reboot the host machine, just the application server.

Note: To verify that the probe was configured correctly, check for entries in the **<probe_install_dir>/log/<probe_id>/probe.log** file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the Xbootclasspath correctly.

Note: For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

WebLogic 7.0

Note: For WebLogic 7.0 sp06 you must also configure JRockit as described in the section on configuring a WebLogic 8.1 application sever for the JRockit JVM, on page 190.

To configure a WebLogic 7.0 application server:

- 1 Locate the startup script used to start WebLogic for your domain. This file is typically located in a path similar to this example:

```
D:\bea\weblogic700\server\config\\start<Dom_Name>.cmd
```

Replace **<Dom_Name>** by the name of the script that starts your application. For example, if your domain name is `petstore`, the path looks like this:

```
D:\bea\ weblogic700\server\config\petstore\startPetStore.cmd
```

- 2 Create a backup copy of the startup script prior to making any changes to the script.
- 3 Use your editor to open the startup script.
- 4 To change the script, add the following immediately before the call to **startWLS.cmd** (at the end of the file):

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

For Windows:

```
set JAVA_OPTIONS="-Xbootclasspath/p:<probe_install_dir>\
classes\Sun\1.3.1_04;<probe_install_dir>\classes\boot" %JAVA_OPTIONS%
@rem Call WebLogic Server
call "C:\bea7\weblogic700\server\bin\startWLS.cmd"
```

For UNIX:

```
export JAVA_OPTIONS="-
Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.3.1_04;<probe_install_dir>\class
s\boot" $JAVA_OPTIONS
# Call WebLogic Server
call /bea/weblogic700/server/bin/startWLS.sh"
```

- 5 Save the changes to the startup script.
- 6 Restart the WebLogic application server. You do not need to reboot the host machine for your application server.
- 7 To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>/log/<probe_id>/probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the Xbootclasspath correctly.

Note: For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

WebLogic 8.1

To configure a WebLogic 8.1 application server for the Sun JVM:

- 1 Locate the startup script used to start WebLogic for your domain. This file is typically located in a path similar to this example:

```
D:\bea\weblogic81\config\<Dom_Name>\start<Dom_Name>.cmd
```

Replace **<Dom_Name>** by the name of the script that starts your application. For example, if your domain name is `petstore`, the path looks like this:

```
D:\bea\weblogic81\config\petstore\startPetStore.cmd
```

- 2 Create a backup copy of the startup script prior to making any changes to the script.
- 3 Use your editor to open the startup script.
- 4 Add the **Xbootclasspath** parameter to the Java command line that starts the application server. The parameter must be placed at the beginning of the Java parameters following any JIT options, such as **-hotspot** or **-classic**.

Following is an example of the **Xbootclasspath** parameter:

```
JAVA_OPTIONS="-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;<probe_install_dir>\classes\boot"
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

Following is an example of a WebLogic startup script before adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m"-
Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.3.1_04;<probe_install_dir>\classes\boot" -classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer
-Dbea.home="C:\\bea"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\weblogic81\lib\weblogic.policy" weblogic.Server
```

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

Following is an example of a WebLogic startup script after adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -Xbootclasspath/p:"C:\Program Files\Mercury Interactive\common\JavaProbe\classes\boot"-classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer -Dbea.home="C:\bea"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\weblogic81\lib\weblogic.policy" weblogic.Server
```

- 5 Save the changes to the startup script.
- 6 Restart the WebLogic application server. You do not need to restart the application server host machine.
- 7 To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>\log\<probe_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter, or did not enter the Xbootclasspath correctly.

Note: For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

To configure a WebLogic 8.1 application server for the JRockit JVM:

- 1 Locate the command file that invokes the WebLogic application server (for example, `startWLS.cmd`). This file is typically located in a path similar to the following example:

```
C:\bea\weblogic81\server\bin\startWLS.cmd
```

- 2 Create a backup copy of the command file prior to making any changes to the script. You may want to give the new copy a name such as `startWLSWithJRockit.cmd`, and use this as the new version of the command file that will be manipulated in the following steps.
- 3 Use your editor to open the startup script.

- 4 Set the JAVA executable invoked by WebLogic to JRockit.
 - a Locate the line in the command file where the value of the **JAVA_VENDOR** parameter is set.
 - b Change the value of the **JAVA_VENDOR** variable to point to the JRockit folder as follows:

```
set JAVA_VENDOR=<BEA_HOME_DIR>\jrockit
```

For example:

```
set JAVA_VENDOR=BEA
```

- 5 Modify the Java command line that starts the application server.
 - a Locate the line in the command file which begins as follows:

```
%JAVA_HOME%\bin\java %JAVA_VM% %JAVA_OPTIONS% .....
```

- b Indicate the JRockit management URL by specifying the **Xmanagement:class** parameter immediately following the **%JAVA_OPTIONS%** variable.

The following is an example of the **Xmanagement:class** parameter:

```
-Xmanagement:class=com.mercury.opal.capture.proxy.JRockitManagement
```

- c Allow the probe to hook into the application server process by adding the **Xbootclasspath** parameter immediately following the **%JAVA_OPTIONS%** variable.

Following is an example of the **Xbootclasspath** parameter:

```
-Xbootclasspath/p:<probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

The following is an example of a WebLogic startup script before adding the **Xmanagement:class** and **Xbootclasspath** parameters:

```
"%JAVA_HOME%\bin\java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%  
-Dweblogic.Name=%SERVER_NAME%  
-Dweblogic.management.username=%WLS_USER%  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.management.server=%ADMIN_URL%  
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%  
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy" weblogic.Server
```

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

The following is an example of a WebLogic startup script after adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" %JAVA_VM% %MEM_ARGS% %JAVA_OPTIONS%  
-Xmanagement:class=com.mercury.opal.capture.proxy.JRockitManagement  
-Xbootclasspath/p:"C:\Program Files\Mercury Interactive\common\JavaProbe\classes\boot"  
-Dweblogic.Name=%SERVER_NAME%  
-Dweblogic.management.username=%WLS_USER%  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.management.server=%ADMIN_URL%  
-Dweblogic.ProductionModeEnabled=%PRODUCTION_MODE%  
-Djava.security.policy="%WL_HOME%\server\lib\weblogic.policy" weblogic.Server
```

- 6** Save the changes to the command file.
- 7** Restart the WebLogic application server (not the computer; just the application server).
- 8** To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>\log\<probe_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the **Xbootclasspath** parameter correctly.

Note: For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

Configuring the Oracle9i Application Server

Supported version	Operating systems
9.0.3	Solaris 8

Oracle9i application servers are configured by adding the `Xbootclasspath` property to the XML file used to start the application server. Because the files that Oracle9i provides are frequently customized by the site administrator, it is not possible to provide detailed configuration instructions that will apply exactly for each situation. Instead, the following sections provide instructions configuring an Oracle9i application server for a generic implementation. Your site administrator should be able to use these instructions to guide you through making these changes in your customized environment.

Note: Make sure that you understand the structure of the startup scripts, how the property values are set, and the use of environment variables before you make any configuration changes for the probe. Always create a backup copy of any file that you are going to update, prior to making the changes.

To configure an Oracle9i application server:

- 1** Locate the XML file that is used to control the configuration of the application server when the server is started. This file is typically located at `<Oracle 9iAS_Install_Dir>/opmn/conf/opmn.xml`.
- 2** Create a backup copy of the `opmn.xml` file prior to making any changes.
- 3** Open the `opmn.xml` file to be edited using your editor.

- 4 Add the **Xbootclasspath** property. The property must be added to the “java-option value.”

The following is an example of the **Xbootclasspath** parameter:

```
-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;
  <probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

Note: When modifying the **-Xbootclasspath** parameter. Be sure to use quotes if there are spaces in the path that you specify.

The following image is an example of an Oracle 9iAS startup script before adding the **Xbootclasspath** parameter:

```
- <ias-instance xmlns="http://www.oracle.com/ias-instance">
- <notification-server>
  <port local="6100" remote="6200" request="6003" />
  <log-file path="/opt/oracle/ora9ias/opmn/logs/ons.log" level="3" />
</notification-server>
- <process-manager>
  - <ohs gid="HTTP Server" maxRetry="3">
    <start-mode mode="ssl" />
  </ohs>
  - <oc4j instanceName="home" numProcs="1" maxRetry="3">
    <config-file path="/opt/oracle/ora9ias/j2ee/home/config/server.xml" />
    <oc4j-option value="-properties" />
    <port ajp="3000-3100" mmi="3101-3200" jms="3201-3300" />
  - <environment>
    <prop name="LD_LIBRARY_PATH" value="/opt/oracle/ora9ias/lib" />
  </environment>
</oc4j>
  - <oc4j instanceName="OC4J_Demos" gid="OC4J_Demos">
    <config-file path="/opt/oracle/ora9ias/j2ee/OC4J_Demos/config/server.xml" />
    <java-option value="-Xmx512M" />
    <oc4j-option value="-userThreads -properties" />
    <port ajp="3001-3100" mmi="3101-3200" jms="3201-3300" />
  - <environment>
    <prop name="%LIB_PATH_ENV%" value="%LIB_PATH_VALUE%" />
  </environment>
</oc4j>
- <custom gid="dcm-daemon" numProcs="1" noGidWildcard="true">
  <start path="/opt/oracle/ora9ias/dcm/bin/dcmctl daemon -logdir /opt/oracle/ora9ias/dcm/logs/daemon_logs" />
  <stop path="/opt/oracle/ora9ias/dcm/bin/dcmctl shutdowndaemon" />
</custom>
  <log-file path="/opt/oracle/ora9ias/opmn/logs/lpm.log" level="3" />
</process-manager>
</ias-instance>
```


The following image is an example of an Oracle 9iAS startup script after adding the `Xbootclasspath` parameter:

```

- <ias-instance xmlns="http://www.oracle.com/ias-instance">
- <notification-server>
  <port local="6100" remote="6200" request="6003" />
  <log-file path="/opt/oracle/ora9ias/opmn/logs/ons.log" level="3" />
</notification-server>
- <process-manager>
- <ohs gid="HTTP Server" maxRetry="3">
  <start-mode mode="ssl" />
</ohs>
- <oc4j instanceName="home" numProcs="1" maxRetry="3">
  <config-file path="/opt/oracle/ora9ias/j2ee/home/config/server.xml" />
  <java-option value="-Xmx512m -Xbootclasspath/p:C:\Program
Files\MercuryInteractive\common\JavaProbe\classes\boot" />
  <oc4j-option value="-properties" />
  <port ajp="3000-3100" mi="3101-3200" jms="3201-3300" />
  <environment />
</oc4j>
- <oc4j instanceName="OC4J_Demos" gid="OC4J_Demos">
  <config-file path="/opt/oracle/ora9ias/j2ee/OC4J_Demos/config/server.xml" />
  <oc4j-option value="-userThreads -properties" />
  <port ajp="3001-3100" mi="3101-3200" jms="3201-3300" />
  <environment>
    <prop name="%LIB_PATH_ENV%" value="%LIB_PATH_VALUE%" />
  </environment>
</oc4j>
- <custom gid="dcm-daemon" numProcs="1" noGidWildcard="true">
  <start path="/opt/oracle/ora9ias/dcm/bin/dcmctl daemon -logdir /opt/oracle/ora9ias/dcm/logs/daemon_logs" />
  <stop path="/opt/oracle/ora9ias/dcm/bin/dcmctl shutdowndaemon" />
</custom>
  <log-file path="/opt/oracle/ora9ias/opmn/logs/ipm.log" level="3" />
</process-manager>
</ias-instance>

```

- 5 Save the changes to the XML file.
- 6 Restart the Oracle application server. You do not need to reboot the host for the application server.
- 7 To verify that the probe was configured correctly, check for entries in the `<probe_install_dir>\log\<probe_id>\probe.log` file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the `Xbootclasspath` parameter correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

Configuring the JBoss Application Server

This section explains how to configure the JBoss 3.2.1 application server.

Supported versions	Operating systems
3.0.4	Windows 2000 Server
3.2.1	Linux Debian

You configure JBoss application servers by adding the **Xbootclasspath** property to the script that is used to start the application server. JBoss is started by running shell scripts in a UNIX environment, or command scripts in a Windows environment. Because the startup scripts that JBoss provides are frequently customized by the site administrator, it is not possible to provide detailed configuration instructions that will apply exactly for each situation. Instead, the following sections provide instructions for each of the certified versions of the JBoss application server for a generic implementation. Your site administrator should be able to use these instructions to guide you to make these changes in your customized environment.

Note: Make sure that you understand the structure of the startup scripts, how the property values are set, and the use of environment variables before you make any configuration changes for the probe. Always create a backup copy of any file that you are going to update prior to making the changes.

To configure a JBoss application server:

- 1 Locate the startup script that is used to start JBoss for your application. This file is typically located in path similar to the following example:

```
D:\JBoss\bin\run.bat
```

Note: For UNIX the file extension is “.sh”.

- 2 Create a backup copy of the startup script prior to making any changes to the script.
- 3 Open the startup script to be edited using your editor.
- 4 Add the **Xbootclasspath** parameter to the Java command line that starts the application server. The parameter must be placed at the beginning of the Java parameters following any JIT options, such as **-hotspot** or **-classic**.

The following is an example of the **Xbootclasspath** parameter:

```
-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;  
  <probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

Note: When modifying the **-Xbootclasspath** parameter. Be sure to use quotes if there are spaces in the path that you specify.

The following is an example of a JBoss startup script before adding the **Xbootclasspath** parameter:

```
"%JAVA%" %JAVA_OPTS% -classpath "%CLASSPATH%" org.jboss.Main %ARGS%
```

The following is an example of a JBoss startup script after adding the **Xbootclasspath** parameter:

```
"%JAVA%" "-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;  
          <probe_install_dir>\classes\boot" %JAVA_OPTS%  
-classpath "%CLASSPATH%" org.jboss.Main %ARGS
```

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

- 5 Save the changes to the startup script.
- 6 Restart the JBoss application server with the probe by running the modified script. You do not need to restart the application server host.
- 7 To verify that the probe was configured correctly, check for entries in the **<probe_install_dir>/log/<probe_id>/probe.log** file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the **Xbootclasspath** parameter correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

Configuring the SAP NetWeaver Application Server

The following instructions describe how to configure the SAP NetWeaver application server so that your applications can be monitored by the probe.

Supported versions	Operating systems
3.0.4	Windows 2000 Server
3.2.1	Linux Debian

Configuring the NetWeaver application server means instrumenting the JVM and adding the **Xbootclasspath** property to the script that is used to start the application server. The following sections provide instructions for a generic NetWeaver implementation. Your site administrator should be able to use these instructions to guide you in making the changes that are appropriate to your specific environment.

Note: Make sure that you understand the structure of the startup scripts, how the property values are set, and the use of environment variables before you make any changes to the configuration of your application server for the probe. You should always create a backup of files before making any changes.

To configure a SAP NetWeaver application server:

- 1** Add the JVM that runs the NetWeaver application server.
- 2** Instrument the JVM. The JRE Instrumenter provides the **Xbootclasspath** parameter. This parameter must be added to the NetWeaver Configtool's JMV parameters window as described in the next step.
- 3** Run the NetWeaver application server configuration tool. The configuration tool is called **configtool.bat** and it is located in the **usr\sap\j2e\j2ee\configtool** directory.
- 4** Add the **-Xbootclass** parameter into the Java parameters text window. This window is in the General tab when you select your server instance. For example, cluster-data | instance_ID70323 | server_ID7032350.

- 5 Save your changes and exit the configuration tool.
- 6 Assign the following values to these properties in the **etc/capture.properties** file:
 - minimum.buffer.size = 250000**
 - initial.private.buffer.count = 50**
 - maximum.private.buffer.count = 200**
 - gentle.reserve.buffer.count = 50**
 - hard.reserve.buffer.count = 50**
- 7 Restart the NetWeaver application server.
- 8 To verify that the probe was configured correctly, check for entries in the **<probe_install_dir>/log/<probe_id>/probe.log** file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the **Xbootclasspath** parameter correctly.

Configuring a Generic Application Server

Note: You should only use the instructions for a generic application server when you do not find configuration instructions for your specific application server in this document.

Your site administrator may configure the application server using an alternative, site-specific method. In this case, the generic procedure may be sufficient for the administrator to understand what changes must be made.

Important: Before making any changes, back up all startup scripts.

To update the application server configuration:

- 1** Locate the application server startup script or the file where the JVM parameters are set.
- 2** Create a backup copy of the application server startup script before you make any changes to the script.
- 3** Use an editor or the application server console to open the startup script.
- 4** Add the **Xbootclasspath** parameter to the Java command line that starts the application server, using the following syntax:

```
-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;  
  <probe_install_dir>\classes\boot
```

where **<probe_install_dir>** is the path to the directory where the probe was installed.

This connects the probe to the application. The parameter must be placed at the beginning of the Java parameters, following any JIT options such as **-hotspot** or **-classic**.

Following is an example of a WebLogic Java command line in a startup script before adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath "%CLASSPATH%"  
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer -Dbea.home="C:\bea"  
-Dweblogic.management.password=%WLS_PW%  
-Dweblogic.ProductionModeEnabled=%STARTMODE%  
-Dcloudscape.system.home=./samples/eval/cloudscape/data  
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

Note: The startup script examples are shown with line breaks. The actual scripts do not have line breaks and the text of the commands will wrap on your screen as necessary.

The following is an example of a WebLogic Java command line in a startup script after adding the **Xbootclasspath** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m  
"-Xbootclasspath/p:<probe_install_dir>\classes\Sun\1.4.2_04;  
<probe_install_dir>\classes\boot"  
-classpath "%CLASSPATH%"  
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer  
-Dbea.home="C:\bea" -Dweblogic.management.password=%WLS_PW%  
-Dweblogic.ProductionModeEnabled=%STARTMODE%  
-Dcloudscape.system.home=./samples/eval/cloudscape/data  
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

- 5 Save the changes to the startup script.
- 6 Restart the application server under test. You do not need to restart the application server host machine.
- 7 To verify that the probe was configured correctly, check for entries in the **<probe_install_dir>/log/<probe_id>/probe.log** file. If there are no entries in the file, this indicates that you did not run the JRE Instrumenter or did not enter the **Xbootclasspath** parameter correctly.

Note: Alternatively, you can configure the JVM process definitions of the application server by going into the Administrative Console. However, this does not apply to WebLogic servers.

For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 155.

Adjusting the Heap Size for the J2EE Probe in the Application Startup Script

The size of the heap can impact the performance of the J2EE Probe.

The default value for the heap size is 64 MB. The heap size is set in the application’s start up script using the following VM argument:

```
-Xmx<size>
```

You can increase the heap size by updating the value specified in the **-Xmx<size>** option. Refer to your JVM documentation for more help on setting this parameter.

Part IV

Setting Up Integration with Other Mercury Products

10

Setting Up Mercury Business Availability Center 6.1 or Later to Use Diagnostics

To begin using Diagnostics in Mercury Business Availability Center, you enter the Diagnostics Server details and configure the relevant Mercury Business Availability Center components to view Diagnostics data.

This chapter describes:	On page:
About Setting Up Mercury Business Availability Center to use Diagnostics	208
Specifying the Diagnostics Server Details	209
Changing the Diagnostics Server Details	213
Understanding the Diagnostics Configuration Page	213
Enabling the Diagnostics View	215
Enabling Diagnostics Monitoring in Business Process Monitor (BPM) Related Views	217
Assigning Permissions for Diagnostics Users	219
Accessing the Diagnostics Pages in Windows 2003	221

About Setting Up Mercury Business Availability Center to use Diagnostics

Mercury Diagnostics 6.5 can be integrated with Mercury Business Availability Center 6.1 or later to provide information to help you understand and improve the performance of your applications.

The process involved in setting up Mercury Business Availability Center to use Mercury Diagnostics differs according to the version of Mercury Business Availability Center that you are using.

To set up Diagnostics in Mercury Business Availability Center 6.1:

1 Specify the Diagnostics Server details.

Enter the Diagnostics Server details in Mercury Business Availability Center. For more information, see “Specifying the Diagnostics Server Details” on page 209.

2 Enable the Diagnostics View.

Enable the Diagnostics View in Dashboard and in the IT Universe tab. For more information, see “Enabling the Diagnostics View” on page 215

3 Enable Diagnostics monitoring in Business Process Monitor related views.

Change the Business Process Monitoring (BPM) source template to enable Diagnostics monitoring in BPM related views. For more information, see “Enabling Diagnostics Monitoring in Business Process Monitor (BPM) Related Views” on page 217.

4 Assign relevant permissions.

Grant different permissions to different Diagnostics users. (This procedure is optional.) For more information, see “Assigning Permissions for Diagnostics Users” on page 219.

To set up Diagnostics in Mercury Business Availability Center 6.2 or later:

1 Specify the Diagnostics Server details.

Enter the Diagnostics Server details in Mercury Business Availability Center. For more information, see “Specifying the Diagnostics Server Details” on page 209.

2 Assign relevant permissions.

Grant different permissions to different Diagnostics users. (This procedure is optional.) For more information, see “Assigning Permissions for Diagnostics Users” on page 219.

Specifying the Diagnostics Server Details

To make Mercury Diagnostics accessible from Mercury Business Availability Center, you specify the Diagnostics Server details.

Note: If you are using Windows 2003, you need to configure your Internet browser settings to access the Diagnostics Configuration page. For more details, see “Accessing the Diagnostics Pages in Windows 2003” on page 221.

To specify the Diagnostics Server details in Mercury Business Availability Center:

- 1 Log on to Mercury Business Availability Center.
- 2 Select **Admin > Diagnostics** to open the Diagnostics Configuration page.

Business Availability Center - Diagnostics Configuration User: administrator

Diagnostics Server Details

Make sure that the Diagnostics Server is accessible from the Business Availability Center machine and from users' Web browsers through the values you enter in the fields below.
Note: The values defined here are needed for application links and data connections.

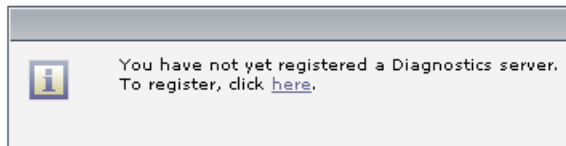
Enter Diagnostics Server details:

Diagnostics Server host name:

Diagnostics Server port number:

Diagnostics Server protocol:

Note: If you try to access Mercury Diagnostics (by clicking **Diagnostics** on the Site Map page or by clicking **Applications > Diagnostics**) before you have configured the Diagnostics Server, you receive a message instructing you to register the Diagnostics Server.



Click the link to open the Diagnostics Configuration page.

3 Enter the details for the Diagnostics Server in Commander mode.

- ▶ **Diagnostics server host name.** Enter the name of the machine that is host to the Diagnostics Server in Commander mode.

Note: Even if the Diagnostics Server is installed on the same machine as Mercury Business Availability Center, you still need to enter the actual name of the host in the **Diagnostics server host name** box. It is not sufficient to type **localhost** instead of the host name.

- ▶ **Diagnostics server port number.** Enter the port number used by the Diagnostics Server in Commander mode. The default port number is 2006.
- ▶ **Diagnostics server protocol.** Select the communication protocol through which Mercury Business Availability Center connects to **Diagnostics** - either **HTTP** or **HTTPS**.

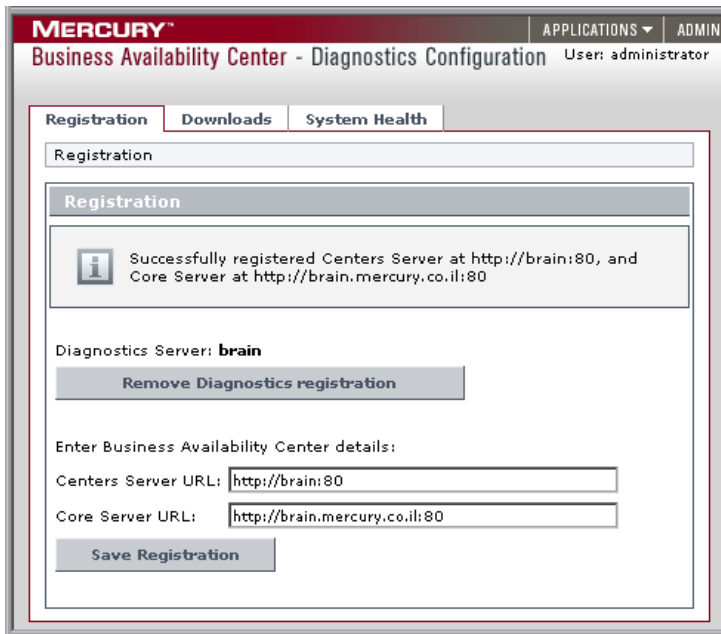
Note: If you select HTTPS as your communication protocol, additional configuration steps are required. For more information about the steps required, see Chapter 22, “Enabling HTTPS Between Diagnostics Components.”

4 After you have entered the Diagnostics Server details and ensured that these details are accurate, click **Submit** to complete the Diagnostics Server configuration process.

If the server name you have entered is incorrect or if the server is unavailable, an error message is displayed.

When you click **Submit**, the Mercury Diagnostics Server details are saved in Mercury Business Availability Center and the Mercury Business Availability Center server details are automatically registered on the Diagnostics Server machine.

The **Registration** tab in the Diagnostics Configuration page opens, displaying the Diagnostics Server details that you just specified and the Mercury Business Availability Center server details.



Where necessary, you can manually change the Mercury Business Availability Center server details in the **Enter Business Availability Center details** section of the Registration tab.

Note: Verify that the root URL in the **Centers Server URL** box, matches the root URL that you use to access Mercury Business Availability Center.

Changing the Diagnostics Server Details

You may want to change the Diagnostics Server details or remove the Diagnostics registration completely.

To remove the Diagnostics registration:

- 1** Select **Admin > Diagnostics**.
- 2** In the **Registration** tab, click **Remove Diagnostics Registration**.
- 3** In the message that opens, click **OK** to confirm that you want to remove the Diagnostics registration.

A message is displayed, confirming that you successfully removed the Diagnostics registration.

To register a new Diagnostics Server, select **Admin > Diagnostics** and follow the procedure explained in “Specifying the Diagnostics Server Details” on page 209.

Understanding the Diagnostics Configuration Page

You access the Diagnostics Configuration page by selecting **Admin > Diagnostics**. The Diagnostics Configuration page consists of the following three tabs, which are described in this section:

- Registration Tab
- Downloads Tab
- System Health Tab

Registration Tab

The Registration tab displays the following details:

- ▶ The Diagnostics Server details that you registered in Mercury Business Availability Center. To change these details, see “Changing the Diagnostics Server Details” on page 213.
- ▶ The Mercury Business Availability Center server details that were automatically registered on the Diagnostics Server machine. You can manually change the Mercury Business Availability Center server details in the **Enter BAC server details** section.

For information about registering Diagnostics in Mercury Business Availability Center for the first time, see “Specifying the Diagnostics Server Details” on page 209.

Downloads Tab

The Downloads tab provides links to the Mercury Diagnostics Probe installers, enabling you to download the Mercury Diagnostics Probe for your relevant platform.

If you did not specify the path to the Probe installers during the installation of the Mercury Diagnostics Server, the Downloads tab will not display any components. For more information, see Chapter 2, “Installing the Mercury Diagnostics Server.”

System Health Tab

Provides you with a map of all the components of your Mercury Diagnostics deployment and gives you real-time status and health information for each component. For detailed information about the System Health monitor, see Appendix C, “Using the System Health Monitor.”

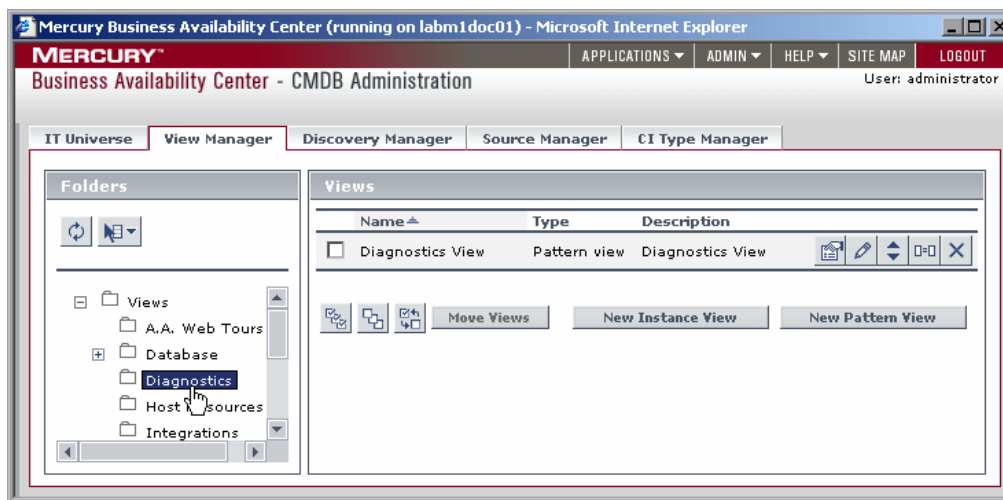
Enabling the Diagnostics View

Within Mercury Business Availability Center, both Dashboard and the IT Universe tab contain a *Diagnostics View* for working with Diagnostics data.

- ▶ In Mercury Business Availability Center 6.1, the Diagnostics View is not assigned by default and you therefore need to enable this view.
- ▶ In Mercury Business Availability Center 6.2 or later, the Diagnostics View is enabled by default.

To enable the Diagnostics View in Mercury Business Availability Center:

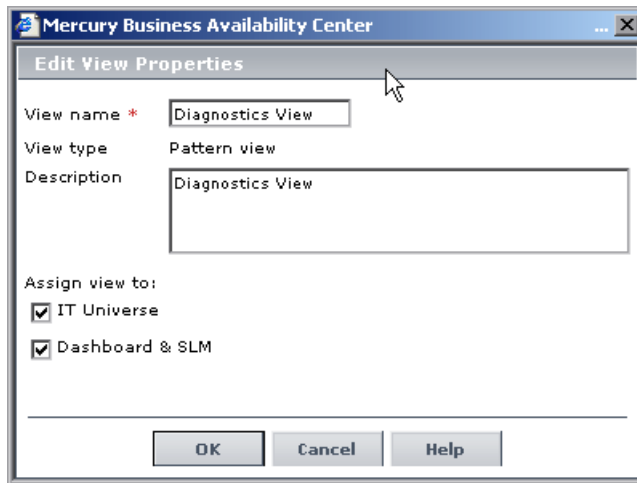
- 1** Select **Admin > CMDB** to open the CMDB Administration page.
- 2** Click the **View Manager** tab.
- 3** In the **Folders** pane (on the left), under the **Views** folder, click **Diagnostics**.





- 4 In the **Views** pane (on the right), click the **Edit view properties** button.

The Edit View Properties dialog box opens.



- 5 In the **Assign View to** section, select the **IT Universe** check box to enable the Diagnostics View in the IT Universe tab, and select the **Dashboard and SLM** check box to enable the Diagnostics View in Dashboard.

For more information about working with the Diagnostics View in Dashboard, see the *Mercury Diagnostics User's Guide*. For more information about working with Configuration Items in IT Universe, refer to *IT Universe Administration* in the *Mercury Business Availability Center Documentation Library*.

Enabling Diagnostics Monitoring in Business Process Monitor (BPM) Related Views

Important: This section does not apply to Mercury Business Availability Center 6.2 or later, where Diagnostics monitoring in BPM related views is enabled by default.

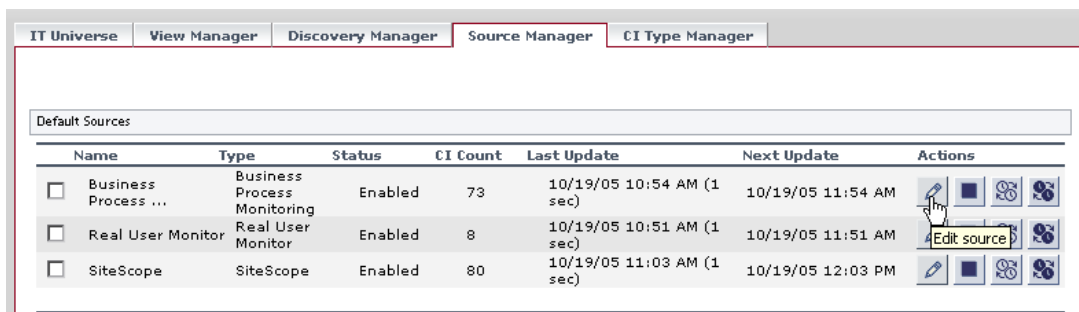
From Mercury Business Availability Center's Dashboard, you monitor the status of critical performance variables over time using Key Performance Indicators (KPIs). Diagnostics related KPIs are known as Application KPIs.







In Mercury Business Availability Center 6.1, you need to enable Application KPIs For BPM related views, before you can start using them to view the status of BPM transactions monitored by Diagnostics.

To enable Application KPIs in BPM related views, you replace the existing Business Process Monitoring source template with a modified template that has been provided with the Diagnostics product.

To enable Application KPIs in Business Process Monitor related views:

- 1 Select **Admin > CMDB** to open the CMDB Administration page.
- 2 Click the **Source Manager** tab to open the Source Manager.



Default Sources							
Name	Type	Status	CI Count	Last Update	Next Update	Actions	
<input type="checkbox"/> Business Process ...	Business Process Monitoring	Enabled	73	10/19/05 10:54 AM (1 sec)	10/19/05 11:54 AM		
<input type="checkbox"/> Real User Monitor	Real User Monitor	Enabled	8	10/19/05 10:51 AM (1 sec)	10/19/05 11:51 AM		
<input type="checkbox"/> SiteScope	SiteScope	Enabled	80	10/19/05 11:03 AM (1 sec)	10/19/05 12:03 PM		

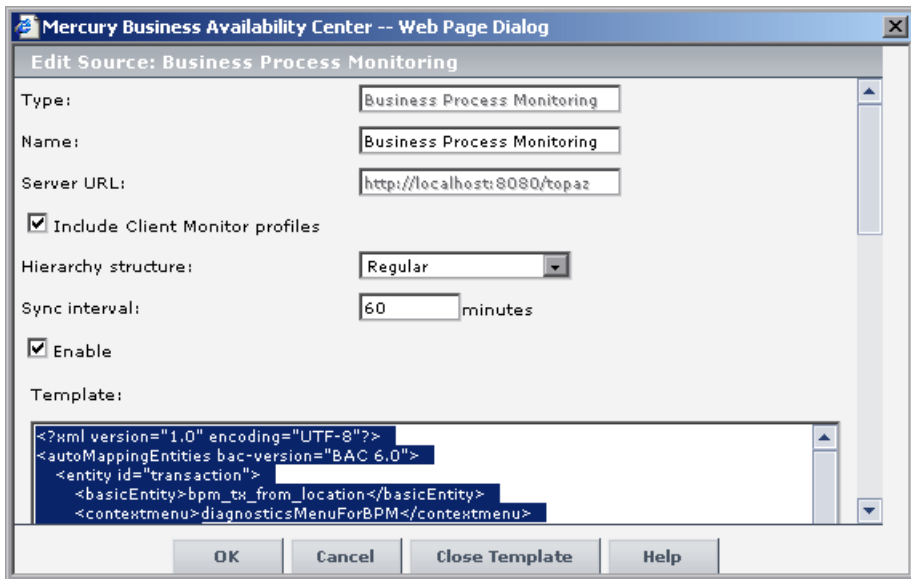


- 3 In the **Business Process Monitoring** row, click the **Edit Source** button to open the Edit Source dialog box.

- 4 Click the **Edit Template** button in the dialog box.

Mercury Business Availability Center displays a warning about making changes to the template. Click **OK**.

- 5 In the **Template** box, replace the existing template with the modified template as follows:
 - a Using a text editor, open the **BPM_template_with_Application_KPI.xml** file located in the Mercury Diagnostics CD-ROM in the **BAC_Integration** directory, and copy the entire file to the clipboard.
 - b In the **Template** box, delete all of the existing text and paste the new text that you copied from the modified template.



- c Click **OK**.

The new template is saved and Application KPIs will now be displayed, where relevant, in the BPM related views in Dashboard. This template modification procedure needs to be performed only once. Application KPIs are enabled for this session and for future sessions.

Assigning Permissions for Diagnostics Users

Mercury Business Availability Center enables you to apply permissions to users and user groups for specific resources that are defined in the system. There are specific types of Permission operations that administrators can grant Diagnostics users.

To assign permissions for Diagnostics Users in Mercury Business Availability Center:

- 1 Select **Admin > Platform** to open the Platform Administration page.
- 2 In the **Users and Permissions** tab, click **Permissions Management**.
- 3 In the left pane, in the **Select Context** list, select **Monitors**. The resource tree opens, displaying the resources included in the Monitors context.
- 4 In the resource tree, click **Diagnostics**.

The screenshot shows the Mercury Business Availability Center Platform Administration interface. The top navigation bar includes 'APPLICATIONS', 'ADMIN', 'HELP', 'SITE MAP', and 'LOGOUT'. The user is logged in as 'administrator'. The main navigation tabs are 'Setup and Maintenance', 'Data Collection', 'Alerts and Recipients', 'Scheduled Reports', and 'Users and Permissions'. The 'Users and Permissions' tab is active.

In the 'Users and Permissions' section, the 'Select context:' dropdown is set to 'Monitors'. The left pane shows a tree view of resources under 'Monitors', with 'Diagnostics' selected. The right pane shows the 'Resource: Diagnostics' configuration, with the 'Users' tab active. A table lists the users assigned to this resource:

Login Name	User Name
fist_administrator_1	fist_administrator_1
fist_administrator_2	fist_administrator_2
fist_superuser_1	fist_superuser_1
fist_superuser_2	fist_superuser_2
fist_systemmodifier_1	fist_systemmodifier_1
fist_systemmodifier_2	fist_systemmodifier_2
fist_systemviewer_1	fist_systemviewer_1
fist_systemviewer_2	fist_systemviewer_2

Below the user list, the 'Assign permissions to: fist_administrator_1' section is shown, with the 'Operations' tab active. A table lists the permissions granted to this user:

Operation	Grant	Granted from Group/Role/Parent	Inherit
Change	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
View	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Execute	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Full Control	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

An 'Apply Permissions' button is located at the bottom of the interface.

- 5 Select a user or user group in the user selection area on the upper right-hand area of the page.
- 6 In the **Operations** tab in the lower right-hand area of the page, select from the available operations for the highlighted resources.

Diagnostics administrators can grant users the following types of permission operations:

- **Change:** Enables viewing Diagnostics administration and configuring the Diagnostics settings.
 - **View:** Enables viewing the Diagnostics application when accessing Diagnostics from the Mercury Business Availability Center.
 - **Execute:** Enables changing the settings in the Mercury Diagnostics UI, such as setting thresholds.
 - **Full Control:** Enables performing all operations on Diagnostics, and granting and removing permissions for those operations.
- 7 Click **Apply Permissions** to complete the process.

For more detailed information about how to assign user permissions in Mercury Business Availability Center, refer to the section on “Configuring User Permissions” in *Platform Administration* in the *Mercury Business Availability Center Documentation Library*.

Accessing the Diagnostics Pages in Windows 2003

When your Internet browser is running in a Windows 2003 environment, you have to change your Internet browser settings in order to access the Diagnostics configuration and application pages in Mercury Business Availability Center.

To access the Diagnostics pages in a Windows 2003 environment:

- 1** In Internet Explorer, select **Tools > Internet Options** to open the Internet Options dialog box.
- 2** In the **Privacy** tab, under the **Web Sites** section, click **Edit** to open the Per Site Privacy Actions dialog box.
- 3** In the **Address of Web site** box, enter the name of the Diagnostics Server.
 - ▶ If you entered an IP address when you registered the Diagnostics Server in Mercury Business Availability Center, enter the IP address. If you entered a host name in Mercury Business Availability Center, enter the host name.
 - ▶ Include the `http://` or `https://` prefix, and the port number, as illustrated in the following example:
`http://<Diagnostics_server_host>:2006`
- 4** Click **Allow**. Click **OK** to close the Per Site Privacy Actions dialog box.
- 5** Click **OK** to close the Internet Options dialog box.

11

Installing LoadRunner 8.1 and the LoadRunner Diagnostics AddIn

To enable the Mercury Diagnostics functionality within LoadRunner, you must install the Mercury LoadRunner Diagnostics AddIn. You can only install the LoadRunner Diagnostics AddIn once you have already installed the Diagnostics Server in Commander mode and Mercury LoadRunner 8.1.

This chapter describes:	On page:
Understanding the LoadRunner Diagnostics AddIn	223
Installing Mercury LoadRunner	224
Installing the Mercury LoadRunner Diagnostics AddIn	224
Configuring LoadRunner for Mercury Diagnostics	226

Understanding the LoadRunner Diagnostics AddIn

The LoadRunner Diagnostics AddIn makes it possible for you to access Diagnostics functionality from within LoadRunner. Once the LoadRunner Diagnostics AddIn has been installed, you can configure LoadRunner to connect to the Diagnostics components, use the System Health Monitor to check on the status of the Diagnostics components, and use the Diagnostics components to gather performance metrics during your load tests.

Note: The version of the Diagnostics Server in Commander mode and the version of the LoadRunner Diagnostics AddIn must be exactly the same.

Installing Mercury LoadRunner

If you have not yet installed LoadRunner, you must install it before you can install the Mercury LoadRunner Diagnostics AddIn. Mercury Diagnostics 6.5 can be integrated with LoadRunner 8.1 FP2. For instructions on installing LoadRunner, refer to the *Mercury LoadRunner Installation Guide*.

Installing the Mercury LoadRunner Diagnostics AddIn

The LoadRunner Diagnostics AddIn must be installed on the host machine for the LoadRunner Controller.

To install the LoadRunner Diagnostics AddIn:

Insert the Mercury Diagnostics Collectors CD into a CD-ROM drive. In the **LR_AddIn** directory, double-click **setup.exe** to start the InstallShield Wizard.

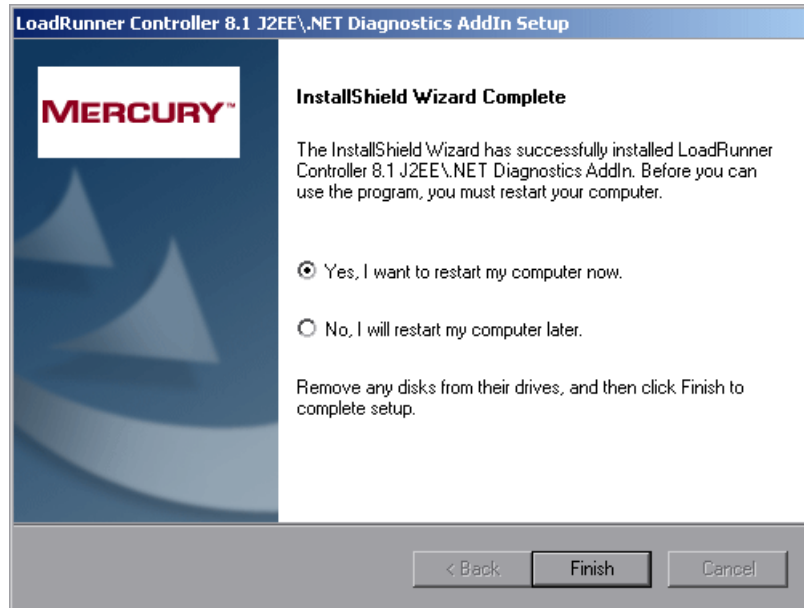
- 1 The software license agreement is displayed. Read the agreement and click **Yes** to accept it.
- 2 The Registration Information dialog box opens.

The screenshot shows a Windows-style dialog box titled "LoadRunner Controller 8.1 J2EE\NET Diagnostics AddIn Setup". The dialog has a "Registration Information" header and the Mercury logo. Below the header, there is a text prompt: "Please type your name, the name of your company and your maintenance number. The maintenance number was provided with your LoadRunner or Add-In package." There are three input fields: "Name:" with "Mercury" entered, "Company:" with "Mercury Interactive" entered, and "Maintenance number:" which is empty. At the bottom of the dialog, there are three buttons: "< Back", "Next >", and "Cancel". The "InstallShield" logo is visible in the bottom left corner.

In the Registration Information dialog box, type your name, the name of your company, and your LoadRunner maintenance number. You can find the maintenance number in the maintenance pack shipped with LoadRunner.

Click **Next** to start the installation process. The installation process begins.

- 3 When the installation process is complete, the installation wizard displays a confirmation message.



To complete the installation process, choose **Yes, I want to restart my computer now** and click **Finish**.

Note: If you are installing the LoadRunner Diagnostics AddIn on a Windows XP machine with service pack 1 and Windows XP Hotfix Q328310 applied, you will receive an Application Error message for **ikernel.exe**. This message is issued because the Windows XP Hotfix Q328310 contains a Win32 API that does not execute as expected by the InstallShield engine. To resolve this problem, see the recommended solutions at the Java Technology Help web site, <http://java.com/en/download/help/ikernel.jsp>.

Configuring LoadRunner for Mercury Diagnostics

Before you can use the Diagnostics features from within LoadRunner you need to configure LoadRunner and provide the necessary information to enable communication with the Diagnostics components. See “Setting Up LoadRunner 8.1 to use Mercury Diagnostics” on page 227 for detailed instructions for configuring LoadRunner.

12

Setting Up LoadRunner 8.1 to use Mercury Diagnostics

This chapter describes how to configure LoadRunner 8.1 to enable Mercury Diagnostics for use in a load test.

This chapter describes:	On page:
About Setting Up LoadRunner to Use Mercury Diagnostics	227
Specifying the Diagnostics Server Details	228
Configuring LoadRunner Scenarios to use Mercury Diagnostics	230

About Setting Up LoadRunner to Use Mercury Diagnostics

LoadRunner and Mercury Diagnostics are integrated products that have been designed to work together to provide information to help you understand and improve the performance of your applications.

Before using Mercury Diagnostics with LoadRunner 8.1, you provide LoadRunner with the information that it needs in order to communicate with the Diagnostics components.

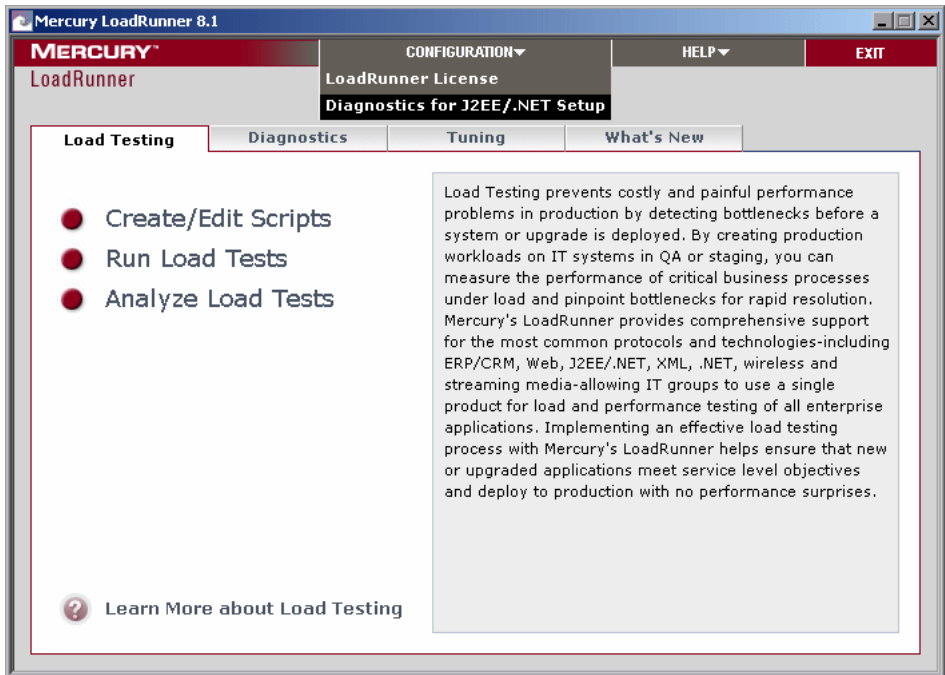
Specifying the Diagnostics Server Details

To make Mercury Diagnostics accessible from LoadRunner, you specify the Diagnostics Server details. You only need to specify these details the first time you use LoadRunner with Mercury Diagnostics.

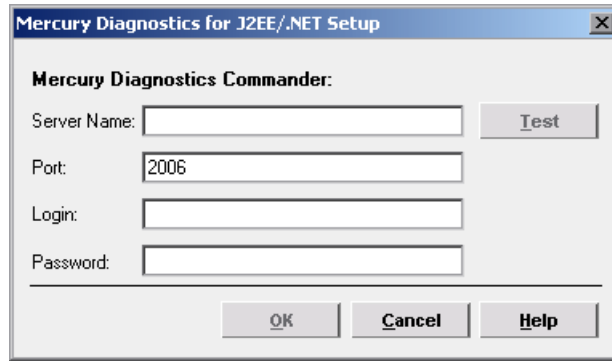
Note: Before specifying the Diagnostics Server details, ensure that the LoadRunner Controller is closed. When the Controller is open, you can view the Diagnostics configuration settings, but you cannot change them.

To specify the Diagnostics Server details in LoadRunner:

- 1 Select **Start > Programs > Mercury LoadRunner > LoadRunner** to open the Mercury LoadRunner launcher window.



- From the Mercury LoadRunner launcher window menu, select **Configuration > Diagnostics for J2EE/.NET Setup** to open the Diagnostics for J2EE/.NET Setup dialog box.



- Enter the details for the Diagnostics Server in Commander mode.
 - ▶ **Server Name.** Enter the name of the machine that is host to the Diagnostics Server in Commander mode.
 - ▶ **Port.** Enter the port number used by the Diagnostics Server in Commander mode. The default port number is 2006.

Note: LoadRunner does not support communication with the Diagnostics Server in Commander mode using HTTPS.

- ▶ **Login.** Enter the user name with which you log on to Mercury Diagnostics. The default username is `admin`.
- ▶ **Password.** Enter the password with which you log on to Mercury Diagnostics. The default password is `admin`.

Note: The user name that you specify should have **view**, **change** and **execute** privileges. For more information about user privileges, see “Understanding User Privileges” on page 491.

- 4 Click **Test** to verify that you entered the correct information for the Diagnostics Server in Commander mode and that there is connectivity between the Diagnostics Server in Commander mode and LoadRunner.
- 5 Click **OK** to complete the configuration process.

Configuring LoadRunner Scenarios to use Mercury Diagnostics

You configure your scenario for Diagnostics from the LoadRunner Controller for each load test scenario when you want to collect Diagnostics data.

To capture Diagnostics metrics in a load test scenario:

- 1 Configure the Diagnostics parameters for the scenario
- 2 Select the Probes that will be included in the scenario.

For detailed information about configuring LoadRunner scenarios to use Mercury Diagnostics, refer to the *Mercury Diagnostics User's Guide*.

13

Setting Up Performance Center 8.1 to Use Mercury Diagnostics

This chapter describes how to configure Performance Center 8.1 to enable Mercury Diagnostics for use in a load test.

This chapter describes:	On page:
About Setting Up Performance Center to Use Mercury Diagnostics	231
Specifying the Diagnostics Server Details	232
Configuring Performance Center Load Tests to Use Mercury Diagnostics	234

About Setting Up Performance Center to Use Mercury Diagnostics

Performance Center and Mercury Diagnostics are integrated products that have been designed to work together to provide information to help you understand and improve the performance of your applications.

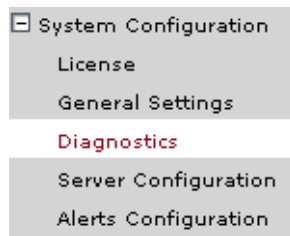
Before you can use Mercury Diagnostics with Performance Center 8.1, you provide Performance Center with the information that it needs in order to communicate with the Diagnostics components.

Specifying the Diagnostics Server Details

To make Mercury Diagnostics accessible from Performance Center, you specify the Diagnostics Server details. You only need to specify these details the first time you use Performance Center with Mercury Diagnostics. You provide this information on the Diagnostics page of the Performance Center Administration Site.

To specify the Diagnostics Server details in Performance Center:

- 1 Log on to the Performance Center Administration Site.
- 2 From the left navigation menu, choose **System Configuration > Diagnostics** to open the Diagnostics page.



- 3 In the **Setting J2EE/.NET Environment** section of the Diagnostics page, enter the details for the Diagnostics Server in Commander mode.

A screenshot of a web form titled 'Setting J2EE/.Net Environment'. The form contains the following fields and controls:

- J2EE/.Net Diagnostics Server Name**
- Server Name:
- Port Number:
- Login Name:
- Password:
- Use HTTPS:
- Save button

- ▶ **Server Name.** Enter the name of the machine that is host to the Diagnostics Server in Commander mode.
- ▶ **Port Number.** Enter the port number used by the Diagnostics Server in Commander mode. The default port number is 2006.
- ▶ **Login Name.** Enter the user name with which you log on to Mercury Diagnostics. The default user name is **admin**.
- ▶ **Password.** Enter the password with which you log on to Mercury Diagnostics. The default password is **admin**.

Note: The user name that you specify should have **view**, **change** and **execute** privileges. For more information about user privileges, see “Understanding User Privileges” on page 491.

- ▶ If Performance Center is communicating with Diagnostics through HTTPS, select the **HTTPS** box.

Note: If you select HTTPS as your communication protocol, additional configuration steps are required. For more information about the steps required, see Chapter 22, “Enabling HTTPS Between Diagnostics Components.”

- 4 Click **Save** to complete the configuration process.

Note: If you want to integrate Performance Center with a different Diagnostics Server in Commander mode, you need to update the above information.

Configuring Performance Center Load Tests to Use Mercury Diagnostics

You configure your load test for Diagnostics from the Performance Center User Site for each load test where you want to capture Diagnostics metrics.

To capture Diagnostics metrics in a load test:

- 1** Configure the Diagnostics parameters for the load test.
- 2** Select the Probes that will be included in the load test.

You provide this information on the Load Test Configuration page of the Performance Center User Site.

For detailed information about configuring Performance Center load tests to use Mercury Diagnostics, refer to the *Mercury Diagnostics User's Guide*.

Part V

Diagnostics Instrumentation

14

Instrumenting an Application

This chapter describes how to control the instrumentation that Mercury Diagnostics applies to the classes and methods of your applications to enable it to gather the performance metrics.

This chapter describes:	On page:
About Instrumentation and Capture Points Files	238
Locating the Capture Points Files	239
Coding Points in the Capture Points File	241
Defining Points With Code Snippets	247
Instrumentation Examples	257
Understanding the Overhead of Custom Instrumentation	266
Instrumentation Control	267
Advanced Instrumentation	266

About Instrumentation and Capture Points Files

Instrumentation refers to bytecode that the Probe inserts into the class files of your application as they are loaded by the class loader of your virtual machine. Instrumentation enables a Probe to measure execution time, count invocations, retrieve arguments, catch exceptions and correlate method calls and threads. The instrumentation definitions for each Probe instance are specified in the capture points file.

The capture points file is a robust instrumentation control mechanism that allows you to control the scope of the instrumentation so that Diagnostics can give you all of the information that you need to understand the performance of your applications without flooding you with costly or confusing extraneous information. The instrumentation definitions contained in the capture points file are instructions sets called *points* that tell the Probe which methods to instrument, how they should be instrumented, and which collection instrumentation should be installed.

Points can include regular expressions that "wild card" the instructions so that they apply to more than one method, class, and package or namespace specification. For more information about using regular expressions, see Appendix I, "Using Regular Expressions."

You can customize the points in the capture point file to include methods, classes, packages and namespaces for areas of the application that do not fall within the default points. A common situation that may require custom points is when a J2EE application contains business logic that is not derived from the `javax.ejb.SessionBean` interface. Another situation for custom points is when you want to override a default point to alter its layer or to track it from a specific caller method.

The Microsoft specification for .NET does not include a unified or recommended interface that business logic should implement. This means that the .NET Probe will always require custom points in the capture points file in order to enable it to gather meaningful metrics for the performance of the business logic classes and methods in .NET applications.

The points in the capture points file are grouped into layers. Layers are used to organize the performance metrics into meaningful tiers of information that can be compared as part of the Diagnostics triage process and to control the collection behavior of the instrumentation.

The points in the capture points file that is installed with the probe are grouped into default layers. You can customize the default layers and create new layers. For more information about layers see Chapter 16, “Instrumentation Layers.”

Locating the Capture Points Files

When you install the Probe, a predefined default capture points file is installed with a set of points for the platform you are using.

.NET Capture Points Files

For .NET, a default capture points file for the .NET platform is located at `<probe_install_dir>\etc\aspnet.points`.

For .NET, the Probe installer creates a separate capture points file for each Application Domain that it detects. These capture points files are located at `<probe_install_dir>\etc\<ApplicationDomain>.points` file.

These points files are read along with the default ASPNET.points file by the .NET probe.

Java Capture Points Files

For Java, a default J2EE defined points is located at:

`<probe_install_dir>\etc\auto_detect.points`

Note: You should you create a copy of the default `auto_detect.points` file and use the copy to make all of your instrumentation customizations. This is a precaution to prevent you from losing your custom instrumentation when you upgrade to a new version of the Probe and the installer overlays the `auto_detect.points` file.

To override the default file name so that the copy will be used instead use the `-Dprobe.points.file.name=<newPointsFileName_NoExtension>` JVM property. The default file name is specified in `<probe_install_dir>\etc\probe.properties`.

Coding Points in the Capture Points File

The following is a list of the arguments that can be used to define a point in the capture points file:

```
[Point-Name]    =<unique name for the point>
;-----
class           = <class/package name/s to capture>
method         = <method name/s to capture>
signature      = <signature/s of method/s>
ignore_cl      = <classes to ignore>
ignore_method  = <method prototypes to ignore>
deep_mode      = <soft or hard mode>
scope          = <comma separated list of methods>
ignoreScope    = <comma separated list of methods>
detail         = <args,caller,return>
code-key       = <key that you generate to secure the code snippet>
tierDefinition = <hex_identifier>
layer          = <layer name>
layerType      = <layer type>
merge_classes  = <class names>
merge_url_to   = <comma separated list of methods>
ignore_tree    = args,caller,return
active         = <true, false>
```

The arguments are described in the following sections.

Mandatory Point Arguments

Every point, except for the points for LWMD, RMI and SAP RFC, HttpCorrelation, and JDBC SQL, must contain the following arguments:

Argument	Description
Point-Name	A unique name for the point.
class	The class argument specifies the name of the class or interface to be instrumented. The name should include the full package/namespace name using periods between the package levels. Any valid regular expression may be used.
method	The method argument specifies the name of the method to be instrumented. To be successful, the method name must match a method defined in the class or interface specified by the class argument. Any valid regular expression may be used.
signature	The signature argument specifies the signature (parameter and result types) of the method using javap symbolic encoding for method signatures (<jdk_install>/bin.javap -s). The only valid regular expression is !.* which includes all signatures.

Argument	Description
layer	<p>The layer argument specifies a layer, sub-layer or tier under which the data from this point is grouped. Layers are a part of the instrumentation collection control.</p> <p>Layers in a point can be specified with nested layers or sub layers by separating the layer names with a / (slash). The layer specified following the slash is a sub layer of the layer specified before the slash. A sub-layer can be shown to have its own sub-layers by coding another slash and layer name following a sub-layer name.</p> <p>In the UI, the sub-layers for a layer are shown under their parent layer. For example, if the sub-layers of the Web layer are JSP and Struts, they will be shown under the Web layer and a drilldown will exist from Web to JSP and Struts.</p>
active	<p>The active attribute activates or deactivates a point. When true the point is activated. When false the point is inactive and is ignored by the probe.</p>

The following is an example of a custom point that contains the mandatory arguments:

```
[MyCustomEntry_1
; comments here....
class = myPackage.myClass.MyFoo
method = myMethod
signature = !.*
layer = myCustomStuff
```

Note: Regular expressions may be used for most of the arguments in a point. They must be prefaced with an exclamation point. For more information about using regular expressions, see Appendix I, “Using Regular Expressions.”

Optional Point Entries

Point definition may contain one or more of the following arguments:

Argument	Description
ignore_cl	The ignore_cl argument is used to specify a comma-separated list of classes to ignore.
ignore_method	The ignore_method is used to specify a comma-separated list of methods to ignore. Any class matching one of the classes specified with ignore_method is not instrumented.
deep_mode	<p>The deep_mode argument specifies how subclasses are handled. This argument accepts the following three values:</p> <ul style="list-style-type: none"> ▶ none - A value of "none" is like not specifying a deep_mode argument. It has no effect on how subclasses are handled. ▶ soft - A value of "soft" requests that for every class or interface matching the class, method, and signature entries, any subclasses or sub-interfaces that also implement the matching method and signature should also be instrumented. ▶ hard - A value of "hard" requests that for every class or interface matching the class, method, and signature entries, any subclasses or sub-interfaces at any depth should have all their methods instrumented. Hard mode is typically used for points for interfaces. Caution: "hard" mode can lead to extensive instrumentation and very high Probe overhead.
scope	The scope argument constrains the context in which instrumentation is performed. If this entry is specified, the inserted bytecode will be caller side. Any valid regular expression may be used for the value of this argument. Scope values are a comma-separated list of package, class, and method names in standard Java notation.
ignoreScope	The ignoreScope argument is used to exclude certain packages, classes, and methods from those included in the scope specified in scope argument.

Argument	Description
detail	<p>The detail argument specifies more specific capture instructions. It is a comma-separated list of the following:</p> <ul style="list-style-type: none"> ▶ caller - causes caller side instrumentation to be performed. If this keyword is not specified, the default instrumentation, callee side instrumentation, is performed. ▶ args:n - calls the <code>toString()</code> method of the <i>n</i>-th argument. The string that is returned is displayed in the method's argument field in the Diagnostics console. The captured string can be used as the aggregation parameter in the layer argument. The value for <i>n</i> can be 1 through 256. ▶ args:0 - calls the <code>toString()</code> on the current class instance or callee object. Static methods return the class name of the callee object ▶ args:code - inserts the code-snippet specified in the code argument of the point into the bytecode for methods that comply with the point. The final string value on the stack when the code-snippet has executed is displayed in the method's argument field in the Diagnostics console and may also be used as the aggregation parameter in the layer argument. ▶ disabled - prevents the instrumentation inserted into the bytecode from reporting data. A disabled point can be dynamically enabled using the Instrumentation control webpage so that it will begin reporting data. This webpage can be accessed using the Profiler URL <a href="http://<probe_install_dir>:<probe_port>/inst/layer">http://<probe_install_dir>:<probe_port>/inst/layer. ▶ outbound - flags the method so it is listed on the "Outbound Calls" screen. Also, causes the Diagnostics argument for this instrumentation entry to be parsed to determine if additional information about the outbound request can be displayed in the Diagnostics dashboards.

Argument	Description
detail (continued)	<p>► ws-operation - specifies that the instrumentation entry is for an inbound Web services call. Also, causes the Diagnostics argument for this instrumentation entry to be parsed to determine if additional information about the Web service request can be displayed in the Diagnostics dashboards.</p>
code-key	<p>The code-key argument specifies the secure code key that you generated for the code snippet that you created for the points code argument. Information about code snippets can be found at Defining Points With Code Snippets and information on code keys can be found at “Securing Code Snippets” on page 254.</p>
layerType	<p>Points with this argument result in a special layer breakdown for portal components. All necessary instances of these are pre-configured in the auto-detect.points file, and it is important that they are not removed.</p> <p>Other values for layerType are not supported at this time</p>

Defining Points With Code Snippets

Custom code arguments specify a snippet of code that is to be inserted into the bytecode for a point. Code snippets in a point are used when the value returned by calling an object's `toString()` method, as specified in the `args:n` argument, is not going to provide useful information for the Diagnostics console or when there is a requirement to display more than one argument for an instrumented method.

A code snippet in a point is declared using the key word `args:code` in the detail argument of the point. The code snippet is typically secured using a code-key argument to prevent un-authorized modifications of the code snippet. The values for the code-key arguments can be generated using any running Probe instance's code-key generator page and are valid on any probe installation. For more information on the code-key see "Securing Code Snippets" on page 254.

The actual code snippets for a point are entered into the `<probe_install_dir>/etc/code/custom_code.properties` file. The code snippets in this file are associated with the point in the capture points file using the value of the code-key. Code Snippets are created using pseudo java code that uses syntax similar to OGNL. Using code snippets, calls can be made from the instrumented byte code to methods that can be accessed by the instrumented method. Objects returned by code snippets can be cast and can have their methods executed as well. Code Snippets must end with a string or an object where `toString()` can be left on the "stack" of statements being parsed into bytecode. This final string of the Code Snippet is used for the returned argument value this is displayed in the Diagnostics console.

Note: Code snippets are a very powerful tool that should be used carefully because of the potential impact to the overhead incurred by the Probe. For this reason, Diagnostics requires that a code-key be specified along with the code snippet before the probe will use the code snippet during instrumentation.

Using Code Snippets

To use code snippets when specifying a point in `<probe_install_dir>/etc/auto_detect.points` you must use the detail and code-key arguments as shown in the following example.

```
class    = javax.jms.TopicPublisher
method  = publish
signature = !\(\Ljavax/jms/Topic.*
deep_mode = soft
layer   = Messaging/JMS/Producer
detail  = outbound,no-correlation,args:code
code-key = 5d741d0f
```

The `args:code` entry in the `detail` argument indicates that a code snippet has been entered for the point. The `code-key` value is used to secure the code that is in the code snippet and to tie the point with the actual code snippet.

The code snippet associated with the point must be entered in `<probe_install_dir>/etc/code/custom_code.properties` as shown in the following example:

```
# Used by [JMS-TopicPublisher2]
5d741d0f = "DIAG_ARG:type=jms&name=topic:" + #arg1.getTopicName() + "\n\
"&target=" + #arg1.getTopicName();
```

The code snippet is associated with the point in the capture points file using the value of the code-key.

Code Snippet Grammar

The following describes the syntax that can be used to create the code snippets:

► Literals

Only the following literal types are supported in code snippets.

Literal Type	Syntax Example
string	"a string"
boolean	true, false
integer	42
null constant	null

► String concatenation

Basic string concatenation is supported in code snippets.

Concatenation Type	Syntax Example
Two strings	"a string" + "another string"
A string and a literal	"a string" + 42

► Local members

Default local members provide a way for code snippets to reference the current instance or objects that were passed to the instrumented method. These local members are used to call methods or retrieve values from those references:

Variable	Use
#callee	References the callee object for an instance method. Equivalent to the java "this" reference. This variable may not be used when referencing a static method.
#arg1, #arg2, ..., #argN	References the arguments for the callee method call
#classloader	Reserved for Mercury internal use.

Note: Some instrumentation points support "special" variable references. For example, the `CLApplicationDiscoveryPoint` supports a `#classloader` variable.

► **Class references and static members**

Static members/methods may be accessed by prepending the class with an `@` symbol to identify it as a Static and then marking the method being accessed with an `@` symbol as in the examples below:

```
@java.lang.System@.out ("Hello World");
```

```
@com.mercury.diagnostics.capture.metrics.countingCollector@.incrementCounter  
();
```

The arguments in the code snippets support Java class syntax when the Java class is surrounded with a marker that the parser can get hold of.

The following examples show how to use the `@` symbol as a marker:

```
@java.lang.System@
```

```
@java.lang.System@out (Static field)
```

► **Spanning multiple lines with the stack of method calls**

The stack of method calls in a code snippet can span multiple lines. The parser which builds the bytecode requires a `"\"` (back slash) before each carriage return when it is supposed continue parsing the stack of statements. The final line of the Code Snippet stack of statements should not contain a `"\"` (back slash) and should simply end with carriage return.

```
@java.lang.System@.out ("Hello World");\  
"Callee Name="+#callee.getName().toString();
```


► Casting

When calling a method that returns an object, casting is typically required to call members on the returned object. Casting is supported on object references. To cast an object to another type, place the casting reference between the symbols "<" and ">" following the reference to that object. The following are examples of casting.

```
#arg1<com.myCompany.myFoo>.myMethod();
```

This is equivalent to the Java statement:

```
((com.myCompany.myFoo)arg1).myMethod();
```

```
@some.class.Foo@foo<com.myCompany.myFoo>.myMethod();
```

Would be equivalent to the java statement:

```
((com.MyCompany.myFoo)some.class.Foo.foo).doSomething();
```

```
#foo = #arg1<bar>.b(); #foo.toString();
```

Creates the following java equivalent:

```
String foo = ((Bar)arg1).b(); ((Object)foo).toString();
```

Note: Casting is not supported for special types such as #classloader.

► **Method calls**

Method calls can be included in snippet arguments. The support of method calls includes calls with or without arguments and method chaining. The following are examples of method calls that have been included in code snippet arguments:

```
#arg1.toString()
```

```
#arg2.getSomething().getSomethingElse()
```

```
#callee.getSomething("foo", #arg1).somethingElse()
```

```
@some.Class@.staticMethod()
```

The dot still needs to appear after the static reference for the method call to be parsed properly.

```
@java.lang.System@out.println("Here I am!")
```

To speed up the generation of bytecode at runtime (by avoiding reflection), you can specify the type that is returned from a method as shown in the following example:

```
#arg1.getSomething()<some.class.Here>
```

This will not help if the method takes arguments, or if a static field is used.

► **Multiple statements**

Code snippets can include multiple statements in a single code snippet. This is necessary for instrumentation, such as `CLApplicationDiscoveryPoint`, that expect multiple objects to be left on the stack and it can be handy in other situations.

```
@java.lang.System@out.println("Look out!");  
#arg2.getSomething();
```

► Local Member assignment

In addition to the default supported "local" variables, you can create your own local members to hold object references returned by called methods.

To create a new Local Member enter the "#" symbol before the name of the local member and the parser creates the local member for you.

```
#myBar = #arg2.getName();\
#myUpperBar = #myBar.toUpper();\
"Target Name=http://" + myUpperBar + "/services";
```

► Conditional Logic

Code snippets syntax allows for limited conditional logic which is equivalent to the java if-else statement. This syntax allows you to compare object references of the same type using both the == and != operators. Literal value and primitive comparisons are not valid using this syntax.

The following is an example of how to compare references:

```
(value1 == value2 ? <if_True_codeSnippet>:<if_False_codeSnippet>)
```

The following is an example of how to verify that an object is not null before calling a method:

```
(#arg1 == null ? "Unknown" : #arg1.getSomething())
```

This would be equivalent to the following java statement:

```
if (arg1==null) return "Unknown" else return arg1.myMethod();
```

Securing Code Snippets

By default, you must specify a valid code-key along with the code snippet before the probe will use the code snippet during instrumentation. Requiring the code-key prevents someone from accidentally introducing instrumentation that could significantly increase the overhead of the probe.

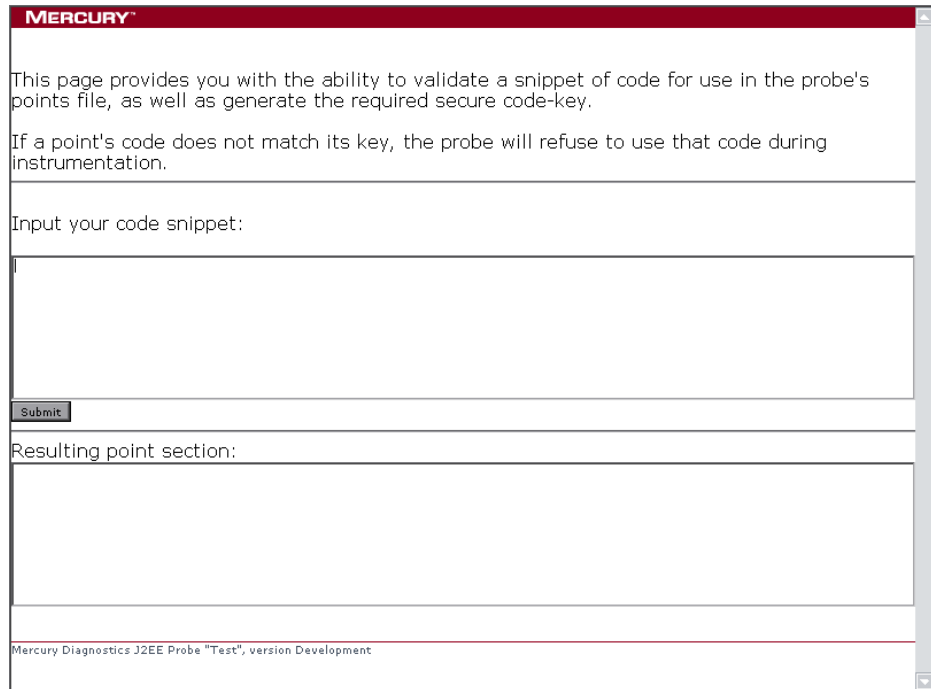
When you generate the code-key, Diagnostics checks the syntax of the code snippet to make sure that it is valid before it generates the key. When Diagnostics instruments your application, it checks the value entered for the code-key argument to make sure that it matches the code-key that it calculates for the code snippet for the point. If the code-keys do not match, Diagnostics ignores the code snippet and does not create the instrumentation point.

Generating the Code Snippet Code-Key

The J2EE Probe is installed with a tool that generates the code-key from the code snippet that you input.

To generate a code-key:

- 1 Open the page at the following URL in your browser: <http://<probe-host>:<probe-port>/inst/code-key>. Diagnostics displays the page where you can validate the code snippet syntax and generate the code-key as shown in the following example:



The screenshot shows a web browser window titled "MERCURY". The page content includes:

- A header bar with the "MERCURY" logo.
- Text: "This page provides you with the ability to validate a snippet of code for use in the probe's points file, as well as generate the required secure code-key."
- Text: "If a point's code does not match its key, the probe will refuse to use that code during instrumentation."
- A section titled "Input your code snippet:" followed by a large text input field.
- A "Submit" button.
- A section titled "Resulting point section:" followed by a large text area for the output.
- A footer: "Mercury Diagnostics J2EE Probe 'Test', version Development".

- 2 Enter the code snippet that you specified in the code argument in the **auto_detect.points** file into the **Input your code snippet** text box and click **Submit**

Note: The code snippet must include all of the text following the code = argument name.

- 3 Diagnostics presents the results of the code snippet validation and the code-key generation in the **Resulting point section** text box.

If the code snippet was valid, Diagnostics displays the value of both the code-key and code arguments. Enter these values into the points file.

If the code snippet was not valid, Diagnostics displays an error message that indicates the problem that was detected. Correct the problem and click **Submit** again to validate the corrected code.

Disabling the Code-Key Security Check

By default, Diagnostics verifies that the value of the code-key argument matches the value that it generates when it is instrumenting the application. It is possible to disable this security check by inserting the **require.code.security.key** property into the `<probe_install_dir>/etc/inst.properties` file with a value of `false`.

Note: You should be very careful when using this property. If you disable this check, you could allow unexpected processing overhead and unpredictable performance monitoring results.

Instrumentation Examples

The examples in this section have been provided to illustrate how you can customize the instrumentation of your application by creating and modifying the points in the capture points file.

General Examples

Custom layer and sub-layer

- The point in the following example creates a custom sublayer called "BAR" within the layer called "FOO" for the method myMethod in myCompany.myFoo class:

```
[myCompany.myFoo_customLayer]
class = myCompany.myFoo
method = myMethod
signature = !.*
layer = FOO/BAR
```

Wild-card method

- The point in the following example captures all methods in the MyCompany.MyFoo class:

```
[myCompany.myFoo_AllMethods]
class = myCompany.myFoo
method = !.*
signature = !.*
layer = FOO/BAR
```

Ignore Specified Methods

- ▶ The point in the following example captures all methods in the `MyCompany.MyFoo` class except for the methods `setHomeInterface` and `getHomeInterface`:

```
[myCompany.myFoo_AllMethodsExcept]
class = myCompany.myFoo
method = !.*
signature = !.*
ignoreMethod = !setHomeInterface.*, !getHomeInterface.*
layer = FOO/BAR
```

- ▶ The point in the following example captures all methods in the `MyCompany` package/namespace except for those contained in the `MyCompany.logging` class:

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany\..*
method = !.*
signature = !.*
ignore_cl = MyCompany.logging
layer = FOO/BAR
```

Capture Only a Specific Method In a Class

- ▶ The point in the following example captures all methods in the constructor for the `MyCompany.MyFoo` class:

```
[myCompany.myFoo_Constructor]
class = myCompany.myFoo
method = <init>
signature = !.*
layer = FOO/BAR
```


- ▶ The point in the following example captures all methods in the singleton constructor for the MyCompany.MyFoo class:

```
[myCompany.myFoo_Singleton]
class = myCompany.myFoo
method = <clinit>
signature = !.*
layer = FOO/BAR
```

- ▶ The point in the following example captures the setFoo method in the MyCompany.MyFoo class:

```
[myCompany.myFoo_setFoo]
class = myCompany.myFoo
method = setFoo
signature = !.*
layer = FOO/BAR
```

- ▶ The point in the following example captures all "set" methods in the MyCompany.MyFoo class:

```
[myCompany.myFoo_AllSets]
class = myCompany.myFoo
method = !set.*
signature = !.*
layer = FOO/BAR
```

- ▶ The point in the following example captures all methods in the MyCompany package/namespace:

```
[myCompany_All_Methods]
class = !myCompany.*
method = !.*
signature = !.*
layer = FOO/BAR
```

Capture a Specific Method That Returns a String

- The point in the following example captures the `getFoo` method that returns a `java.lang.String` in the `MyCompany.MyFoo` class:

```
[myCompany.myFoo_GetFoo_String]
class = myCompany.myFoo
method = getFoo
signature = ()Ljava/lang/String
layer = FOO/BAR
```

Capture with a Controlled Scope

- The point in the following example captures all methods in the `MyCompany` package/namespace that are called from the `MyCompany.logging` class. For more details see “Using Caller Side Instrumentation” on page 268:

```
[myCompany_All_Methods_except_from_MyCompany_Logging]
class = !myCompany.*
method = !.*
signature = !.*
scope = MyCompany.logging
layer = FOO/BAR
```

Deep_mode Examples

The following interface definition will be used for both soft and hard `deep_mode` examples:

```
public interface Interface1 {

    public void callerMethod();

}
```

The following class will be used for both soft and hard deep_mode examples:

```
public class Class1 implements Interface1 {
    public void callerMethod(){
        calleeMethod();
        calleeMethod2();
    }

    public void calleeMethod(){
        System.out.println("hello world");
        //more code lines here...
    }

    public void calleeMethod2(){
        System.out.println("hello world 2");
    }
}
```

- The point in the following example captures the "callerMethod" in the Class1 class:

```
[Training-1]
class    = Interface1
method  = !.*
signature = !.*
deep_mode = soft
layer   = Training
```

- The point in the following example captures all methods in Class 1 (i.e. "callerMethod", "calleeMethod1" and "calleeMethod2").:

```
[Training-1]
class    = Interface1
method  = !.*
signature = !.*
deep_mode = hard
layer   = Training
```

Argument Capture Examples

- ▶ The point in the following example captures the first argument to the setFoo method in the MyCompany package/namespace:

```
[myCompany_setFoo_Method_first_argument]
class = !myCompany\..*
method = !.*
signature = !.*
detail = args:1
layer = FOO/BAR
```

- ▶ The point in the following example uses the second argument to the setFoo method in the MyCompany package/namespace as the sublayer name.

```
[myCompany_setFoo_Method_second_argument_as_layer]
class = !myCompany\..*
method = !.*
signature = !.*
detail = args:2
layer = FOO/{ARG}
```

Code Snippet Examples

- ▶ The point in the following example returns the String "HelloWorld" to the Diagnostics Server for the arguments passed to the setBar method.

Note: Notice the extra double quote before the "Hello World" string in the code section. The double quote is required by the parser in some versions of the Probe to indicate the first statement on the "stack" is a string and is not a local member. If the extra double quote is required, do not include it in code-key generation steps:

The point in the capture points file is entered as follows:

```
class = com.myCompany.myFoo
method = setBar
signature = !.*
detail = args:code
code-key = 20e704af
layer = FOO
```

The code snippet in the custom_code.properties file is entered as follows:

```
20e704af = "HelloWorld"
```

- The point in the following example calls the static incrementCounter method in the com.mercury.diagnostics.capture.metrics class. This method returns a string that includes the string value for the first object passed to the 'lookup' method of the javax.naming.Context class to the Diagnostics console as the method's arguments.

```
[JNDI-lookup]
;----- Server side JNDI hook -----
class    = javax.naming.Context
method   = lookup
signature = (Ljava/lang/String;)Ljava/lang/Object;
detail = args:code
code-key = ec79232c
deep_mode = soft
layer    = Directory Service/JNDI
```

The code snippet in the custom_code.properties file is entered as follows:

```
ec79232c =
@com.mercury.diagnostics.capture.metrics.countingCollector@.incrementCounter("jndi");" + #arg1;
```

- ▶ The point in the following example calls the static `incrementCounter` method in the `com.mercury.diagnostics.capture.metrics` class. It does not return a value to the Diagnostics Console for the method's arguments.

Note: This point is coded so that it is disabled by default. This means that it does not report data to the Diagnostics Console, but always calls the "incrementCounter" method.

```
class = com.myCompany.myFoo
method = doBar
signature = !.*
detail = disabled,args:code
code-key = 55aa1f8a
layer = FOO
```

The code snippet in the `custom_code.properties` file is entered as follows:

```
55aa1f8a =
@com.mercury.diagnostics.capture.metrics.countingCollector@.incrementCounter();""
```

- ▶ When the detail argument in a point contains the "outbound" or "ws-operation" keyword, Diagnostics attempts to parse the final string on the Code Snippet stack for additional information to display about the method call.

```
[Apache_WS_WebService_Invoke]
class = org.apache.axis.client.Call
method = invoke
signature = (Ljava/lang/String;Ljava/lang/String;Ljava/lang/Object;)Ljava/lang/Object;
detail = outbound,args:code
code-key = 333c286d
layer = Web Services
```

The code snippet in the `custom_code.properties` file is entered as follows:

```
333c286d = "DIAG_ARG:type=ws&ws_name=" + \n\  
#callee.getTargetEndpointAddress() + \n\  
&ws_op="#" + callee.getOperationName().getLocalPart():
```

The point in the previous example creates a string similar to the following example:

```
"DIAG_ARG:type=ws&ws_name=http://192.0.0.1&ws_op=myService"
```

The following are the most common available argument strings that Diagnostics parses for additional information to display about the method call.

For CICS

```
"DIAG_ARG:type=cics&name=<Descriptive Text>"
```

For WebService (**ws_op**, **target** and **ws_ns** are optional entries)

```
"DIAG_ARG:type=ws&ws_name=<WebServiceName>+"&ws_op=<OperationName>&t  
arget=<TargetName>&ws_ns=<TargetNameSpace>
```

For JMS

```
"DIAG_ARG:type=jms&name=<Descriptive Text>"
```

Understanding the Overhead of Custom Instrumentation

When you are creating custom instrumentation you should beware of over-instrumenting the application because it can introduce excessive latency into the probed application. The excessive latency arises from an increase in the classloader latency as more and more classes are instrumented. The custom instrumentation does not have the same impact on the method latency or the CPU overhead because the overhead of instrumentation is nearly fixed for every method because the amount of bytecode is almost always the same. This means that the physical percentages of the CPU and latency overhead will vary in direct proportion to the length of time the method takes to execute.

For example if a method take 100ms and instrumentation makes it execute in 101ms, then overhead is 1%. If a method takes 10ms and instrumentation changes it's response to 11ms, then overhead is 10%. If this method is not called very often its overall latency effect on the application is minimal. However, the overall latency effect of an instrumented method that is called more frequently could have an impact on the latency of the application's response even though is overhead percentage is much smaller.

Unlike a traditional profiler, Mercury Diagnostics uses bytecode instrumentation. This allows the default instrumentation to be selective so as to minimize the overhead caused by instrumentation to an average of 3-5%. Methods with higher latency overhead introduced by instrumentation are only instrumented when they are called infrequently in relation to other components in the application and when the instrumentation provides specific information needed for triage activities (i.e. JNDI lookups)

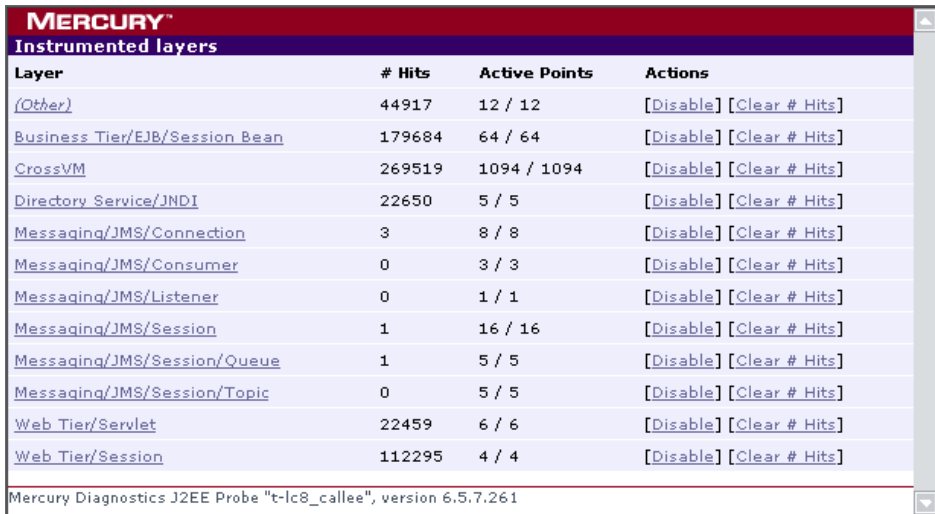
You should also consider Diagnostics data overhead when you are customizing the instrumentation for your application. The more methods that you instrument the more data that the Probe must serialize and pass over the network to the Diagnostics Server. You can tune the Probe's default configuration so that it can adjust the volume of Diagnostics data to avoid any unnecessary effect on the performance of the system being monitored. Please note that improper probe tuning can cause CPU, Memory and Network overhead on the physical machine where your probe resides. See Chapter 21, "Advanced J2EE Probe and Application Server Configuration," for more information about managing Latency, CPU, Memory and Network overhead.

Instrumentation Control

By default, the layers defined in the capture points file are enabled. If you include the `details=disabled` argument in a point the layer is disabled when the Probe is started.

The classmap in JDK 1.5 provides the capability to dynamically instrument methods and classes using the JVMTI interface without restarting the JVM instance. All other virtual machines require that the JVM instance be restarted in order to apply changes that you make to the capture points files. Once instrumentation is placed within a method, its data collection and running CPU and method latency overhead can be controlled on a per layer basis using the Probe's Instrumented Layers page.

You can access the Instrumented Layers page from the URL: <http://<probe-host>:<probe-port>/inst/layer>.



Layer	# Hits	Active Points	Actions
<i>(Other)</i>	44917	12 / 12	[Disable] [Clear # Hits]
Business_Tier/EJB/Session_Bean	179684	64 / 64	[Disable] [Clear # Hits]
CrossVM	269519	1094 / 1094	[Disable] [Clear # Hits]
Directory_Service/JNDI	22650	5 / 5	[Disable] [Clear # Hits]
Messaging/JMS/Connection	3	8 / 8	[Disable] [Clear # Hits]
Messaging/JMS/Consumer	0	3 / 3	[Disable] [Clear # Hits]
Messaging/JMS/Listener	0	1 / 1	[Disable] [Clear # Hits]
Messaging/JMS/Session	1	16 / 16	[Disable] [Clear # Hits]
Messaging/JMS/Session/Queue	1	5 / 5	[Disable] [Clear # Hits]
Messaging/JMS/Session/Topic	0	5 / 5	[Disable] [Clear # Hits]
Web_Tier/Servlet	22459	6 / 6	[Disable] [Clear # Hits]
Web_Tier/Session	112295	4 / 4	[Disable] [Clear # Hits]

Mercury Diagnostics J2EE Probe "t-lc8_callee", version 6.5.7.261

To disable a layer from the Instrumented Layers page, click the **Disable** link associated with the layer on the page. The layer is disabled and the link toggles to **Enabled** so that you can enable the layer again when necessary.

Advanced Instrumentation

Using Caller Side Instrumentation

By default all instrumentation in Diagnostics is Callee side instrumentation where the bytecode is placed within the method call. Caller side instrumentation refers to the process of placing the bytecode for measurement around the call to the method to be instrumented instead of within.

Caller side instrumentation allows finer control of instrumentation placement, but may increase application classloader time as each class specified in the scope must be checked for references to the class/method that is specified in the points.

A common use for caller side instrumentation is to instrument calls to methods in `rt.jar`. Classes loaded into the virtual machine using the bootclassloader and not from a conventional class loader cannot be directly instrumented. To instrument calls to these methods you must use caller side instrumentation.

In the point in the following example, the "parse" methods for the `javax.xml.parsers.SAXParser` and `javax.xml.parsers.DocumentBuilder` is instrumented by placing bytecode around the calls to "parse" in any (!.*) method from any class. Caller side instrumentation is required because both the `javax.xml.parsers.SAXParser` and `javax.xml.parsers.DocumentBuilder` classes are contained in the `rt.jar` and loaded into the virtual machine by the bootclassloader.

```
[XML-DOM-JDK14]
;----- Interface -----
Class = !javax.xml.parsers\.(SAXParser|DocumentBuilder)
method  = parse
signature = !.*
scope = !.*
layer  = XML
```

- The point in the following example instruments calls to `javax.naming.Context`'s "lookup" method that are called from the `com.myCompany.myFoo` classes and places them in the JNDI sublayer in the FOO layer.

```
[JNDI-lookup-FOO]
;----- Server side JNDI hook -----
class  = javax.naming.Context
method  = lookup
signature = (Ljava/lang/String;)Ljava/lang/Object;
detail  = args:1
scope = !com\myCompany\myFoo.*
deep_mode = soft
layer  = FOO/JNDI
```

Note: This instrumentation should be placed above any other points that may be in conflict.

To verify that the point has caused the bytecode to be properly placed, check the `<probeInstall>/log/<probeName>/detailReport.txt` file for the entries Unique Header Name (i.e. `[JNDI-lookup-FOO]`)

- During final triage steps for a performance issue in an application it may be impractical to use the classmap and individually build points for every method called by a suspect area of the application. It is very common to use one or more levels of caller side instrumentation to identify the time spent within an individual method or methods that have a suspected bottleneck. This is a useful way to fill in the next level to a Call Profile in Diagnostics.

The point in the following example instruments any call to a method that is performed within the `com.myCompany.myFoo` class by the "myMethod" method:

```
[MethodsCalledByFoo.myMethod]
scope = !.*
method = !.*
scope = !com\.myCompany\.myFoo\.myMethod.*
layer = FOO\other
```

The following is another example point that would capture the arguments to any "set" method called in `com.myCompany.myFoo` class by the "myMethod" method:

```
[SetMethodsCalledByFoo.myMethod]
scope = !.*
method = !set.*
scope = !com\.myCompany\.myFoo\.myMethod.*
detail = arg:1
layer = FOO\other
```

URI Aggregation Instrumentation

Applications typically use the same URL to access different workflow. If the application uses a URI (i.e. `http://server/myApplication?page=home`) argument to differentiate the between the workflow, Diagnostics can be configured to parse and treat the different URIs as different server requests.

URI aggregation is controlled from the [HttpCorrelation] point. A valid regular expression entry for "args_by_class" should be created for each URI pattern.

The point example below allows the following ServerRequests to appear uniquely in the Diagnostics console:

```
http://server/myApplication?page=home
http://server/myApplication?page=openReport
```

```
[HttpCorrelation]
args_by_class=!.*&page
```

More than one URI parameter can be used for URI parsing as shown in the following example:

```
args_by_class=!.*&page&role
```

Note: You should avoid using a session parameter or highly unique URI value because of the impact to overhead and data storage.

In a WebLogic environment you should set the `use.weblogic.get.parameter=true` in `<probeInstall>/etc/inst.properties` when using URI aggregation to prevent URI aggregation from consuming the ServletRequest's inputstream.

Using RMI Instrumentation

The RMI (Cross-VM) point in the capture points file is inactive by default. You must activate this point to capture the cross-vm processing in your application. If you have probes with this point activated on both sides of an RMI call, Diagnostics can correlate the call tree data from both virtual machines.

```
[RMI]
keyword = rmi
layer  = CrossVM
active = false
```

Note: If you are planning to use RMI instrumentation in a clustered environment, all servers in a cluster must have RMI instrumentation turned on in order to avoid application failure.

15

Instrumenting Java Applications From the Profiler

This chapter describes how to maintain the instrumentation that Mercury Diagnostics applies to the classes and methods of your Java applications from the Configuration tab in the J2EE Diagnostics Profiler.

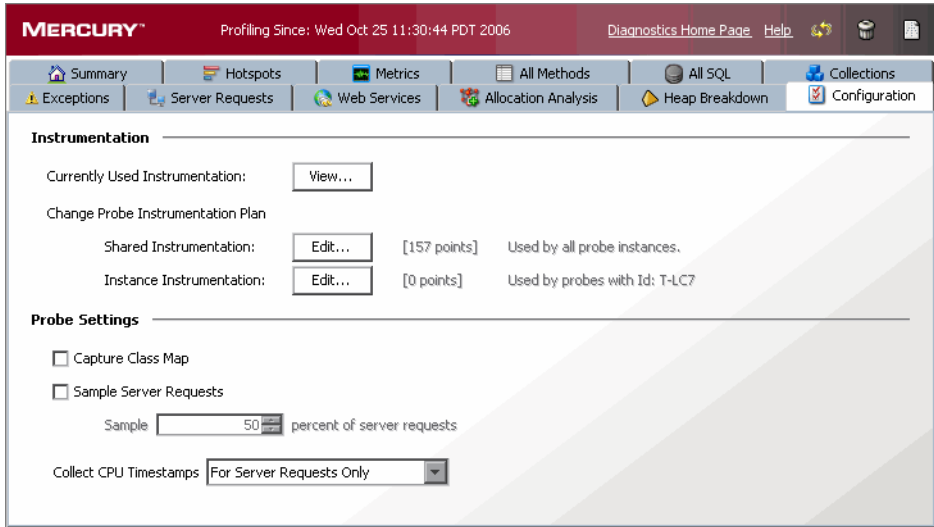
This chapter describes:	On page:
About the Configuration Tab	273
Maintaining Instrumentation from the Configuration Tab	275
Maintaining Probe Settings from the Configuration Tab	284

About the Configuration Tab

The Configuration tab in the J2EE Diagnostics Profiler provides a way for you to maintain the instrumentation points and the Probe configuration without having to manually edit the capture points file or property files.

You can access the Configuration tab from the J2EE Diagnostics Profiler whether profiling has been started on the Probe or not.

The Configuration tab in the J2EE Diagnostics Profiler is divided into the Instrumentation section and the Probe Settings section as shown in the following screen image:

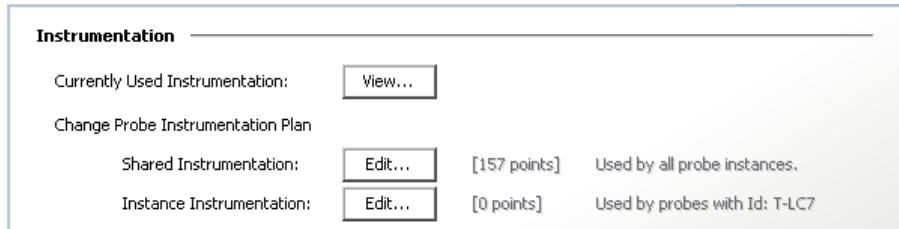


The Instrumentation section gives you access to view and update the instrumentation for the application that the Probe is monitoring. The edit dialogs that you access from the Instrumentation section allow you to view and edit the instrumentation points as defined in the capture points file that Diagnostics uses to instrument your applications.

The Probe Settings section of the Configuration tab allows you to configure the server request sampling and CPU timestamping the Probe uses as it captures the performance metrics for your application. For more information on the Probe Settings section see “Maintaining Probe Settings from the Configuration Tab” on page 284.

Maintaining Instrumentation from the Configuration Tab

From the Instrumentation section of the Configuration tab you can control the instrumentation that Diagnostics applies to your application.



Reviewing the Current Instrumentation

To review the layers, classes, and methods that have been instrumented as a result of the points in the current capture points file click **View...** in the Instrumentation section of the Configuration tab. The Profiler displays the Instrumented Layers page as shown in the following image:

Instrumented layers			
Layer	# Hits	Active Points	Actions
(Other)	55591	18 / 18	[Disable] [Clear # Hits]
Business Tier/EJB/Entity Bean	0	38 / 38	[Disable] [Clear # Hits]
Business Tier/EJB/Session Bean	26399	30 / 30	[Disable] [Clear # Hits]
Database/JDBC/Connection	212	59 / 59	[Disable] [Clear # Hits]
Database/JDBC/Execute	111	41 / 41	[Disable] [Clear # Hits]
Directory Service/JNDI	148	5 / 5	[Disable] [Clear # Hits]
Messaging/JMS/Connection	19	8 / 8	[Disable] [Clear # Hits]
Messaging/JMS/Consumer	0	3 / 3	[Disable] [Clear # Hits]
Messaging/JMS/Listener	0	6 / 6	[Disable] [Clear # Hits]
Messaging/JMS/Session	13	33 / 33	[Disable] [Clear # Hits]
Messaging/JMS/Session/Queue	8	9 / 9	[Disable] [Clear # Hits]
Messaging/JMS/Session/Topic	2	9 / 9	[Disable] [Clear # Hits]
Web Services	16249	2 / 2	[Disable] [Clear # Hits]
Web Tier/JSP	1605	1 / 1	[Disable] [Clear # Hits]
Web Tier/Servlet	42021	7 / 7	[Disable] [Clear # Hits]

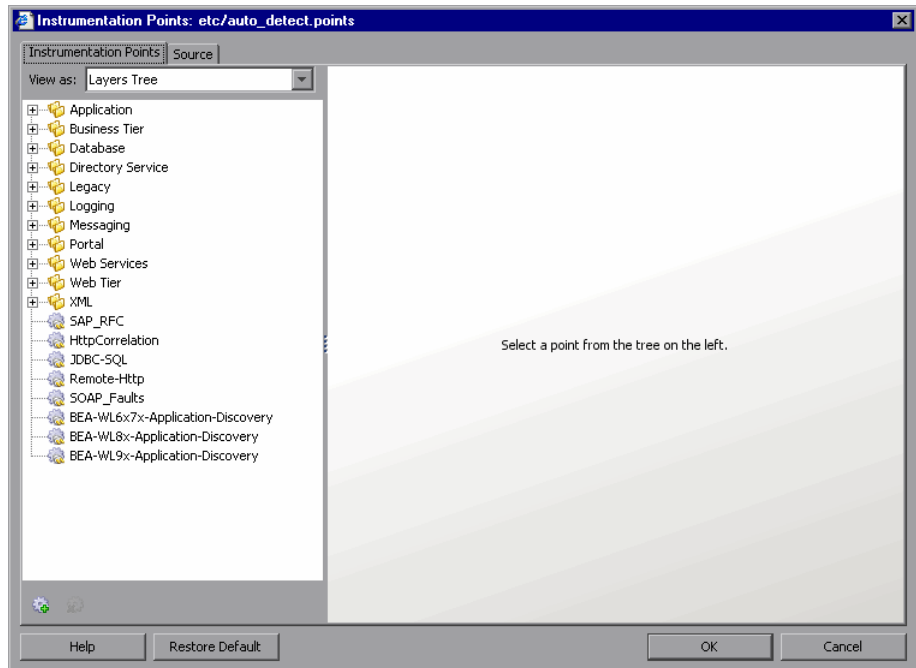
Mercury Diagnostics J2EE Probe "T-LC7", version 6.5.13.383

The Instrumented Layers page lists the layers that have been instrumented along with the number of times that the instrumentation points in the layer have been triggered and the number of points that are currently active in the layer. The following table describes the columns on this page:

Column	Description
Layer	Contains a list of the layers that have been instrumented. The layer names in this column are links that allow you to navigate to a page that provides the details about the processing in the layer that was monitored by the Probe. Note: Only the layers defined in points that were actually instrumented are listed.
# Hits	Contains a count of the number of times that the classes and methods that are monitored by the points in the listed layer have been invoked. You can reset the count using the Clear # of Hits link in the Actions column.
Active Points	Contains the count of the number of points that are currently active as well as the total number of points that have been defined for the particular layer.
Actions	Contains links that allow you to manipulate the information for the listed layers. The available action are: <ul style="list-style-type: none"> <li data-bbox="648 1058 1222 1306">▶ Disable: Disables all of the points in the selected layer so that they no longer capture data. The instrumentation stays in place and can be enabled again. Enabling or disabling points here is effective only until the next restart of your application. To change the default enabled state for a point see “Coding Points in the Capture Points File” on page 241. <li data-bbox="648 1318 1222 1372">▶ Clear # Hits: Resets the hit count displayed in the # Hits column for the selected layer.

Maintaining the Instrumentation Points

To maintain the points that provide the instrumentation instructions the tell the Probe what to monitor in your application, navigate to the Configuration tab in the J2EE Diagnostics Profiler and click **Edit...** for either the Shared Instrumentation or the Instance Instrumentation. The Instrumentation Points dialog is displayed as shown in the following image:

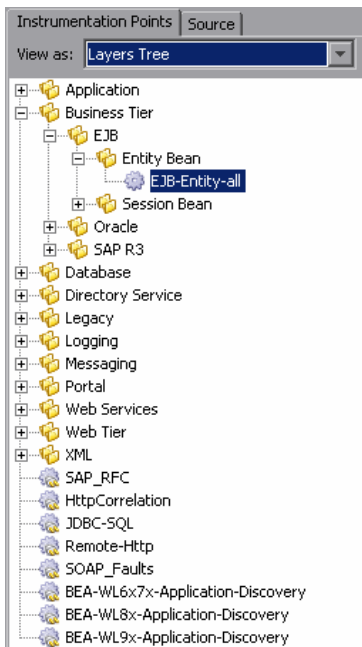


There are two ways you can edit the instrumentation: Visually using a list or tree of points on the Instrumentation Points tab or via the source of the capture points file on the Source tab.

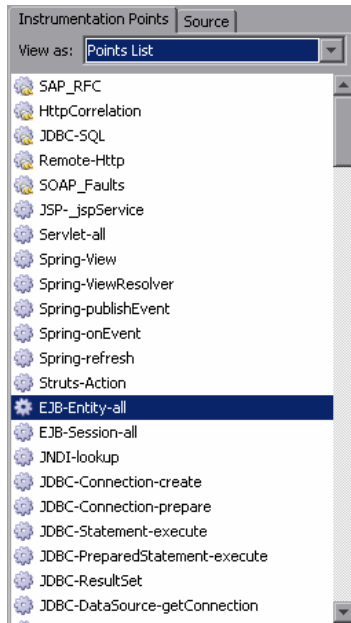
Selecting and Viewing an Existing Point

The navigation bar in the Instrumentation Points dialog helps you to locate the points in the capture points file that you would like to maintain. By making a selection from the **View as** dropdown you can choose the format in which the points are listed.

When you select Layer Tree from the **View as** dropdown, the Probe lists the points in the capture points file in a tree structure according to the layers and sublayers that you assigned to the point as shown in the following image:



When you select **Points List** from the **View as** dropdown, the Probe lists the points in the capture points file in ascending alphabetical order as shown in the following image:

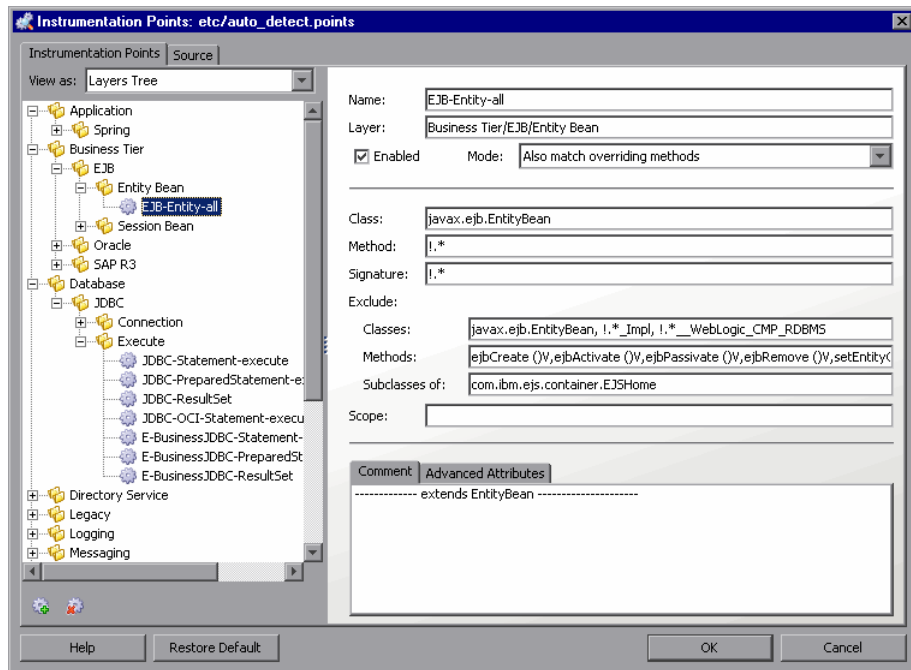


When you have located the point that you want to view or maintain, select the point in the navigation bar. The Probe displays the details of the selected point in the view/edit panel where you can maintain the point.

Updating an Existing Point

When you select a layer or sublayer from the navigation bar, the view/edit panel contains only a prompt to remind you to select a point.

To update an existing point, select the point from the navigation bar so that the Profiler displays the details for the point in the Instrumentation Points tab of the view/edit panel as shown in the following example:

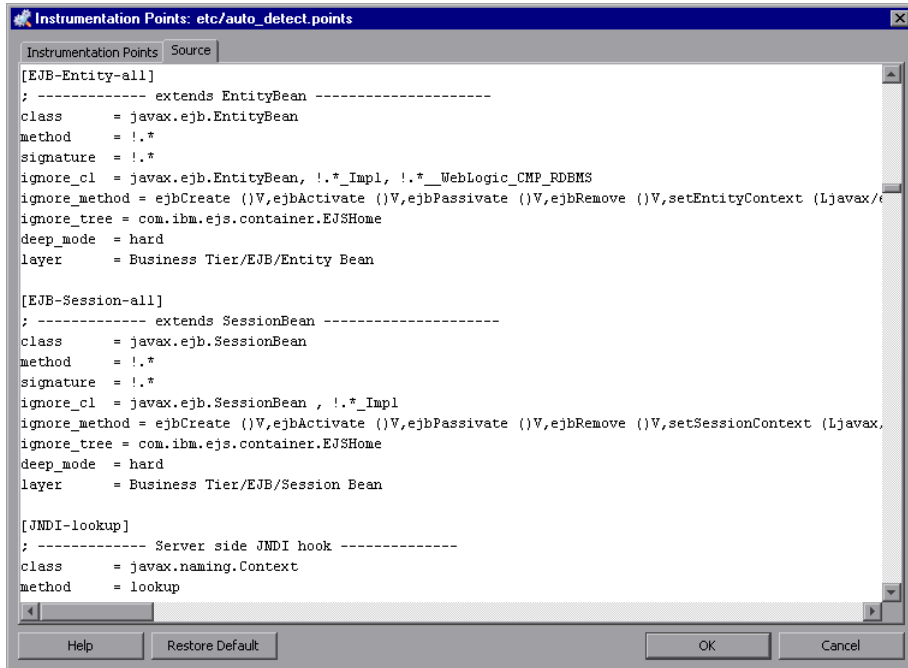


The arguments that are commonly used when defining a point in the capture points file are displayed as separate data fields to make it easier for you to make any necessary updates. Other, more advanced arguments are displayed in the Advanced Attributes tab at the bottom of the Instrumentation Points tab. Comments for the point are displayed in the Comments tab.

All of the arguments that can be used to define a point in the capture points file have been documented in “Coding Points in the Capture Points File” on page 241. The following table will help you to cross reference the arguments displayed in the Instrumentation Points tab with the capture points file arguments documented in this section. You may also find this cross reference table useful when reviewing the points displayed in the Source tab.

Instrumentation Point Tab	Capture Points File Argument
Name	Point-Name
Layer	layer
Enabled	keyword = disabled
Mode	deep_mode
Class	class
Method	method
Signature	signature
Exclude Classes	ignore_cl
Exclude Methods	ignore_method
Exclude Subclasses	ignore_tree
Scope	scope
Advanced Attributes	detail, code-key, tierDefinition, layerType, merge_classes, merge_url_to, ignore_tree, ignoreScope

An example of the Source tab is shown in the following image:



Deleting an Existing Point or Layer

You may delete a point or layer listed in the navigation bar.

To delete a point or layer:

- 1 Select the point or layer from the Navigation bar on the Instrumentation Points tab.



- 2 Click Delete Point. The Profiler deletes the selected entity from the list in the navigation bar.

The selected entity is not actually deleted from the capture points file until you apply all of your instrumentation points updates from the Configuration tab in the Profiler.

- 3 Close the Instrumentation Points dialog by clicking **OK**.
- 4 Apply all of the changes made using the Configuration tab by clicking **Apply Changes**.

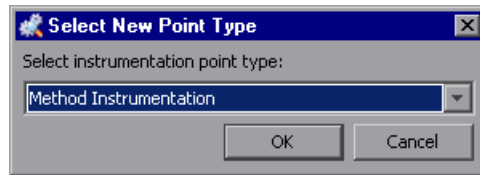
Adding a New Point

You may add a point to the instrumentation.

To add a point:



- 1 Click New Point. The Profiler displays the Select New Point Type dialog as shown in the following image:



- 2 Select the appropriate point type from the dropdown and click **OK**.

The Profiler displays the Instrumentation Points tab with the view/edit section initialized for creating a new point for the selected point type.

- 3 Enter the arguments and comments for the new point into the appropriate locations on the tab.

When you enter the Layer information the entry for the new point in the navigation bar is updated to show the point in the correct existing layer or, if the layer that you specified does not already exist, with a brand new layer.

The new point is not actually added to the capture points file until you apply all of your instrumentation points updates from the Configuration tab in the Profiler.

- 4 Close the Instrumentation Points dialog by clicking **OK**.
- 5 Apply all of the changes made using the Configuration tab by clicking **Apply Changes**.

Restoring Default Points

When you have finished diagnosing a problem using the Profiler or Mercury Diagnostics you may want to restore the default instrumentation to avoid incurring the overhead from a more robust instrumentation.

To restore the default settings to the instrumentation:

1 Click **Restore Defaults**.

The instrumentation points are not actually added to the capture points file until you apply all of your instrumentation points updates from the Configuration tab in the Profiler.

2 Close the Instrumentation Points dialog by clicking **OK**.

3 Apply all of the changes made using the Configuration tab by clicking **Apply Changes**.

Maintaining Probe Settings from the Configuration Tab

From the Probe Settings section of the Configuration tab you can configure the server request sampling and CPU timestamping the Probe uses as it captures the performance metrics for your application.

Probe Settings

Capture Class Map

Sample Server Requests

Sample percent of server requests

Collect CPU Timestamps

Controlling Class Map Capture

The class map allows Diagnostics to provide more details about the classes and methods that are invoked by a server request. This information can help you to narrow your search for the source of a performance issue. The class map information can be viewed by drilling down on the Layers listed on the Instrumented Layers page as described in “Reviewing the Current Instrumentation” on page 275. The cost for using class map comes from the additional overhead that creating the map places upon the Probe host.

To enable class map capture:

- 1** Check the **Capture Class Map** check box.
- 2** When you have completed making changes to the Configuration tab, click **Apply Changes**.
- 3** Restart the Probe to allow the Probe to begin class map capture.

To disable class map capture:

- 1** Uncheck the **Capture Class Map** check box.
- 2** When you have completed making changes to the Configuration tab, click **Apply Changes**.
- 3** Restart the Probe to allow the Probe to stop class map capture.

Controlling Server Request Sampling

Server Request sampling allows Diagnostics to gather detailed information for a subset of all of the server requests executed in your application. This reduces the overhead of the Probe because no information is captured for Server requests that are not sampled. As long as the sampling rate is not too low, the average results from all sampled server requests should provide results that are reasonably accurate and meaningful for understanding the performance of your applications.

You control when sampling is enabled, and what percentage of the server requests will be sampled.

To enable server request sampling:

- 1 Check the **Sample Server Requests** check box.
- 2 Set the sampling percentage using the spinner.
- 3 When you have completed making changes to the Configuration tab, click **Apply Changes**.

Your changes take effect immediately. There is no need to restart the Probe.

To disable server request sampling:

- 1 Uncheck the **Sample Server Requests** check box.
- 2 When you have completed making changes to the Configuration tab, click **Apply Changes**.

Your changes take effect immediately. There is no need to restart the Probe.

Controlling CPU Timestamp Collection

The CPU timestamps are used to calculate the amount of exclusive CPU time that a method uses. This information can be viewed on the **Hotspots** tab in the J2EE Diagnostics Profiler.

CPU timestamping is supported on the following platforms:

- Windows
- Solaris 8+
- AIX 5+

Note: Use caution when enabling the collection of CPU timestamps because of the increase in Diagnostics overhead. The increased overhead is caused by an additional call for each method that is needed to collect the timestamp.

To select CPU timestamp collection method:

- 1 Select the CPU Timestamp collection method from the **Collect CPU Timestamps** dropdown.

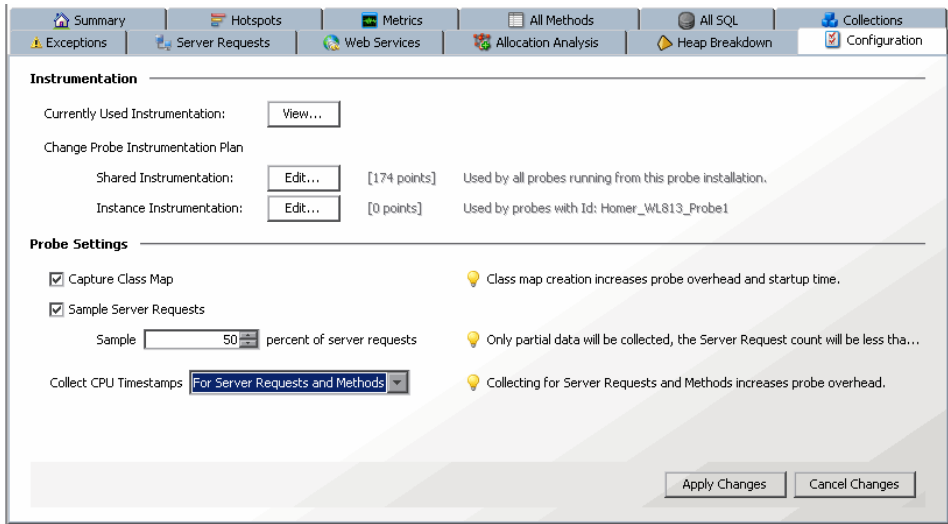
CPU Timestamp Collection Method	Description
None	No CPU Timestamps
For Server Requests Only	CPU timestamps will only be collected for server requests
For Server Request and Methods	CPU timestamps will be collected for both server requests and methods

- 2 When you have completed making changes to the Configuration tab, click **Apply Changes**.

Your changes take effect immediately. There is no need to restart the Probe.

Restarting the Probe

When you click **Apply Changes** on the Configuration tab all of the updates that you made in the Instrumentation section and Probe Settings sections of the Configuration tab are applied to the capture points file and the property files. If the changes that you made require that the Probe be restarted so that the changes that you made can take effect, the Profiler displays messages in the bottom left corner of the Configuration tab as a reminder. The only field that requires a Probe restart is the Capture Class Map check box.



16

Instrumentation Layers

This chapter describes the default instrumentation layers that are defined in the points of the capture points file when a Probe is first installed.

This chapter describes:	On page:
About Instrumentation Layers	289
Understanding Diagnostics Layers	290

About Instrumentation Layers

Mercury Diagnostics groups the performance metrics for classes and methods into *layers* and *sub-layers* based upon the instructions provided in the capture points file. The default layers were defined so that the performance metrics for processing in the application that used similar system resources could be reported together. The layers make it easier for you to isolate and identify the areas of the system that may be contributing to performance issues.

For information on maintaining points and layers see “Mandatory Point Arguments” on page 242.

Understanding Diagnostics Layers

The following tables list the default layers and sub-layers that have been defined for typical J2EE and .NET classes and methods, and indicate the points in the capture points file where the classes and methods are assigned to the layers. This information should help you interpret and customize the points in the capture points file.

Platform specific layers have also been defined in the capture points file. These layers are, for the most part, sub-layers of the top-level parent layers defined in the tables below. For information about the platform specific layers, see the *Mercury Diagnostics User's Guide*.

J2EE Layers

Layer	Sub-Layers	Parent Layer	Defined in Points
Web Tier	JSP Servlets Struts		JSP-_jspService Servlet-all Struts-Action
JSP		Web Tier	JSP-_jspService
Servlets		Web Tier	Servlet-all
Struts		Web Tier	Struts-Action
Business Tier	EJB		EJB-Entity-all EJB-Session-all
EJB	Entity Bean Session Bean	Business Tier	EJB-Entity-all EJB-Session-all
Entity Bean		EJB	EJB-Entity-all
Session Bean		EJB	EJB-Session-all
Directory Service	JNDI		JNDI-lookup
JNDI		Directory Service	JNDI-lookup

Layer	Sub-Layers	Parent Layer	Defined in Points
Database	JDBC		JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect JDBC-OCI-Statement-execute
JDBC	Execute Connections	Database	JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect JDBC-OCI-Statement-execute
Execute		JDBC	JDBC-Connection-create JDBC-Connection-prepare JDBC-Statement-execute JDBC-PreparedStatement-execute JDBC-ResultSet JDBC-OCI-Statement-execute

Layer	Sub-Layers	Parent Layer	Defined in Points
Connections		JDBC	JDBC-DataSource-getConnection JDBC-XADataSource-getConnection JDBC-Driver-connect
Messaging	JMS		JMS-MessageProducer JMS-MessageListener JMS-MessageConsumer
JMS	Producer Listener Consumer	Messaging	JMS-MessageProducer JMS-MessageListener JMS-MessageConsumer
Producer		JMS	JMS-MessageProducer
Listener		JMS	JMS-MessageListener
Consumer		JMS	JMS-MessageConsumer

.NET Layers

Layer	Sub-Layers	Parent Layer	Defined in Points
Web Tier	ASP.NET		
Database	ADO		ADO 1 ADO 2 ADO 3 ADO 4
ADO	Execute Connection	Database	ADO 1 ADO 2 ADO 3 ADO 4

Layer	Sub-Layers	Parent Layer	Defined in Points
Execute		ADO	ADO 1 ADO 2 ADO 3
Connection		ADO	ADO 4
Messaging	MSMQ		MSMQ Synchronous MSMQ Asynchronous MSMQ ReceiveCompleted EventHandler MSMQ PeekCompleted EventHandler
MSMQ		Messaging	MSMQ Synchronous MSMQ Asynchronous MSMQ ReceiveCompleted EventHandler MSMQ PeekCompleted EventHandler

Portal Layers

Diagnostics groups the performance metrics for the classes and method calls associated with processing for portals into Portal Component layers. Each Portal Component layer is broken down to layers for the portal lifecycle methods. For more information about portal layers, see the *Mercury Diagnostics User's Guide*.

17

Using Solution Templates

This chapter describes the Mercury Diagnostics instrumentation solution templates and how to view the performance metrics gathered by the points that make up the templates.

This chapter describes:	On page:
About Solution Templates	295
Understanding Solution Template Data	296
Viewing Solution Template Performance Metrics	296

About Solution Templates

Solution templates are predefined instrumentation points included in the capture points file that have been created for several leading application middleware servers and frameworks. The solution templates allow Diagnostics to capture platform specific performance metrics and present them in the Diagnostics views. For more information about the capture points file and the instrumentation points see Chapter 14, “Instrumenting an Application.”

Mercury Diagnostics provides solution templates for the following servers and frameworks:

- ▶ **WebLogic:** The WebLogic Portal Server solution template collects metrics for BEA WebLogic Portal Server WL 8.1 SP3, SP4.
- ▶ **WebSphere:** The WebSphere Portal Server solution template collects metrics for WebSphere 5.1.

- ▶ **Oracle 11i:** The Oracle 11i solution template collects metrics for the Web-based applications framework.
- ▶ **SAP:** The SAP solution template collects metrics for iViews in the SAP Web Application Server 6.40 SP 10, or later.

Understanding Solution Template Data

Mercury Diagnostics groups the performance metrics for classes and methods into *layers* and *sub-layers* based upon the resources that the application invokes to perform the processing. These layers make it easier to isolate and identify the areas of the system that may be contributing to performance issues. For more information on layers see Chapter 16, “Instrumentation Layers.”

The solution templates allow you to see performance metrics for business services that are associated with various middleware servers and frameworks in new layers across the Diagnostics application.

Viewing Solution Template Performance Metrics

Mercury Diagnostics displays unique performance metrics for each of the solution templates. The following discussion describes the performance metrics that are gathered by the instrumentation points in each of the solution templates.

WebLogic

The WebLogic Portal Server solution template lets you view the performance metrics for the following business services in layers across the Diagnostics application:

Service	Layer	Description
Entitlements	Portal/BEA/Entitlement	Entitlement API's
Content management	Portal/BEA/Content	Content management API's
User profiles	Portal/BEA/UserProfile	User profile API's
Personalization	Portal/BEA/Personalization	Personalization API's
Portal components	Portal/BEA/Portlet	Portal component rendering API's

Note: The Portal/BEA/Web Services layer, representing BEA Web Service, is commented out of the points file by default.

The following is an example of a Diagnostics view that displays the WebLogic business services layers with metrics gathered using the solution template.



Note: You can also see WebLogic performance metrics on the Portal views in Mercury Diagnostics. For more information about using the Portal views, refer to the *Mercury Diagnostics User's Guide*.

WebSphere

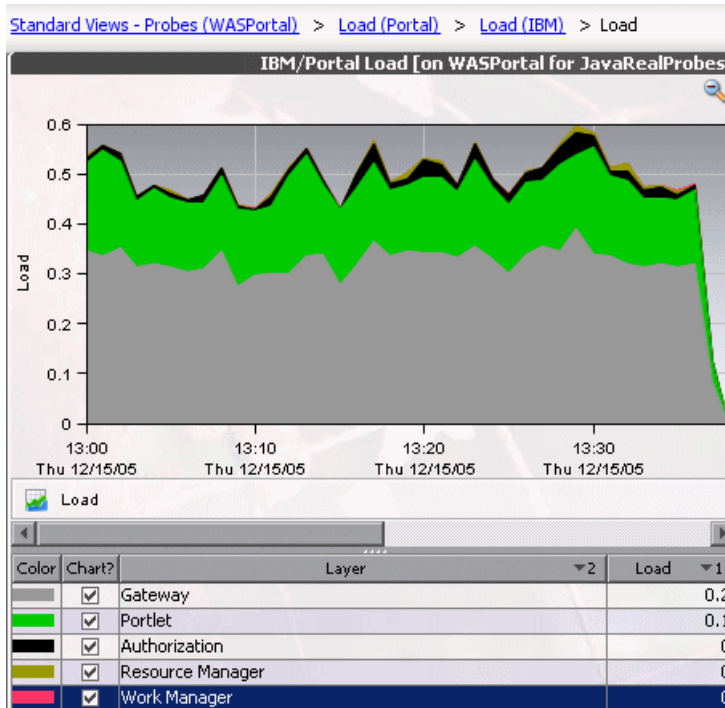
The WebSphere Portal Server solution template lets you view the performance metrics for the following business services in layers across the Diagnostics application:

Service	Layer
WAS Portal Gateway Servlet	Portal/IBM/Gateway
WAS Portal Authentication	Portal/IBM/Authentication
WAS Portal PortalAuthorization	Portal/IBM/Authorization
WAS Portal Resource Manager	Portal/IBM/ResourceManager
WAS Portal Portlet Adapter Interface Actions	Portal/Jetspeed/Portlet/Action
WAS Portlet Lifecycle	Portal/Jetspeed/Portlet/Lifecycle
WAS Portal Rendering	Portal/IBM/Portlet/Rendering
WAS Portal Work Manager	Portal/IBM/WorkManager
WAS Portal Portlet Services	Portal/IBM/Portlet/Services
WAS Portal Phases	Portal/IBM/Portlet/Phases
WAS Portal Services	Portal/IBM/Portlet/Services
WAS Portal Portlet Filter	Portal/IBM/Portlet/Filter

Note: The Portal/IBM/Portlet/Phases layer, representing WAS Portal Phases, is commented out of the points file by default. Portal Phases do not need to be captured unless you want to understand how the container processes each phase of the portlet's Init-Action-Render cycle.

The Portal/IBM/Portlet/Services layer, representing WAS Portal Services, is also commented out of the points file by default. Many portal services such as the AccessControlService service and PortletFilterService are exposed in other layers, but you might want to uncomment this section if you are interested in some of the other services provided by the portal container.

The following is an example of a Diagnostics view that displays the WebSphere business services layers with metrics gathered using the solution template:



Oracle

The Oracle 11i solution templates you view the performance metrics for the following two layers found underneath the root **Business Tier** layer:

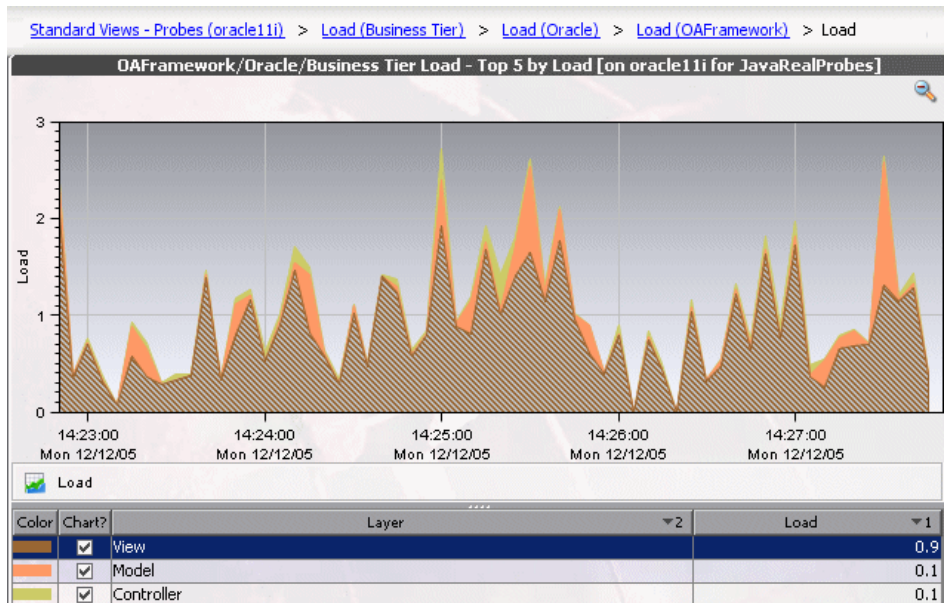
- Caching
- OAFramework (Oracle Application Framework)

You can drill down further into the OAFramework layer to display the following three layers:

- Model
- View
- Controller

The MVC layers (Model, View, and Controller) relate to the different parts of the Oracle 11i application framework.

The following is an example of a Diagnostics view that displays the MVC layers with metrics gathered using the solution template.



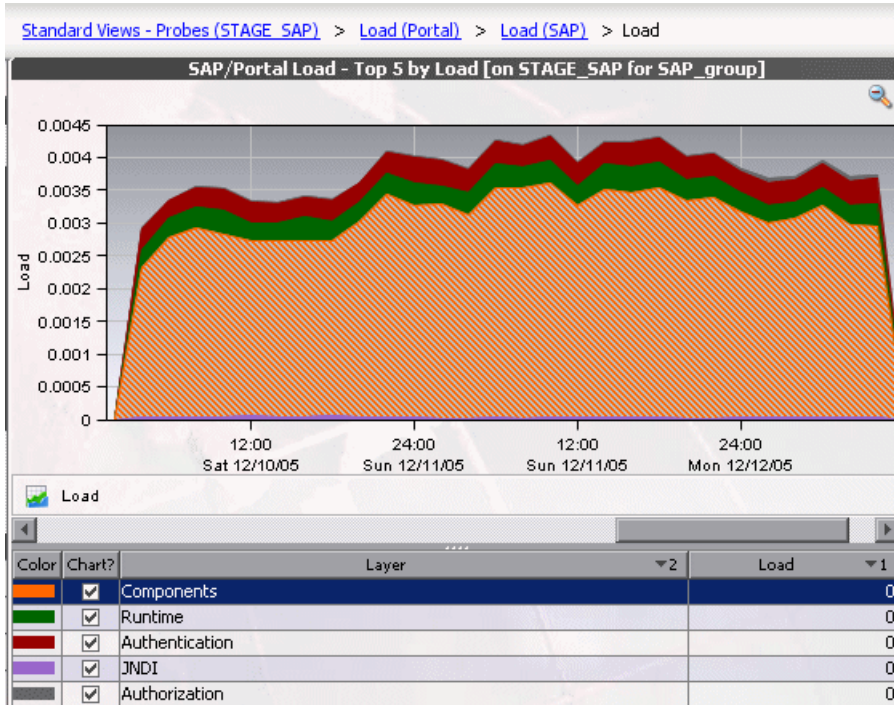
SAP

The SAP Portal Server Solution Template lets you view the performance metrics for the following business services in layers across the Diagnostics application:

Service	Layer	Comment
Runtime	Portal/SAP/Runtime	General runtime part of infrastructure
Authorization	Portal/SAP/Authorization	Core application component dedicated to authorization services
Authentication	Portal/SAP/Authentication	Core application component dedicated to authentication services
JNDI Services:	Portal/SAP/JNDI	Core application component dedicated to JNDI services
Components Runtime	Portal/SAP/Components/ Runtime	General runtime part of Portal components
Content Generation (Portal Components)	Portal/SAP/Components/ Content Generation	Content generation part of Portal components
Content Generation (Service Components)	Portal/SAP/Components/ Service	Content generation Service components (Note: Commented out. Optimize before use)
Content Generation (Request Object Specific)	Portal/SAP/Components/ Request	Content generation Request object specific (Note: Commented out. Optimize before use)
Content Generation (Profile Components)	Portal/SAP/Components/ Profile	Content generation Profile components
Content Generation (Config Components)	Portal/SAP/Components/ Config	Content generation configuration components

Service	Layer	Comment
Content Generation (Response Object Specific)	Portal/SAP/Components/Response	Content generation Response object specific
R3 Content Generation	Business Tier/SAP R3	Content generation specific to R3 backend
Dynpage Content Generation	Web Tier/SAP/DynPage	Content generation specific to DynPage
JSP Dynpage Content Generation	Web Tier/SAP/JSPDynPage	Content generation specific to JSP DynPage
Page Processor Component	Web Tier/SAP/Page Processor	Content generation specific to the Page Processor

The following is an example of a Diagnostics view that displays the SAP business services layers with metrics gathered using the solution template. The names of the layers are equivalent to the names of the remote function calls (RFCs).

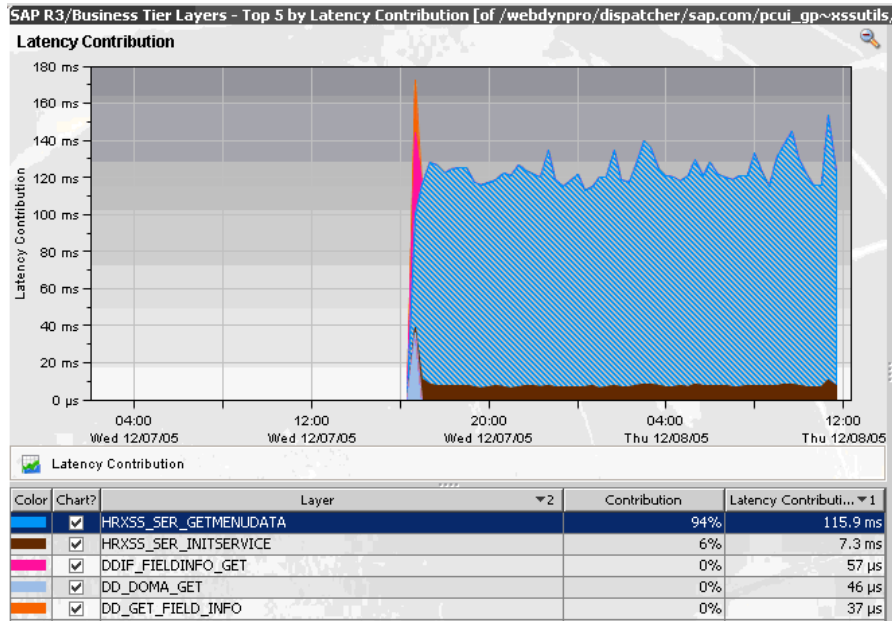


SAP R3 Layer

In a typical SAP Web Application Server deployment (WAS), the SAP WAS is connected to a backend SAP R3 Server. Communication with the SAP R3 Server takes place through RFCs, using SAP JCO (Java Connector). Mercury Diagnostics captures each distinct call and presents them in sub-layers.

You access these sub-layers by drilling down from the Business Tier layer. For example, if you start from the Server Summary view in Mercury Diagnostics, you can drill down to the SAP R3 sub-layers as follows: **Standard views - Server Summary > Server Requests > Layers (Business Tier) > Layers (SAP R3) > Layers.**

The following is an example of a Diagnostics view that displays the SAP R3 sub-layers with metrics gathered using the solution template.



Note: You can also see SAP Portal performance metrics on the portal views in Mercury Diagnostics. For more information about using the Portal views, refer to the *Mercury Diagnostics User's Guide*.

Part VI

Advanced Component Configuration

18

Advanced Diagnostics Server Configuration

This chapter describes advanced configuration instructions for the Diagnostics Server that are intended for experienced users with in-depth knowledge of this product. Please use caution when modifying any of the Diagnostics components' properties.

This chapter describes:	On page:
Synchronizing Time Between Diagnostics Components	310
Configuring the Diagnostics Server for a Large Installation	314
Overriding the Default Diagnostics Server Host Name	317
Changing the Default Diagnostics Server Port	318
Configuring the Diagnostics Server for Multi-Homed Environments	319
Reducing Diagnostics Server Memory Usage	323
Configuring Fragment Name Based Trimming	324
Preparing a High Availability Diagnostics Server	325
LoadRunner / Performance Center Diagnostics Server Assignments	327
Configuring the Diagnostics Server for the LoadRunner Offline File	328
Configuring Diagnostics Using the Diagnostics Server Configuration Pages	331

Synchronizing Time Between Diagnostics Components

In order for Diagnostics data to be stored and correlated properly, it is critical that time is synchronized between the Diagnostics components. To facilitate synchronization of data, the Diagnostics data is adjusted and saved to the synchronized GMT time of the Diagnostics Server in Commander mode. Synchronization makes it possible to display the data correctly for any local time in which the UI may be located.

The sections that follow describe how time synchronization works, and how to configure the components properly so that the time will be synchronized.

Understanding Time Synchronization

Time synchronization in Diagnostics begins with the Diagnostics Server in Commander mode determining the difference between its time and the GMT time provided by a designated **Time Source**. The **Time Source** to be used is set using the `timemanager.time_source` property in `<diagnostics_server_install_dir>/etc/server.properties`.

The valid values for the `timemanager.time_source` property are:

- **NTP**. Indicates that an NTP Server is to be used as the source of GMT time. This is the default value.

The NTP servers that are to be used are listed as values of the `timemanager.ntp.servers` property in `<diagnostics_server_install_dir>/etc/server.properties`.

Note: Ensure that one of the NTP servers in the list can be contacted from the Diagnostics Server, or add your local NTP server as the first server in the list.

- **BAC.** Indicates that the registered Mercury Business Availability Center core server is to be used as the source of GMT time.

Note: If Mercury Business Availability Center is configured to use Database time, you should also configure the Diagnostics Server in Commander mode to use this setting as the time source.

- **SERVER.** Indicates that the Diagnostics Server in Commander mode is to be used as the **Time Source**.

This should only be used when the Diagnostics Server is being used in Standalone mode.

The Diagnostics Servers that are in Mediator mode synchronize their time by establishing the time difference between the Diagnostics Server in Mediator mode and the Diagnostics Server in Commander mode.

If the Diagnostics Server in Commander mode has not yet synchronized with the **Time Source**, then the Diagnostics Servers in Mediator mode are considered to be “unsynched”. The Diagnostics Servers in Mediator mode that are unsynched will attempt to synchronize their time every 15 seconds until they succeed.

When a Diagnostics probe connects to a Diagnostics Server in Mediator mode or to a Diagnostics Server in Commander mode, the time difference is established between the Diagnostics Server and the probe.

If the probe attempts to connect to a Diagnostics Server that is still “unsynched,” then the probe connection is not allowed, and is dropped.

Since the data is stored based upon the GMT, differences in time zones or daylight savings times for the various components are not an issue. For example, the data that is displayed in the Diagnostics UI can be adjusted to display correctly for the time zone in which the UI is running.

Note: All data has been adjusted and saved to the synchronized GMT time of the Diagnostics Server in Commander mode. So, if the UI is running on a machine whose time has not been synchronized properly with the **Time Source**, the data displayed in the UI will appear shifted by the amount of time the UI machine is off from the synchronized GMT time.

Configuring the Time Synchronization on the Diagnostics Server

You synchronize the Diagnostics Server in Commanding mode by performing the procedure below.

Note: Time Synchronization settings for Diagnostics Servers in Mediator mode are ignored because their time is automatically synchronized with the Diagnostics Server in Commander mode.

To ensure that time on the Diagnostics Server in Commander mode can be synchronized:

- 1 The default configuration for the Diagnostics Server is set such that the value of the **timemanager.time_source** property in `<diagnostics_server_install_dir>/etc/server.properties` is **NTP**.

If the Diagnostics Server has an internet connection and the ability to connect to a server in the list of available NTP servers specified in the **timemanager.ntp.servers** property, then the default configuration will work and no changes are necessary.

Since Mercury Business Availability Center also uses NTP for time synchronization by default, this is the recommended setting.

2 If the Diagnostics Server does not have an internet connection or the ability to connect to the list of available NTP servers specified in **timemanager.ntp.servers** property, then you *must* do one of the following:

- Setup a local NTP server that can be contacted by the Diagnostics Server in Commander mode. List this local NTP server as the first entry in the **timemanager.ntp.servers** property in **<diagnostics_server_install_dir>/etc/server.properties**.

Note: It is recommended to have backup NTP servers in case the primary NTP server is not available.

- If you are using Diagnostics in a Mercury Business Availability Center or Mercury Managed Services environment, then you can set the **timemanager.time_source** property in **<diagnostics_server_install_dir>/etc/server.properties** to **BAC** to indicate Mercury Business Availability Center. This causes the Diagnostics Server to connect to the registered Mercury Business Availability Center core server to establish the time.

Note: To set up Mercury Business Availability Center to use Diagnostics, see Chapter 10, “Setting Up Mercury Business Availability Center 6.1 or Later to Use Diagnostics.”

- If the Diagnostics Server in Commander mode is to be used in Standalone mode, with no intention of using it with Mercury Business Availability Center, and you have no internet connection that will allow you to synchronize time with an NTP server, then you can set the **timemanager.time_source** property in **<diagnostics_server_install_dir>/etc/server.properties** to **SERVER**. This causes the Diagnostics Server to use its own time as the **Time Source**.

Note: It is recommended that you do not change the value of the `timemanager.time_source` property in `<diagnostics_server_install_dir>/etc/server.properties` once data has been captured and persisted using the designated **Time Source**. Changing the **Time Source** after data has been captured could result in a significant corruption to the data that has been captured and persisted. This is because the data that was persisted may have been captured while the Diagnostics Server was not synchronized with GMT. If the data that is captured later is captured while the Diagnostics Server is synchronized with GMT, it is possible that data could get re-aggregated multiple times or could get recorded into time buckets where it does not belong.

Configuring the Diagnostics Server for a Large Installation

If you are using a Diagnostics Server in Mediator mode with more than 20 probes, it is recommended that you make modifications to the default configuration of the Diagnostics Server.

Note: These changes to the configuration are not needed for the Diagnostics Server in Commander mode unless it also has probes assigned to it so that it also serves as a Diagnostics Server in Mediator mode.

Adjusting the Heap Size

The size of the heap can impact the performance of the Diagnostics Server in Mediator mode. If the heap is too small you may experience problems where the Diagnostics Server in Mediator mode “hangs” for periods of time. If the heap is too large it is possible that the Diagnostics Server in Mediator mode will experience long garbage collection delays.

The default value for the heap size is 512 MB. The heap size is set in the **server.nanny** file located at:

<diagnostics_server_install_dir>\nanny\windows\dat\nanny\
for Windows, or

<diagnostics_server_install_dir>/nanny/solaris/launch_service/dat/nanny/
for Solaris.

Use the following VM argument to set the size, where ??? is the size in MB:

-Xmx???m

If you encounter problems with the Diagnostics Server in Mediator mode "hanging", you can increase the heap size specified by updating the value specified in the **-Xmx???m** option.

To adjust the heap size of the Diagnostics Server in Mediator mode:

- 1 Determine the amount of heap that the Diagnostics Server in Mediator mode will need based upon the following table:

Number of Probes	Recommended Heap Size
0 - 10	512 MB
11 - 20	700 MB
20 - 30	1,400 MB

Note: The recommended heap size should not exceed more than 75% of the physical memory of the machine. If a machine has 1 G, then the heap size should not exceed 768 MB.

- 2 Open the **server.nanny** file that is to be edited. This file is located at:

<diagnostics_server_install_dir>\nanny\windows\dat\nanny\
for Windows, or

<diagnostics_server_install_dir>/nanny/solaris/launch_service/dat/nanny/
for Solaris.

- 3 On the `start_<platform>` line that is appropriate, replace the heap size specified in the `-Xmx???m` option with the optimal heap size that you calculated:

-Xmx???m

Continuing the previous example, the current heap size, represented by "???" is replaced with 768 MB.

-Xmx768m

Before you modify this line in the `server.nanny` file it will look like this:

```
start_nt="C:\MercuryDiagnostics\Server\jre\bin\javaw.exe" -server -Xmx512m
-Dsun.net.client.defaultReadTimeout=70000
-Dsun.net.client.defaultConnectTimeout=30000
"-javaagent:C:\MercuryDiagnostics\Server\probelib\probeagent.jar"
-classpath "C:\MercuryDiagnostics\Server\lib\mediator.jar;
C:\MercuryDiagnostics\Server\lib\loading.jar;
C:\MercuryDiagnostics\Server\lib\common.jar;
C:\MercuryDiagnostics\Server\lib\mercury_picocontainer-1.1.jar"
com.mercury.opal.mediator.util.DiagnosticsServer
```

After you modify this line in the `server.nanny` file it will look like this:

```
start_nt="C:\MercuryDiagnostics\Server\jre\bin\javaw.exe" -server -Xmx768m -
Dsun.net.client.defaultReadTimeout=70000
-Dsun.net.client.defaultConnectTimeout=30000
"-javaagent:C:\MercuryDiagnostics\Server\probelib\probeagent.jar"
-classpath "C:\MercuryDiagnostics\Server\lib\mediator.jar;
C:\MercuryDiagnostics\Server\lib\loading.jar;
C:\MercuryDiagnostics\Server\lib\common.jar;
C:\MercuryDiagnostics\Server\lib\mercury_picocontainer-1.1.jar"
com.mercury.opal.mediator.util.DiagnosticsServer
```

Overriding the Default Diagnostics Server Host Name

In situations where a firewall or NAT is in place, or where the host for the Diagnostics Server in Mediator mode has been configured as a multi-homed device, it may not be possible for the Diagnostics Server in Commander mode to communicate with the Diagnostics Server in Mediator mode using the host name that was assigned when the Diagnostics Server in Mediator mode was installed. The **registered_hostname** property allows you to override the default host name that the Diagnostics Server in Mediator mode uses to register itself with the Diagnostics Server in Commander mode.

To override the default host name for a Diagnostics Server in Mediator mode set the **registered_hostname** property located in `<diagnostics_server_install_dir>/etc/server.properties` to an alternate machine name or IP address that will allow the Diagnostics Server in Commander mode to communicate with the Diagnostics Server in Mediator mode.

Changing the Default Diagnostics Server Port

If the configuration of the Diagnostics Server host will not allow the default Diagnostics port to be used, you may choose a different port for the Diagnostics Server communications with the probes and other Diagnostics Servers.

If you decide to use an alternative port number after you have deployed Diagnostics, you must update the properties in the following table for each of the indicated components in your deployment with the new port number in order to ensure that the proper communications can take place.

Component Type	Properties
Diagnostics Server in Commander mode	<diagnostics_server_install_dir>/etc ► webservice.properties - jetty.port ► mediator.properties - commander.url ► probe/etc/dispatcher.properties - registrar.url
Diagnostics Server in Mediator mode	<diagnostics_server_install_dir>/etc ► mediator.properties - commander.url <diagnostics_server_install_dir>/probe/etc ► dispatcher.properties - registrar.url
Probes	<probe_install_dir>/etc ► dispatcher.properties - registrar.url

Configuring the Diagnostics Server for Multi-Homed Environments

The machines that host the Diagnostics Server can be configured with more than one Network Interface Card (NIC), and the Diagnostics Server process listens on all interfaces on its host. Some customer environments do not allow applications to listen on all network interfaces on a machine. If your environment has this restriction, use the following instructions to configure the Diagnostics Server to listen on specific network interfaces.

Setting the Event Host Name

If the Diagnostics Server host has multiple network interfaces, and you want to specify the hostname that the Diagnostics Server will listen on, you need to set the **event.hostname** property.

This property can be found in:

`<diagnostics_server_install_dir>/etc/server.properties`

Uncomment the property, **event.hostname**, and specify the hostname value.

By default, the **event.hostname** property is not set. This means that the Diagnostics Server will listen on all hostnames.

Modifying the jetty.xml File

The **jetty.xml** file has a section which defines the interfaces on which the Diagnostics Server is permitted to listen. By default, the **jetty.xml** file included with the Diagnostics Server has no listeners defined and the Diagnostics Server listens on all of the interfaces.

To configure the Diagnostics Server to listen on specific network interfaces on a machine:

- 1 Open `<diagnostics_server_install_dir>/etc/jetty.xml` and locate the following line:

```
<Configure class="org.mortbay.jetty.Server">
```

- 2 Add the following block of code after this line, changing the `<Set name="Host">.....</Set>` to contain the NIC's IP address.

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Host">127.0.0.1</Set>
      <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
</Call>
```

- 3 Repeat the previous step adding a new copy of the block of code and setting the IP address for the NIC for each interface on which the Diagnostics Server is to listen.

Make sure that the `</Configure>` tag follows the listener code for the last NIC.

Note: Make sure that components that access the Diagnostics Server can resolve the hostnames of the Diagnostics Server to the IP address that you specify in the `jetty.xml` file for the host values. It is possible that some systems may resolve the host name to a different IP address on the Diagnostics Server host. For more information, see “Overriding the Default Diagnostics Server Host Name” on page 317.

Sample jetty.xml File

The following example shows the **jetty.xml** file for the Diagnostics Server, where the Diagnostics Server will listen on loopback and one IP address on the system.

```
<!-- Configure the Jetty Server -->
<!-- ===== -->
<Configure class="org.mortbay.jetty.Server">
<!--===== -->
<!-- Configure the Request Listeners -->
<!--===== -->
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Host">127.0.0.1</Set>
      <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
</Call>

<-Listen on specific IP Address on this machine for incoming Commander calls->
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Host">10.241.3.109</Set>
      <Set name="Port"><SystemProperty name="jetty.port" default="2006"/></Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
</Call>
</Configure>
```


Reducing Diagnostics Server Memory Usage

The Transaction Timeout Period is a safety mechanism that is used to prevent the Diagnostics Server from using excessive amounts of memory because it is holding on to old data for too long. The Diagnostics Server holds on to all of the information that it receives for a transaction until it receives the End of Transaction Notification (ELT), which tells the Diagnostics Server the transaction is complete. The timeout period for a transaction is reset each time the Diagnostics Server receives data for the transaction.

If the machine that the Diagnostics Server in Commander mode is running on is overloaded (CPU is heavily loaded or there are too many transactions per second for it to handle), or if there are network connectivity issues between the Load Generators or Mercury Business Availability Center and the Diagnostics Server in Commander mode, or between Business Process Monitor and Mercury Business Availability Center, the Diagnostics Server may not receive the ELT that lets it know when a transaction has ended. If the ELT is not received by the time the transaction timeout period expires, the Diagnostics Server assumes that the ELT is not coming and proceeds to process the data for the transaction and free up the memory that the transaction data is using.

The **correlation.txn.timeout** property sets the duration of the transaction timeout period. If you are experiencing out of memory conditions in the Diagnostics Server, you may want to reduce the transaction timeout period so that the Diagnostics Server waits less time for the end of a transaction. Use caution when adjusting the value of this property because multiple probes may be sending data to the Diagnostics Server, and a active transaction may be idle in one Diagnostics Server, so setting the value of this property too low could cause transactions to be reported incorrectly. It is recommended that if you need to reduce the value of this property, set it to 90 seconds more than the longest transaction in your test.

Configuring Fragment Name Based Trimming

Fragment name based trimming lets you configure Diagnostics to filter out server requests that appear to be causing Diagnostics Server performance issues without changing the configuration or the instrumentation in the probes.

Note: Fragment name based trimming is not intended to be used instead of the latency and depth trimming that you configure on the probes.

Using the **trim.fragment** properties in the `<diagnostics_server_install_dir>\etc\trimming.properties` file you can specify the names of the fragments that Diagnostics is to trim. Diagnostics trims the fragments for both Real User and Virtual User server requests.

By default, the properties **trim.fragment.1** and **trim.fragment.2** are commented out in **trimming.properties**. To specify a fragment to be trimmed, uncomment one of the properties and type the fragment name that is to be trimmed as it is listed in the Diagnostics views. If you have more than two fragments to be trimmed you can create additional **trim.fragment** properties being sure to increment the number at the end to ensure that each property name is unique. For example, the next **trim.fragment** property would be named **trim.fragment.3**.

Events and fragments that are trimmed as a result of these property settings are counted in the dropped event and dropped fragment counts.

Preparing a High Availability Diagnostics Server

If your Diagnostics deployment requires that the Diagnostics Server have a high availability, you can use the following instructions to create a standby Diagnostics Server for each Diagnostics Server. The standby is then ready to be used in the event of a hardware failure or some other problem with the host of the Diagnostics Server.

Creating a Standby Diagnostics Server

You can create a standby for each Diagnostics Server by installing the Diagnostics Server onto a standby machine and then periodically replicating the primary Diagnostics Server data into the standby Diagnostics Server.

To configure a standby Diagnostics Server:

- 1** Install the Diagnostics Server onto the standby machine. Make sure that the version of the Diagnostics Server to be installed on the standby server is the same as the Diagnostics Server on the primary server.
- 2** Schedule a periodic remote backup of the primary server into the standby server using the following commands from the host of the standby Diagnostics Server:

```
% cd /opt/MercuryDiagnosticsServer/
% ./bin/remote-backup.sh -h <primary_server_host> -o .
```

Replace `<primary_server_host>` with the host name for the Diagnostics Server that is being replicated.

These commands perform an incremental replication of the Diagnostic data, configuration files, customized views, alerts and comments onto the standby Diagnostics Server. You can schedule the periodic backup using a cron job or a scheduled task on Windows.

Note: The `wget` utility is used to download the backup over HTTP. For Windows, you must have an installation of `cygwin` on the host for the Diagnostics Server. You can get a copy of `cygwin` at <http://www.cygwin.com/>.

Failover to the Standby Diagnostics Server

If the host for the primary Diagnostics Server fails, you must configure the standby Diagnostics Server so that it can begin to function as the primary Diagnostics Server.

To make the standby Diagnostics Server the primary Diagnostics Server:

- 1** Change the hostname of the standby Diagnostics Server to match the hostname of the failed host of the primary Diagnostics Server. This allows the probes to reconnect to the Diagnostics Server when it is started.
- 2** Start the standby Diagnostics Server as a Windows Service, or by using the `bin/server.sh` or `bin\server.cmd` scripts.
- 3** The probes reconnect to the Diagnostics server. Whenever a probe loses its connection to its Diagnostics Server it attempts to reconnect approximately every 30 seconds.
- 4** The standby Diagnostics Server is now the primary Diagnostics Server. Configure a new standby Diagnostics Server as described in “Creating a Standby Diagnostics Server” on page 325.

Note: When the failed Diagnostics Server host is recovered, you should not make it the primary Diagnostics Server because you will lose any data that was gathered from the probes while the new primary Diagnostics Server was being used.

LoadRunner / Performance Center Diagnostics Server Assignments

Default Diagnostics Server Assignment

By default, a probe that is selected for a LoadRunner or Performance Center run will use the Diagnostics Server that is specified in its `<probe_install_directory>/etc/dynamic.properties`.

Diagnostics Server Assignment Override

It is possible to override the probe configuration when the probe is started for a run. To do so, you modify a mapping file on the Diagnostics Server in Commanding Mode, allowing you to override the Diagnostics Server assignment for a probe.

This can be useful when you are running Diagnostics in a combined LoadRunner / Performance Center and Mercury Business Availability Center environment. In such a case, you may want the probes to use different Diagnostic Servers when they are in a LoadRunner / Performance Center run than they do for Mercury Business Availability Center monitoring. You can do so by following these Diagnostics Server override instructions.

It may also be more convenient to use this mechanism than to edit the probe configuration file.

Note: When the probe is not in a run, it will always use the Diagnostics Server specified in its `<probe_install_dir>/etc/dynamic.properties` file.

To override the Diagnostics Server Assignment for a probe, modify the `server_assignment.properties` file in the `<diagnostics_server_install_dir>/etc` directory on the Diagnostics Server in Commander mode host machine.

The format of the **server_assignment.properties** file is:

<ProbeID> = <Server.id>

- ▶ Replace <ProbeID> with the ID of the probe.
- ▶ Replace <Server.id> with the ID of the Diagnostics Server.

The **server_assignment.properties** file is dynamically read at the start of each LoadRunner / Performance Center run. This means that changes made to this file become effective without having to restart the Diagnostics Server in Commander mode.

Configuring the Diagnostics Server for the LoadRunner Offline File

For each LoadRunner scenario or Performance Center test that is run, the Diagnostics Server in Mediator mode produces a file that is needed for LoadRunner Offline analysis which contains the J2EE data captured during the scenario. The size of this file can grow to be quite large. You must ensure that you have enough disk space to hold the LoadRunner Offline file on both the Diagnostics Server in Mediator mode host machine where the file is stored temporarily while the scenario is running and the Load Runner controller host machine where the file is stored when the scenario has ended.

Estimating the Size of the LoadRunner Offline File

Estimating the size of the offline file is highly dependent upon the data that is being captured and the rate at which the data is captured.

To estimate the size of the LoadRunner offline file:

- 1 Run a load test for 5 minutes and monitor the size of the offline file created by the Diagnostics Server in Mediator mode when Load Runner scenario is started.

Locate the offline file on the Diagnostics Server in Mediator mode host machine in <diagnostics_server_install_dir>/data/<newest directory>. The offline file has an extension of “.inuse”.

- 2 After 5 minutes note the size of the offline file.
- 3 Extrapolate the size of the offline file after one hour by multiplying the size of the offline file from the previous step by 12.
- 4 Determine the anticipated size of the offline file at the end of your load test by multiplying the one hour file size calculated in the previous step by the number of hours that you expect your actual load test to run.
- 5 Determine if the Diagnostics Server in Mediator mode host machine and the Controller host machine have enough disk space to accommodate the anticipated offline file size.

Reducing the Size of the LoadRunner Offline File

If you are concerned about the size of the offline file, you can reduce the file size by increasing the offline aggregation periods for the Diagnostics Server in Mediator mode. This will reduce the level of granularity in the offline data and thus reduce the size of the offline files.

The default settings for these properties are **5s** (5 seconds) which causes the Diagnostics Server in Mediator mode to aggregate all data into 5 second time slices. Increasing the value of these properties will make the offline file smaller because fewer data points are required to be stored when the aggregation period is longer. For example, increasing the offline aggregation period properties to **45s** should reduce the file size by roughly 50-75%.

Note: The impact on the size of the offline file size that will be achieved by adjusting the offline aggregation period is highly dependent upon the behavior of your application and the specifics of your load test.

Use the following steps modify the Diagnostics Server in Mediator mode offline aggregation period properties `bucket.lr.offline.duration` and `bucket.lr.offline.sr.duration` in `<diagnostics_server_install_dir>/etc/mediator.properties`.

To reduce the size of the offline files by increasing the Diagnostics Server in Mediator mode offline aggregation periods:

- 1** Ensure that the Diagnostics Server in Mediator mode is not participating in any active LoadRunner / Performance Center runs. This is necessary because the Diagnostics Server in Mediator mode must be restarted before the property changes described in the following steps will take effect.
- 2** Access the Mediator Configuration Page from your browser by navigating to the following URL:

`http://<diagnostics_server_hostname>:8081/configuration/Aggregation?level=60`
- 3** Increase the Offline VU Aggregation Period by increasing the setting for the **Load Runner / Performance Center Offline VU Aggregation Period** property. The value of this property must be a multiple of 5. For example set it to “45s”.
- 4** Increase the Offline Server Request Aggregation Period by increasing the value of the **Load Runner / Performance Center Offline Server Request Aggregation Period** property. The value of this property must be a multiple of 5. For example set it to “45s”.
- 5** Update the Diagnostics Server in Mediator mode with the revised property values by clicking **Submit** at the bottom of the page.

A message will appear at the top of the page to indicate that the changes were saved along with a reminder to restart the Diagnostics Server in Mediator mode. As a further reminder the **Restart Mediator** button is displayed.

For more information on updating property values from the Configuration Page and a screen image showing the command buttons, see “Configuring Diagnostics Using the Diagnostics Server Configuration Pages” on page 483.

To cause the configuration changes to take effect restart the Diagnostics Server in Mediator mode by clicking **Restart Mediator**.

Configuring Diagnostics Using the Diagnostics Server Configuration Pages

The Diagnostics Server Configuration pages provide a way for you to set the property values that control how the Diagnostics Server communicates with the other Diagnostics components, and how it processes the data that it receives from the probes.

Note: To ensure that you are entering valid property values you should use these pages to update the Diagnostics Server properties rather than editing the property files directly.

For information about viewing and modifying Diagnostics using the Diagnostics Server Configuration pages, see Appendix A, “Diagnostics Server Administration Page.”

19

Advanced .NET Probe Configuration

This chapter provides instructions for configuring the .NET Probe.

This chapter describes:	On page:
Enabling the Mercury Diagnostics Probe for .NET	334
Restarting IIS	334
Discovering ASP.NET Applications	336
.NET Probe Automatic Configuration for Discovered ASP.NET Applications	337
Enabling and Disabling Instrumentation for Applications	338
Customizing the Instrumentation for ASP.NET Applications	341
Discovering the Classes and Methods in an Application	344
Configuring Latency Trimming and Throttling	346
Configuring Depth Trimming	351
Configuring the .NET Probe for Light-Weight Memory Diagnostics	352
Disabling Logging	354
Overriding the Default Probe Host Machine Name	354
Authentication and Authorization for .NET Profilers in Standalone Mode	355

Enabling the Mercury Diagnostics Probe for .NET

The .NET Probe is enabled when it is installed. After you restart your Web server and a URL in your application has been accessed, the .NET Probe begins to gather performance information.

You may disable the .NET Probe so that it does not start and does not gather performance metrics.

To disable a .NET Probe:

Select **Start > Programs > Mercury Diagnostics .NET Probe > Disable Mercury .NET Probe**.

To enable a .NET Probe that was disabled:

Select **Start > Programs > Mercury Diagnostics .NET Probe > Enable Mercury .NET Probe**.

Note: Disabling the Probe only disables Probe metrics collector and the Probe processes. It does not disable the system metrics collector. The process of enabling or disabling system metrics is controlled through the standard Windows services manager.

Restarting IIS

After you have modified the instrumentation or configuration for the Probe, you must restart IIS to cause your changes to take effect.

To restart IIS from the command line or from the **Start > Run** menu, type **iisreset** and press ENTER.

This command restarts the Web publishing service but does not immediately start the Probe. The next time that a Web page in your application is requested, the Probe is started, your applications are instrumented, and the Probe registers with the Diagnostics Server in Commander mode.

Note: ASP.NET automatically restarts applications under various circumstances, including when it has detected that applications have been redeployed, or when applications are exceeding the configured resource thresholds.

When ASP.NET restarts an application that is being monitored by a .NET Probe, the Probe is deactivated and a new Probe is started. While this is occurring, there may be a period of overlap where there are multiple Probes simultaneously registered with the Diagnostics Server in Commander mode and connected to the Diagnostics Server in Mediator mode. This condition could cause LoadRunner / Performance Center and Mercury Business Availability Center to report errors during the application restart sequence.

Verifying that the Probe Is Connected

If you are running the Probe in a production environment (with Mercury Business Availability Center or Diagnostics Standalone), or in a test environment (with LoadRunner or Performance Center), you can verify that the Probe is connected using the following instructions.

To verify that the Probe is connected:

- 1** Open web browser and browse to http://<diagnostics_server_host>:<diagnostics_server_port>/registrar for the Diagnostics Server in Commander mode.
- 2** Select **View > Edit Registered Components**
- 3** Look for your application listed in the **Name** column.

Discovering ASP.NET Applications

The .NET Probe installer automatically discovers the ASP.NET applications that you may want to instrument. After you have installed the Probe, you can request that the Probe rescan your IIS configuration to catch any additions or changes.

Discovering ASP.NET Applications During Probe Installation

The .NET Probe installer detects ASP.NET applications on the machine where the Probe is installed. The .NET Probe installer discovers applications by inspecting the IIS configuration and looking for virtual directory entries that might refer to ASP.NET applications.

In some instances, the ASP.NET applications are installed in a manner that prevents them from being detected. An example of a situation that could cause the installer to fail to detect an ASP.NET application is when an ASP.NET application has been installed as a Web directory instead of a virtual directory.

Discovering ASP.NET Applications After Probe Installation

You can request a rescan the IIS configuration when you have modified an existing ASP.NET application deployment or installed new ASP.NET applications.

To request that the Probe rescan the IIS configuration and update the `probe_config.xml` file, select **Start > Mercury Diagnostics .NET Probe > Rescan ASP.NET Applications**.

.NET Probe Automatic Configuration for Discovered ASP.NET Applications

The .NET Probe installer configures the Probe to capture basic ASP/ADO workload for each of the ASP.NET applications detected. The Probe performs the following configuration steps:

- Creates an application-specific capture points file template.

The capture points file defines the instrumentation that controls the workload that the Probe captures for each application. You must modify the instrumentation in the capture points file to provide the instructions that allow the Probe to capture application-specific custom business logic.

- Creates an **appdomain** tag in the **probe_config.xml** file which is located in the **<probe_install_dir>/etc** directory. The attributes of the **appdomain** tag direct the behavior of the Probe. You may control the depth and detail of the application performance the Probe captures by setting the following attributes:

points	References the application-specific capture point file that was created for the detected application.
enabled	Set to false when the appdomain tag is created indicating that the application is not explicitly instrumented when the Probe is started. However, this attribute can be overridden by the enablealldomains attribute in the process tag.

Note: Diagnostics enables the instrumentation for all discovered applications by setting the **enablealldomains** attribute in the **process** tag to true.

For information on enabling and disabling instrumentation for applications see “Enabling and Disabling Instrumentation for Applications” on page 338. For information on restarting IIS see “Restarting IIS” on page 334.

Enabling and Disabling Instrumentation for Applications

When the .NET Probe is first installed, the instrumentation for all discovered applications is enabled by default. You control which applications have their instrumentation enabled or disabled using the attributes in the `probe_config.xml` file for the Probe.

Disabling Instrumentation for Applications

Disabling instrumentation for an application allows you to avoid the processing overhead and distracting information in the Diagnostics views for applications that are not relevant to the environment whose performance you want to monitor.

To disable the instrumentation for an application:

- 1** Set the `enablealldomains` attribute in the `process` tag to `false`. This allows the attributes of each `appdomain` tag to control the state of the instrumentation for each application.
- 2** Set the `enabled` attribute in the `appdomain` tag to `true` for each application that you wish to enable the instrumentation.
- 3** Set the `enabled` attribute in the `appdomain` tag to `false` for each application that is to have its instrumentation disabled.

The following example shows the instrumentation has been disabled for an application.

```
<process name="ASP.NET", <enablealldomains="false">
  <points file="ASP.NET.points" />
  <points file="ADO.points" />
  <appdomain name="your app name", enabled="false">
    <points file="your app.capture points" />
  </appdomain>
</process>
```


Enabling Instrumentation for An Application

Enabling instrumentation for an application allows the Probe to monitor the applications performance so that you can see the performance metrics for the application in the views of the Diagnostics and Profiler user interfaces.

To enable the instrumentation for an application:

- 1** Set the **enablealldomains** attribute in the **process** tag to **false**. This allows the attributes of each **appdomain** tag to control the state of the instrumentation for each application.
- 2** Set the **enabled** attribute in the **appdomain** tag to **true** for each application that you wish to enable the instrumentation.
- 3** Set the **enabled** attribute in the **appdomain** tag to **false** for each application that is to have its instrumentation disabled.

The following example shows the instrumentation has been enabled for an application and disabled for another.

```
<process name="ASP.NET", <enablealldomains="false">
  <points file="ASP.NET.points" />
  <points file="ADO.points" />
  <appdomain name="your app name", enabled="true">
    <points file="your app.capture points" />
  </appdomain>
  <appdomain name="YourAppTwo", enabled="false">
    <points file="YourAppTwo.points"/>
  </appdomain>
</process>
```

Enabling Instrumentation for All Applications

Enabling instrumentation for all application allows the Probe to monitor the performance of all detected applications so that you can see the performance metrics for all of the applications in the views of the Diagnostics and Profiler user interfaces.

To enable the instrumentation for all applications:

- 1** Set the **enable all domains** attribute in the **process** tag to true. This overrides the settings of the attributes in each **appdomain** tag so that the instrumentation can be enabled without going in an setting numerous attributes.

The following example shows the instrumentation has been enabled for all applications:

```
<process name="ASP.NET", <enablealldomains="true">
  <logging level=""/>
  <points file="ASP.NET.points"/>
  <appdomain name="YourAppOne", enabled="false">
    <points file="YourAppOne.points"/>
  </appdomain>
  <appdomain name="YourAppTwo", enabled="false">
    <points file="YourAppTwo.points"/>
  </appdomain>
</process>
```

Customizing the Instrumentation for ASP.NET Applications

When the .NET Probe is installed, the **ASP.NET.points** file is created with the standard instrumentation that the Probe applies to all ASP.NET processing on the monitored server.

You must create application-specific instrumentation points to capture performance metrics for the business logic that has been implemented through application-specific classes and methods. The application-specific instrumentation points must be stored in a custom capture points file that can be associated with the application using the attributes in the `<probe_install_dir>/etc/probe_config.xml` file. If your application was auto-detected during the installation or during a rescan of IIS, a custom capture points file was automatically created for the application at the same time.

Note: If you do not know the classes and methods in an application that you may want to monitor, you can use the Reflector tool that was installed with the Probe to analyze the .dll files in the application and discover the classes and methods. See “Discovering the Classes and Methods in an Application” on page 344 for instructions on using Reflector.

To let the .NET Probe know that you want the instrumentation points in a custom capture points file to apply to an application, you must update the **points** attribute of the **appdomain** element in the **probe_config.xml** file.

To associate a custom capture points file with an application:

- 1 Create a capture points file with the instrumentation for the application specific classes. To create a capture points file, copy an existing capture points file in the `<probe_install_dir>/etc` directory.

Note: If the application was auto-detected during the installation or during a rescan of IIS, a capture points file already exists for the application.

- 2 Customize the capture points file by adding instrumentation points so that the Probe captures custom business logic for your applications.

The following example illustrates how to modify the capture points file so that the Probe captures IBuySpy custom code:

```
[IBuySpy Callee]
class = !IBuySpy.*
method = !.*
signature =
scope =
ignoreScope =
layer = Custom.IBuySpy
```

For more information about instrumentation, see Chapter 14, “Instrumenting an Application”.

- 3 Update the configuration of the Probe in **probe_config.xml** to ensure that the modified capture points file is properly referenced.

Within the ASP.NET **<process>** tag add an **<appdomain>** tag for the application. Include the **<points>** tag with the **file** attribute and the **enabled** attribute.

```
<appdomain name="your app name">, enabled="true"
  <points file="your app.capture points"/>
</appdomain>
```

The example below illustrates this step. A custom capture points file has been created for the MSPetsShop application. The file has been named **MSPetShop.points**. The **<appdomain>** tag for the application, and the capture points file have been added to the ASP.NET **<process>** tag in the **probe_config.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<probeconfig>
  <id probeid="" probegroup="Umatilla"/>
  <credentials username="" password=""/>
  <profiler authenticate=""><authentication username="" password=""/></profiler>
  <diagnosticserver url="http://issaquah:2006"/>
  <mediator host="issaquah" port="2612"/>
  <webserver start="35000" end="35100"/>
  <modes am="true"/>
  <instrumentation><logging level="" threadids="no"/></instrumentation>
  <lwmd enabled="true" sample="1m" autobaseline="1h" growth="10" size="10"/>
  <process name="ASP.NET", enablealldomains="false">
    <logging level=""/>
    <points file="ASP.NET.points"/>
    <appdomain name="MSPetShop", enabled="true">
      <points file="MSPetShop.points"/>
    </appdomain>
  </process>
</probeconfig>
```

4 Restart IIS as instructed in “Restarting IIS” on page 334.

Discovering the Classes and Methods in an Application

If you want to monitor the performance of an application that you are not familiar with, you can use the Reflector automatic discovery tool that is installed with the .NET Probe to find the classes and methods in the application that you want to add to the Probe's instrumentation. The Reflector executable is located at `<probe_install_dir>\bin\reflector.exe`.

To discover classes and methods using Reflector:

- 1 Locate the installation directory for the application that you want to monitor.
- 2 Locate the folder in the application installation directory where the .dll files are stored.
- 3 Open a command prompt and change the directory to the folder where the .dll files for your application are stored.
- 4 Run the Reflector against all of the .dll files and .exe files in the current directory by executing the following the command at the command prompt:

```
<probe_install_dir>\bin\Reflector.exe
```

You can limit the Reflector to certain .dll and .exe files by adding additional parameters to the command. The following example shows another way to enter the command in the previous example:

```
<probe_install_dir>\bin\Reflector.exe *.dll *.exe
```

This command explicitly tells the Reflector to check all of the .dll and .exe files in the target directory.

To limit the Reflector to specific files, you could enter the following:

```
<probe_install_dir>\bin\Reflector.exe WorkHorse.dll Utility.dll
```

This command explicitly tells the Reflector to check only the two .dll files specified.

The following example shows the commands that you might execute if you have an application called PetShop that has .dll files located in a bin folder:

```
C:\>cd "c:\Program Files\Microsoft\PetShop\Web\bin"

C:\Program Files\Microsoft\PetShop\Web\bin>
C:\MercuryDiagnostics\".NET Probe"\bin\Reflector.exe
```

- 5** The Reflector displays a report of the assemblies, namespaces, classes, and methods found in the .dll files that you specified.

```
-----
Assemblies:
-----
C:\Program Files\Microsoft\PetShop\web\bin\PetShop.BLL.dll
C:\Program Files\Microsoft\PetShop\web\bin\PetShop.DAL.dll
C:\Program Files\Microsoft\PetShop\web\bin\PetShop.web.dll
-----

Namespaces:
-----
(8 classes) PetShop.BLL
(6 classes) PetShop.DALFactory
(17 classes) PetShop.web
(11 classes) PetShop.web.Controls
(2 classes) PetShop.web.ProcessFlow
(1 classes) PetShop.web.webComponents
-----

(8 classes) Namespace: PetShop.BLL
-----
PetShop.BLL.Account (10 Methods)
  Equals          System.Boolean(System.Object)
  Finalize        System.Void()
  GetAddress      PetShop.Model.AddressInfo(System.String)
  GetHashCode     System.Int32()
  GetType         System.Type()
  Insert          System.Void(PetShop.Model.AccountInfo)
  MemberwiseClone System.Object()
  SignIn          PetShop.Model.AccountInfo(System.String, System.String)
  ToString        System.String()
  Update          System.Void(PetShop.Model.AccountInfo)

PetShop.BLL.Cart (17 Methods)
  Add             System.Void(System.String)
  Equals          System.Boolean(System.Object)
  Finalize        System.Void()
  get_Count      System.Int32()
  get_Item       PetShop.Model.CartItemInfo(System.Int32)
  get_Total      System.Decimal()
  GetCartItems   System.Collections.ArrayList()
  GetEnumerator   System.Collections.IEnumerator()
  GetHashCode     System.Int32()
  GetInStock     System.Int32(System.String)
  GetOrderLineItems System.Collections.ArrayList()
  GetType         System.Type()
  MemberwiseClone System.Object()
```

Note: You can redirect the output from the Reflector to a file, as shown in the following example:

```
<probe_install_dir>\bin\Reflector.exe sys*.dll > <report_name>.txt
```

The output from Reflector is redirected to the file that you specify.

Use the information in the report to customize the instrumentation for your application, as described in “Customizing the Instrumentation for ASP.NET Applications” on page 341.

Configuring Latency Trimming and Throttling

When the .NET Probe determines that it is running out of resources because the Diagnostics Server is not keeping up with the amount of data that the Probe is capturing, it can automatically reduce the number of methods that it captures using a process called *latency trimming*. By default, latency trimming is enabled so that the Probe can adjust its work load as necessary.

When latency trimming is enabled, the .NET Probe trims the number of methods captured by ignoring methods with a latency below a certain minimum latency threshold. The idea behind trimming is that it is better to miss capturing methods with lower latency that are less likely to be of interest than to allow the Probe to bog down or stop running. Trimming allows the Probe to continue to run so that it can capture the more interesting methods with higher latencies.

Note: Because of threading and buffering, partial information about a method that was trimmed may be transmitted to the Diagnostics Server. When the Diagnostics Server detects that it received only partial information for a method, it issues a warning message. You should ignore these warning messages unless you expected that the information for all methods was to be captured.

Disabling Latency Trimming

By default, trimming is enabled for the .NET Probe. To disable trimming you must change the Probe configuration.

To disable Latency Trimming:

Add the **latency** tag to the `<probe_install_dir>/etc/probe_config.xml` configuration file, as shown in the following example:

```
<trim>
  <latency enabled="false" />
</trim>
```

The attribute of the latency element that turns on latency trimming is **enabled**. Latency trimming is enabled when **enabled** is set to true. When **enabled** attribute is set to false, latency trimming is disabled. The default value for this attribute is true.

For a description of attributes and elements of the **latency** element, see Chapter 20, “Understanding the .NET Probe Configuration File”.

Enabling Latency Trimming

By default, trimming is enabled for the .NET Probe. If you have subsequently disabled trimming, you must change the Probe configuration to enable it once more.

To enable Latency Trimming:

Change the value of the enabled attribute of the **latency** element in the `<probe_install_dir>/etc/probe_config.xml` configuration file, as shown in the following example:

```
<trim>
  <latency enabled="true" />
</trim>
```

The attribute of the latency element that turns on latency trimming is **enabled**. Latency trimming is enabled when **enabled** is set to **true**. When **enabled** attribute is set to **false**, latency trimming is disabled. The default value for this attribute is **false**.

For a description of attributes and elements of the **latency** element, see Chapter 20, “Understanding the .NET Probe Configuration File”.

Setting Latency Trimming Thresholds

By default, the latency trimming thresholds are set so that those methods with a latency less than 2 ms are trimmed, and those methods with a latency greater than 100 ms are never trimmed.

You can set the minimum trimming threshold by adjusting the value of the **min** attribute. You can set the maximum trimming threshold by adjusting the value of the **max** attribute. These attributes are specified in the **latency** element in the `<probe_install_dir>/etc/probe_config.xml` configuration file.

```
<trim>  
  <latency enabled="true" min="50" max="100" />  
</trim>
```

The attributes of the latency element that control the trimming thresholds are:

► **min**

Sets the minimum latency threshold. When latency trimming is enabled, methods with a latency less than or equal to the value of this attribute are trimmed. If you do not specify a value for this attribute, the default value of 2 ms is used.

The lower the value of the **min** attribute the greater the chance that the performance of your application will be adversely impacted. A lower value means that fewer methods are trimmed because more low-latency methods are captured.

If the information for all methods must be captured, disable latency trimming by setting **latency enabled** equal to **false**.

► **max**

Sets the maximum latency threshold. When latency trimming is enabled, methods with a latency greater than or equal to the value of this attribute are never be trimmed. The default value for this attribute, if you do not specify a value, is 100 ms.

For a description of the attributes and elements of the **latency** element, see Chapter 19, “Advanced .NET Probe Configuration”.

Configuring Latency Trimming Throttling

Latency trimming is throttled by default. When throttling is enabled, the amount of trimming that the Probe does is automatically adjusted based on the percentage of the Probe resources that are being used up by the Diagnostics Server processing backlog.

Without throttling, the methods that fall below the minimum method latency threshold are always trimmed.

If the percentage resources used by the Probe increases above a set throttling increment threshold, the effective trimming threshold is incremented so that methods with higher latency are trimmed. If the percentage of Probe resources used increases above the threshold again, the effective trimming threshold is incremented once more so that methods with even higher latency are trimmed. If the percentage of Probe resources used drops below the throttling decrement threshold, the effective trimming threshold is decremented so that the methods with lower latencies are captured once more.

The effective trimming threshold cannot be incremented above the maximum method latency threshold, and it cannot be decremented below the minimum method latency threshold.

Below is an example of the **latency** element in the **probe_config.xml** configuration file that includes the throttling attributes:

```
<trim>
  <latency enabled="true" min="50" max="100" >
    <throttle="true" incrementthreshold="75">
      <decrementthreshold="50" increment="2"/>
    </throttle>
  </latency>
</trim>
```

The attributes of the **latency** element that control throttling are:

► **throttle**

Throttling is enabled when this attribute is set to **true**. When this attribute is set to **false** throttling is disabled. The default value for this attribute is **true**.

► **increment**

Sets the amount that the effective trimming threshold is incremented when the percentage of Probe resources used exceeds the **incrementthreshold**. Sets the amount that the effective trimming threshold is decremented when the **decrementthreshold** is crossed. The default value for this attribute is 2 ms.

► **incrementthreshold**

When the percentage of Probe resource usage rises to the value of this attribute or higher, throttling is triggered so that the effective trimming threshold is incremented. The default value for this attribute is 75 percent.

► **decrementthreshold**

When the percentage of Probe resource usage falls to the value of this attribute or lower, throttling is triggered so that the effective trimming threshold is decremented. The default value for this attribute is 50 percent.

For a description of the attributes and elements of the **latency** element, see “Understanding the .NET Probe Configuration File” on page 357.

Configuring Depth Trimming

The .NET Probe can automatically reduce the number of methods that it captures using a process called *depth trimming*. When the Diagnostics Server is not keeping up with the amount of data that the Probe is capturing, the Probe can use depth trimming to help prevent it from running out of resources. By default, depth trimming is enabled.

When depth trimming is enabled, the .NET Probe trims the number of methods captured by ignoring methods that are called at a stack depth that is greater than the maximum stack depth threshold. Those that are called at a stack depth less than or equal to the stack depth threshold are captured. The idea behind trimming is that it is better to miss capturing methods further down in the call stack, that are less likely to be of interest, so that the Probe is able to continue to run and is able to capture the more interesting methods that occur higher in the call stack.

For example, if the stack depth threshold is 3, and the following method calls are made:

```
/login.do calls a() calls b() calls c()
```

only the /login.do, a, and b methods are captured, and method c is trimmed.

Below is an example of the **depth** element in the **probe_config.xml** configuration file that includes the trimming attributes:

```
<trim>
  <depth enabled="true" depth="10" />
</trim>
```

The attributes of the **depth** element that control trimming are:

► **enabled**

Depth trimming is enabled when this attribute is set to **true**. When this attribute is set to **false** depth trimming is disabled. The default value for this attribute is **true**.

► **depth**

Sets the threshold that are used for depth trimming. Methods that are called at or below the value of this attribute are trimmed when depth trimming has been enabled. The default value for this attribute is 25.

Setting **depth** to a lower value can significantly reduce the overhead of capture. For a description of the attributes and elements of the **depth** element, see “Understanding the .NET Probe Configuration File” on page 357.

Configuring the .NET Probe for Light-Weight Memory Diagnostics

The .NET Probe has three different levels of Light-Weight Memory Diagnostics (LWMD):

- LWMD Off (Do not capture heap or collections metrics)
- Capture heap metrics only
- Capture heap metrics and collections metrics

When the .NET Probe is installed, the default configuration for the .NET Probe is to have Light-Weight Memory Diagnostics (LWMD) turned on only for the capture of heap metrics.

If you want to enable LWMD to capture collection metrics, you must change the Probe’s configuration using the instructions below.

Note: Enabling the Probe to capture collections metrics may incur additional overhead on the host for your application.

To enable the capture of collection metrics:

- Add a **points** tag for the **Lwmd.points** file to either the **process** tag or to one or more **appdomain** tags in the **probe_config.xml** configuration file.

When you install the .NET Probe, the **Lwmd.points** file is installed in the `<probe_install_dir>/etc/` directory along with the **ASP.NET.points** and **ADO.points** files. The **Lwmd.points** file contains the instrumentation instructions needed to enable the capture of collection metrics.

If you add the **points** tag to the **process** tag the **probe_config.xml** configuration file should look similar to the following example:

```
<process name="ASP.NET", <enablealldomains="false">
  <points file="ASP.NET.points" />
  <points file="ADO.points" />
  <points file="Lwmd.points"/>
  <appdomain name="your app name", enabled="true">
    <points file="your app.capture points" />
  </appdomain>
</process>
```

If you add the **points** tag to the **appdomain** tags the **probe_config.xml** configuration file should look similar to the following example:

```
<process name="ASP.NET", <enablealldomains="false">
  <points file="ASP.NET.points" />
  <points file="ADO.points" />
  <appdomain name="your app name", enabled="true">
    <points file="your app.capture points" />
    <points file="Lwmd.points"/>
  </appdomain>
</process>
```

To disable LWMD:

- Delete the **points** tags for the **Lwmd.points** file from either the **process** tag or from the appropriate **appdomain** tags in the **probe_config.xml** configuration file.

Without the **points** tags in the configuration file, the Probe cannot locate the **Lwmd.points** file and so is missing the instruction that enables LWMD.

Disabling Logging

You can disable Probe application logging by changing the **logging level** tag of the ASP.NET process section of the **probe_config.xml** file, as shown in the following example:

```
<process name="ASP.NET">  
    <logging level="off"/>  
</process>
```

You can disable Probe instrumentation logging by changing the **logging level** tag of the instrumentation section, as shown in the following example:

```
<instrumentation>  
    <logging level="off" />  
</instrumentation>
```

Overriding the Default Probe Host Machine Name

The **registered_hostname** property allows you to override the default host machine name that the Probe uses to register itself with the Diagnostics Server in Commander mode. In situations where a firewall or NAT is in place or where your Probe host machine has been configured as a multi-homed device, it may not be possible for the Diagnostics Server in Commander mode to communicate with the Probe unless you override the default host machine name.

To override the default host machine name for a Probe, set the **registered_hostname** attribute, located in the .NET Probe **<registrar>** tag of the **probe_config.xml** file, to an alternate machine name or IP address that allows the Diagnostics Server in Commander mode to communicate with the Probe. Below is an example of the override:

```
<registrar url="http://foo:2006/registrar" registered_hostname="bar"/>
```

Note: Setting the `registered_hostname` attribute because of a NAT or firewall is only an issue for a test environment where you are using LoadRunner, Performance Center, or Diagnostics Standalone.

However, if you should set the `registered_hostname` in a production environment where you are using Mercury Business Availability Center or Diagnostics Standalone, the name that you specify is shown as the host name in System Health.

Authentication and Authorization for .NET Profilers in Standalone Mode

When you install the .NET Probe as a Profiler only (not connected to any Diagnostics Server), you manage the authentication and authorization of users of the Profiler in the `<probe_install_dir>/etc/probe_config.xml` file.

Note: If the .NET Probe is configured to work with a Diagnostics Server, the probe (Profiler) authorization and authentication settings are managed from the Diagnostics Server in Commander mode to which this probe is connected. For more information, see “User Authentication and Authorization” on page 489.

When you access the probe from the Diagnostics Server, the default username is **admin** and the default password is **admin**.

If the .NET Probe is installed as a profiler only, then by default, users are not required to enter a username and password to access the profiler.

However, you can configure the profiler to require user authentication. If you configure the profiler to require user authentication, you can define new usernames and passwords for accessing the profiler.

To configure the profiler to require user authentication:

Go to the `<probe_install_dir>/etc/probe_config.xml` file and set the value of **profiler authenticate** to **true**.

If you do not set a username and password, the default username is **admin** and the default password is **admin**.

To create new usernames and passwords for users of the standalone .NET Diagnostics Profiler:

- 1 Generate a new username and password using the **PassGen.exe** utility located in the `<probe_install_dir>/bin` directory.
- 2 In the `probe_install_dir>/etc/probe_config.xml` file, after the `<profiler authenticate="">` line, enter a username and password for each new user, in the following format:

```
<authentication username="" password=""/>
```

- For **authentication username**, enter the username that you chose.
- for **password**, enter the encoded string that was returned by the **PassGen.exe** utility.

Important: If you defined new usernames and passwords to access the profiler, you can no longer use the default username and password (**admin, admin**). Rather, you have to use one of the new usernames that you defined.

20

Understanding the .NET Probe Configuration File

This chapter provides instructions for configuring the .NET Probe.

This chapter describes:	On page:
About the .NET Probe Configuration File	357
Understanding the .NET Probe Configuration File	358

About the .NET Probe Configuration File

You control the configuration of the .NET Probe by modifying the elements and attributes in the .NET Probe configuration file:

`<probe_install_dir>/etc/probe_config.xml`. This file is defined in the file `<probe_install_dir>/etc/probe_config.xsd`. The sections that follow describe the elements and attributes that make up the configuration file, and how you can use them to configure your Probe.

Understanding the .NET Probe Configuration File

The following topics define the elements and attributes that make up the .NET Probe configuration file. Each element is defined by describing its purpose, attributes, and parent and children elements. Remember to check the `<probe_install_dir>/etc/probe_config.xsd` file if you have any questions about the information presented here.

<probeconfig> element

Purpose

Provides single containing root element for the probe configuration.

Attributes

None.

Elements

Number of Occurrences	1
Parent Elements	None
Child Elements	appdomain, bufferpool, credentials, diagnosticserver, eventserverhost, id, instrumentation, ipaddress, logging, lwmd, modes, points, process, profiler, sample, trim, webserver, topology

<id> element**Purpose**

Provides probe id and probe group id.

Attributes

Attribute	Valid Values	Default	Description
probeid	Letters, digits, underscore, dash, period	\$(AppDomain).NET	Names the probe used by LoadRunner / Performance Center and System Health
probegroup		Default	Defines the grouping for probes used for reporting metrics.

Elements

Number of Occurrences	1 per parent
Parent Elements	probeconfig
Child Elements	none

<credentials> element

Purpose

Supplies credentials that are used to validate for communication with the Diagnostics Server.

Attributes

Attributes	Valid Values	Default	Description
username		none	Used to validate with the Diagnostics Server
password		none	Used to validate with the Diagnostics Server
authenticate	true, false	true	Enables and disables authentication. This should be disabled for backward compatibility with releases prior to D4

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<authentication> element**Purpose**

List of authenticated user names and passwords.

Attributes

Attributes	Valid Values	Default	Description
username		admin	
password		admin	Passwords must be generated using the passgen utility in the <probe_install_dir>\bin directory

Elements

Number of Occurrences	zero to many
Parent Elements	profiler
Child Elements	none

<filter> element

Purpose

Filters out certain metrics that would skew the results or not be representative of the processing being monitored.

Attributes

Attributes	Valid Values	Default	Description
firstserverrequest	true, false	false	Enables/disables skipping the collection of metrics for the first time a particular server requests (URL) gets run after application startup.

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<profiler> element**Purpose**

Contains settings for the Profiler feature of the probe.

Attributes

Attributes	Valid Values	Default	Description
authenticate	true, false	none	Enables/Disables authentication of incoming Profiler connection requests
register	true, false	false	Tells the probe to register even if it is in Profiler only mode
samples	number	60	Tells the Profiler how many samples to keep for lwmd/heap trending
best	number	1	The number of fastest instance trees to keeps
worst	number	3	The number of slowest instance trees to keep
inactivitytime out	string	10m	The length of time that the Profiler continues to run after the user has stopped interacting with the Profiler.

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	authentication

<modes> element

Purpose

Tells the probe which mode(s) to run in.

Attributes

Attributes	Valid Values	Default	Description
enterprise	true false	Depends on mode chosen in installation. <ul style="list-style-type: none"> ➤ true if pro is false ➤ false if pro is true 	Probe in Enterprise mode (probe is working with Diagnostics Server)
ent	true false	➤ false	This is a short form of the enterprise attribute
ad	true false	➤ false	
am	true false	➤ false	
pro	true false	Depends on mode chosen in installation. <ul style="list-style-type: none"> ➤ true if enterprise is false ➤ false if enterprise is true 	Probe in Profiler mode
auto	auto	When pro is true, auto is set to auto.	

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<diagnosticsserver> element**Purpose**

Contains connection and settings information related to the Diagnostics Server which are used for enterprise mode.

Attributes

Attributes	Valid Values	Default	Description
url	Registrar URL. http:// <host>: <port>	none	URL to connect to registrar
delay	number	2	Number of seconds to wait before registering
keepalive	number	15	Number of seconds between keepalives
proxy	URL of proxy	none	Registrar connection proxy.
proxyuser	user id for proxy	none	
proxypassword	password for proxy	none	
registeredhost name	string	none	Name of host to register as (external name for firewall traversing)
register_byip	true, false	false	Register using ipaddress instead of hostname

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<mediator> element

Purpose

Specifies the diagnostics server that is in the Mediator mode to which events are to be sent when in the enterprise mode.

Attributes

Attributes	Valid Values	Default	Description
host	host name	none	Name of mediator
port	number	2612	Mediator port
ssl	true/false	false	When the Diagnostics Server URL starts with http the default is false. When the Diagnostics URL starts with https the default is true.
metricport	number	2006	The port to which the Probe sends the Probe metrics such as heap usage and availability.
block	true/false	false	Block until mediator connection established
ipaddress			local ipaddress to use when connecting to the eventserver.

Attributes	Valid Values	Default	Description
localportstart	number	4000	Beginning of port range to use for tcp event channel connection to the Diagnostics Server in Mediator mode. Used only when ipaddress is specified.
localportend	number	5000	End of port range to use for tcp event channel connection to the Diagnostics Server in Mediator mode. Used only when ipaddress is specified.

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<webservice> element

Purpose

Specifies the local Web server properties for communication with the probe.

Attributes

Attributes	Valid Values	Default	Description
start	number	35000	Starting port for webservice
end	number	35100	Ending port for webservice
ipaddress			local ipaddress to run webservice on.

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<bufferpool> element**Purpose**

Configures the bufferpool behavior.

Attributes

Attributes	Valid Values	Default	Description
size	number	65536	Size of each buffer
buffers	number	512	Number of buffers in pool
sleep	number	1000	Number of milliseconds between flush checks
expires	number	1000	Number of milliseconds before buffer expires

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<lwmd> element**Purpose**

Configures the light-weight memory diagnostics feature.

Attributes

Attributes	Valid Values	Default	Description
enabled	true false	false	Enables lwmd capturing
sample	string	1m	Sample interval (h-hour/m-minute/ s-second)
autobaseline	string	1h	Auto baseline interval
manualbase line	string	none	Manual baseline time
growth	number	15	Number of collections to growth track
size	number	15	Number of collections to size track
include	string	none	Comma separated list of collection classes to include to the exclusion of all others.
exclude	string	none	Comma separated list of collection classes to exclude

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	exclude, include

<logdirmgr> element**Purpose**

Contains the configuration for the log directory manager. The logdirmgr monitors the log directory to ensure that it does not grow unbounded. The logdirmgr scans the logs periodically as indicated by the scaninterval. If the size has exceeded the size indicated by maxdirsize the logdirmgr deletes the oldest files until the size no longer is greater than the maxdirsize.

Attributes

Attributes	Valid Values	Default	Description
enabled	true false	true	
maxdirsize	number	1024 MB	Largest size that you want to be the limit of the size of the log directory.
scaninterval	number	30m	How often the manager scans the logs to check for growth and size.

Elements

Number of Occurrences	1 per parent
Parent Elements	probeconfig
Child Elements	none

<instrumentation> element

Purpose

Contains logging configuration for instrumenter.

Attributes

None.

Elements

Number of Occurrences	1 per parent
Parent Elements	probeconfig, process
Child Elements	logging

<logging> element**Purpose**

Sets the logging level for the Probe instrumentation processing.

Attributes

Attributes	Valid Values	Default	Description
level	off assert break severe warning info debug points eh sig chi classmap ilasm symbol deepmode load all	off	
threadids	true false	true	

Elements

Number of Occurrences	
Parent Elements	probeconfig, process
Child Elements	none

<managedLogging> element**Purpose**

Sets the logging level for the Probe processing for monitoring and reporting application performance.

Attributes

Attributes	Valid Values	Default	Description
level	off severe warning info debug events property webserver http symbols probemetrics registrar threadpool authentication	off	

Elements

Number of Occurrences	
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<points> element**Purpose**

List a capture points file for inclusion in instrumentation.

Attributes

Attributes	Valid Values	Default	Description
file	string	none	Name of instrumentation capture points file

Elements

Number of Occurrences	zero or more
Parent Elements	appdomain, process
Child Elements	none

<sample> element

Purpose

Sets the sampling type and rate.

Attributes

Attributes	Valid Values	Default	Description
method	percent, count, period	percent	Sets the sampling method <ul style="list-style-type: none"> ▶ for percent rate must be 0-100 ▶ for count rate must be >1 ▶ for period rate must be one of standard Diagnostics time strings (3m for 3 minutes, 4s for 4 seconds, and so forth)
rate	number	0	Sets the sampling rate for percent type

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<symbols> element**Purpose**

Limits the number of unique URIs and SQL strings that can be captured to control the amount of memory consumed.

Attributes

Attributes	Valid Values	Default	Description
maxuri	number	1000	Sets the top limit for number of unique URIs that can be captured.
maxuriname	string	Maximum number of unique URIs exceeded	
maxsql	number	1000	Sets the top limit for number of unique URIs that can be captured.
maxsqlname	string	Maximum number of unique SQLs exceeded	

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	none

<trim> element

Purpose

Configures the trimming feature to reduce data volume.

Attributes

None.

Elements

Number of Occurrences	1 per parent
Parent Elements	appdomain, probeconfig, process
Child Elements	depth, latency

<depth> element

Purpose

Configures depth trimming.

Attributes

Attributes	Valid Values	Default	Description
enabled	true false	true	Enables depth trimming
depth	number	25	Sets the depth for depth trimming

Elements

Number of Occurrences	1
Parent Elements	trim
Child Elements	none

<latency> element**Purpose**

Configures latency trimming.

Attributes

Attributes	Valid Values	Default	Description
enabled	true false	true	Enables latency trimming
throttle	true false	true	Enables latency trimming throttling
min	number	2	Minimum latency threshold.
max	number	100	Maximum latency threshold
increment	number	2	Threshold increment
increment threshold	number	75	The percentage of the buffer usage before the throttling should be incremented
decrement threshold	number	50	The percentage of the buffer usage before the throttling should be decremented

Elements

Number of Occurrences	1
Parent Elements	trim
Child Elements	none

<cputime> element**Purpose**

Controls the **cputime** setting property.

Attributes

Attributes	Valid Values	Default	Description
mode	none, serverrequest, method	serverrequest	

Elements

Number of Occurrences	1
Parent Elements	<probeconfig>, <process>, or <appdomain>
Child Elements	none

<topology> element**Purpose**

Controls the **topology** setting property.

Attributes

Attributes	Valid Values	Default	Description
enable	true false	true	Enables gathering topology information and passing it to the Diagnostics Server.

Elements

Number of Occurrences	1
Parent Elements	<probeconfig>, <process>, or <appdomain>
Child Elements	none

<rum> element

Purpose

Controls the settings (enabled/header) for Real User Monitoring.

Attributes

Attributes	Valid Values	Default	Description
enable	true false	false	Enables Real User Monitoring
header	""	"X-Mercury-Diag-Response-Color"	The value for header can be any string but the string must be configured the same for all components

Elements

Number of Occurrences	1
Parent Elements	<probeconfig>, <process>, or <appdomain>
Child Elements	none

<xvm> element**Purpose**

Controls the cross vm settings.

Attributes

None.

Elements

Number of Occurrences	1
Parent Elements	<probeconfig>, <process>, or <appdomain>
Child Elements	<ws>

<ws> element**Purpose**

Controls Web services correlation sampling.

Attributes

None.

Elements

Number of Occurrences	1
Parent Elements	<xvm> element
Child Elements	<sample>

<appdomain> element

Purpose

Builds an appdomain inclusion list for processes that host multiple appdomains.

The value of none means that it applies to all.

Attributes

Attributes	Valid Values	Default	Description
enabled	true false	true	Permits the appdomain to be instrumented.
name	string	none	Name that the .NET AppDomain settings apply to

Elements

Number of Occurrences	zero or more
Parent Elements	process
Child Elements	bufferpool, credentials, diagnosticserver, mediator, id, ipaddress, logging, lwmd, modes, points, profiler, sample, trim, webserver, symbols, filter, topology

<process> element**Purpose**

Provide an inclusion filter list of which applies to the probe.

None means none.

Attributes

Attributes	Valid Values	Default	Description
enablealldomains	true false	false	When set to true the enable attribute on all appdomains that are part of the process is overridden so that all will be enabled.
name	string	none	Name of the .NET process that these setting apply to
monitorheap	string	none	Enables the Heap tab and Heap Breakdown processing when the value is set to true.

Elements

Number of Occurrences	zero or more
Parent Elements	probeconfig
Child Elements	appdomain, bufferpool, credentials, diagnosticserver, mediator, id, instrumentation, ipaddress, logging, lwmd, modes, points, profiler, sample, trim, webserver, filter, symbols, topology

21

Advanced J2EE Probe and Application Server Configuration

This chapter provides instructions for configuring the J2EE Probe and the application server using the properties and settings provided for situations that require more advanced configurations.

This chapter describes:	On page:
Advanced Configuration Directory	388
Configuring the Probe to Work with Mercury Products	389
Configuring the Probes for Multiple Application Server JVM Instances	393
Disabling the J2EE Diagnostics Profiler	397
Specifying Probe Properties as Java System Properties	398
Controlling Probe Logging	399
Setting the Probe Host Machine Name	400
Controlling Automatic Method Trimming on the Probe	402
Controlling Probe Throttling	404
Configuring a Probe for a Proxy Server	407
Configuring Reverse HTTP for a Probe in MMS	407
Diagnostics Probe Administration Page	409
Authentication and Authorization for J2EE Profilers in Standalone Mode	412
Using BEA JRockit Mission Control	414

Advanced Configuration Directory

The following bullet points will help guide you to the probe configuration topic that will enable you to configure the probe to address particular issues in your environment.

- ▶ If you have a probe that was installed to work with one Mercury product and you now would like to work with other products, see “Configuring the Probe to Work with Mercury Products” on page 389.
- ▶ If your application server is using multiple JVMs or if you want to capture data for multiple JVMs you must perform additional probe configuration steps described in “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.
- ▶ If you have a probe that you want to prevent others from using in Profiler mode, see “Disabling the J2EE Diagnostics Profiler” on page 397.
- ▶ If you have more than one JVM using a single probe installation, you may need to set some of the probe properties in the Java system properties in the application start up commands. For more information, see “Specifying Probe Properties as Java System Properties” on page 398.
- ▶ If you want to have log messages posted to the probe logs for lower level messages, you can adjust the log level as described in “Controlling Probe Logging” on page 399.
- ▶ If you have more than one probe installed on the same host you can make sure that the log messages for each probe are stored in a different file as explained in “Changing the Log Directory for a Probe” on page 400.
- ▶ If you want to examine the performance of processing that would normally be trimmed from the metrics reported in Diagnostics you can reduce the level of trimming, or turn off trimming completely as described in “Controlling Automatic Method Trimming on the Probe” on page 402.
- ▶ In some situations, you may want to turn off probe throttling so that you can be sure that you are getting all of the available performance information without regard to the performance impact on the application itself. For information on throttling and how to control it see, “Controlling Probe Throttling” on page 404.

- ▶ If there is a proxy present between the probe and the Diagnostics Server in Commander mode you must set the correct property to tell the probe the URL of the Diagnostics Server in Commander mode. For more information, see “Configuring a Probe for a Proxy Server” on page 407.
- ▶ If you have installed a J2EE Probe that is to be used in a Mercury Managed Services (MMS) environment you are encouraged to disable the reverse http (rhttp) communication between the probe and the Diagnostics Server in Mediator mode. For more information, see “Configuring Reverse HTTP for a Probe in MMS” on page 407.

Configuring the Probe to Work with Mercury Products

Adding a Mercury Product to the Probe Configuration

The J2EE Probe is a lifecycle probe that can be configured to monitor your applications from development through implementation and production. The J2EE Probe can be configured to work with several Mercury products or to work as a Diagnostics Profiler without other Diagnostics components or Mercury products.

The products with which the J2EE Probe can work is determined by the value of the **active.products** property located in the property file `<probe_install_dir>/etc/probe.properties`.

The value of the **active.products** property is set at the time that you install the J2EE Probe. For information on probe installation see Chapter 6, “Installing the Mercury Diagnostics Probe for J2EE.”. To enable the probe to capture data for different Mercury products you must set the value of the **active.products** property by editing the property file and restarting the application server.

Note: If you want to use the J2EE Probe that was downloaded with the Mercury Diagnostics Profiler for J2EE with other Mercury products, contact Mercury Customer Support. The instructions that follow will not work for a probe installed for the J2EE Diagnostics Profiler without intervention from Mercury Customer Support.

To see Diagnostics data in the user interface of the interfacing Mercury Products there are additional configuration steps that must be completed. See the chapters in this guide that correspond to the Mercury product that you are using to complete the configuration.

The sections that follow provide instructions for configuring each product mode.

PRO Product Mode – Mercury Diagnostics Profiler for J2EE

For Mercury Diagnostics Profiler for J2EE, the product mode is PRO. In this mode, the probe gathers additional metrics and presents those metrics on a user interface that it makes available through a URL on the probe host.

If you are running the J2EE Probe as part of the J2EE Diagnostics Profiler in Development Mode, there are restrictions placed upon the probe to limit the load that the probe can handle.

If you are running the J2EE Probe as part of Diagnostics Standalone, or along with another Mercury product, the Profiler is enabled without the load restrictions.

Enterprise Product Mode

When configured in Enterprise mode, the probe works with the Mercury products such as Mercury Business Availability Center, LoadRunner, Performance Center, and as Mercury Diagnostics Standalone.

To configure the probe for Enterprise mode:

- 1** Set the **active.products** property to Enterprise and restart the Application Server.
- 2** Configure the probe to register with the Diagnostics Server in Commander mode.

One of the functions of the Diagnostics Server in Commander mode is to keep track of the Diagnostics components so that it can facilitate communication between them and keep you informed about the status and health of the components.

To configure the probe to register with the Diagnostics Server, set the host name and port using the **registrar.url** property which can be found in the property file: `<probe_install_dir>\etc\dispatcher.properties`.

Below is an excerpt from **dispatcher.properties** showing the **registrar.url** property:

```
## the URL of the registrar
registrar.url=http://host01.company.com:2006/registrar/
```

- 3** Configure the probe to connect to the Diagnostics Server in Mediator mode.

The probe must be able to transmit the processing metrics that it gathers to the Diagnostics Server in Mediator mode in order for Mercury Business Availability Center, LoadRunner, and Performance Center to be able to receive, process, and display the reports. In LoadRunner and Performance Center, Diagnostics assigns a Diagnostics Server to the probe. In Mercury Business Availability Center, you must tell the probe which Diagnostics Server to work with.

To configure the probe to communicate with the Diagnostics Server in Mediator mode, set the host name and the port using the **mediator.host.name** and **mediator.port.number** properties which can be found in `<probe_install_dir>\etc\dynamic.properties`.

The default port for the Diagnostics Server in Mediator mode is 2612. If, when installing the Diagnostics Server in Mediator mode, you set the port for the Diagnostics Server in Mediator mode to a value other than the default, make sure to use that same port number when you set the **mediator.port.number** property for the probe.

Below is an excerpt from the **dynamic.properties** file showing these properties:

```
#the host the Topaz mediator is running on
mediator.host.name=host01.company.com

#the port the Topaz mediator is listening on (default is 2612)
mediator.port.number=2612
```

AM Product Mode

To protect a probe in a production Mercury Business Availability Center deployment from accidentally being included in a LoadRunner or Performance Center run, set **active.products** to AM. When in AM mode the probe is not listed as an available probe in LoadRunner or Performance Center.

AD Product Mode

To prevent a probe in a QA environment from using additional resources and continually reporting data to the Diagnostics console dataset when a load test is not running set **active.products** to AD.

Removing a Product from the Probe Configuration

The product mode for the probe is configured using the **active.products** property located in the file `<probe_install_dir>\etc\probe.properties`.

To configure the probe so that it will no longer support a product, update the **active.products** property so that the product is no longer included in the property value.

Configuring the Probes for Multiple Application Server JVM Instances

When your application server is using multiple JVMs, or when you want to capture data for multiple JVMs, you must perform additional probe configuration steps. There are two options; you may install one probe for each JVM on a host, or you may install one probe that will be shared by all of the JVMs.

Configuring Multiple JVMs to Use a Single Probe Installation

To allow multiple JVMs to share a single probe installation, you must configure a separate instance of the probe for each JVM. This configuration enables the following:

- ▶ Establishment of communication between the JVM and the probe
- ▶ Identification of the probe by the JVM
- ▶ Instrumentation of the JVM to instruct the probe about the performance metrics it will monitor

To configure a J2EE Probe to work with multiple JVMs:

- 1** When a probe is being used to monitor multiple JVM versions, the JRE Instrumenter must be run once for each JVM version to enable the probe to monitor the events of the applications that are running on the JVMs.

If you have not already run the JRE Instrumenter for each of the JVM versions that the probe will be working with, do so now. See “Running the JRE Instrumenter” on page 155.

Note: You must run the JRE Instrumenter even if you let the installer instrument the JRE when the probe was installed. The JRE Instrumenter prepares the applications for multi-JVM support.

Note: If you are using multiple JVM versions and have run the JRE Instrumenter multiple times, be sure to include two paths in the `-Xbootclasspath` parameter exactly as indicated in the output from the Instrumenter. The first path is for the specific JVM and the second path for the default “boot” directory. For example:
`-Xbootclasspath/p:/path/to/javaprobe/classes/IBM/1.3.1:/path/to/javaprobe/classes/boot`

- 2** Specify the range of ports from which the probe can automatically select. The J2EE Probe communicates using the mini Web server. A separate port is assigned for probe communications for each JVM that a probe is monitoring. By default, the port number range is set to 35000 - 35100. You must increase the port number range when the probe is working with more than 100 JVMs.

Note: If a firewall separates the probe from the other Diagnostics components, you must configure the firewall to allow communications using the ports in the range that you specify. For more information, see Appendix , “Configuring Diagnostics Components to Work with a Firewall.”

If you have configured the firewall to allow probe communications on a range of ports that is different than the default, make sure to update the port range values discussed in the following bullets accordingly.

- a** Locate the `webserver.properties` file in the folder `<probe_install_dir>/javaprobe/etc`.
- b** Set the following properties to adjust the range of ports that are available for probe communications.
 - The minimum port in the port number range is set using the following property:
`jetty.port=35000`

- The maximum port in the port number range is set using the following property:

```
jetty.max.port=35100
```

- 3 Assign a custom probe Identifier to the probe for each JVM, using the Java command line.

```
-Dprobe.id=<Unique_Probe_Name>
```

The probe ids defined on the Java command line override the probe names that are defined in the **probe.properties** file using the probe's **id** property.

The following is an example of a WebLogic startup script before adding the **probe.id** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m -classpath "%CLASSPATH%"
-Dweblogic.Domain=petstore -Dweblogic.Name=petstoreServer -Dbea.home="C:\bea"
-Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

The following is an example of a WebLogic startup script after adding the **probe.id** parameter:

```
"%JAVA_HOME%\bin\java" -hotspot -ms64m -mx64m"
-Xbootclasspath/p:C:\Program Files\Mercury Interactive\Diagnostics for
J2EE\Probe\classes\Sun\1.4.1_03;C:\Program Files\
Mercury Interactive\Diagnostics for J2EE\Probe\classes\boot"
-classpath "%CLASSPATH%"
-Dprobe.id=<Unique_Probe_Name> -Dweblogic.Domain=petstore
-Dweblogic.Name=petstoreServer
-Dbea.home="C:\bea" -Dweblogic.management.password=%WLS_PW%
-Dweblogic.ProductionModeEnabled=%STARTMODE%
-Dcloudscape.system.home=./samples/eval/cloudscape/data
-Djava.security.policy=="C:\bea\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

Note: The command-line properties must be entered on one line, without any breaks.

- 4 Specify the points file that the probe will use. By default LoadRunner/Performance Center will set the `points.file.name` to `auto_detect.points`. You may want to specify that a custom points file be used when you need to use more than one custom instrumentation plan, or where you have several JVMs on the same machine using a single probe instance, and one or more of the JVMs needs specific methods and classes included in a layer support custom instrumentation.

```
-Dprobe.points.file.name="<Custom_AutoDetect_Points_File>"
```

Configuring Separate Probe Installations For Each JVM

When there are multiple JVMs on a single host, you may install a separate probe for each JVM instance. To use a separate probe for each JVM, you must install the probe multiple times, and define an instance of each probe by setting the probe's `id` property in the `probe.properties` file in each probe's installation directory.

To define an instance of each installed probe:

- 1 If you have not already manually run the JRE Instrumenter for the JVMs that the probe will be working with, do so now. To run the JRE Instrumenter, see "Running the JRE Instrumenter" on page 155.
- 2 Locate the `probe.properties` file in the `<probe_install_dir>/javaprobe/etc` directory.

For example:

```
C:\Program Files\Mercury Interactive\common\JavaProbe\etc\probe.properties
```

- 3 Assign a name to the **id** property that will be unique on the server and on the Diagnostics Server as follows:

id=<uniqueProbeName>

When the probe instance is started, a log file is created in the <**probe_install_dir**>/javaprobe/log directory where the log messages for the probe are stored.

Note: This configuration enables you to perform the diagnostics; but you will have to spend some time configuring each probe installation. Using a single probe for multiple JVMs can provide you with the same diagnostics detail, while allowing you to spend less time configuring the components.

Disabling the J2EE Diagnostics Profiler

You can disable the Mercury Diagnostics Profiler for J2EE on a J2EE Probe so that it cannot be accessed accidentally. When the J2EE Diagnostics Profiler is disabled, the user interface cannot be accessed from the J2EE Diagnostics Profiler URL: http://<probe_host>:<probeport>/profiler.

To disable the J2EE Diagnostics Profiler, set the **disable.profiler** property in <**probe_install_dir**>/etc/probe.properties to true.

The default value for **disable.profiler** is false. To enable the J2EE Diagnostics Profiler once it has been disabled, change the value of the **disable.profiler** property from true to false.

Specifying Probe Properties as Java System Properties

All of the J2EE Probe properties, except for those defined in the **dynamic.properties** property file, can be specified as Java System properties on the startup command-line for the application server. This is very useful when there is more than one JVM using a single probe installation.

To specify a probe property as a Java System property, prepend the letter **D** and the first part of the properties file name to the property name. This is best explained using some examples:

- To set the **id** property in **probe.properties** from the startup command, you concatenate the **D** and **probe** from the property file name, and then tack on the name of the property that you are specifying, that is, **id**, as follows:

```
-Dprobe.id=SomeId
```

- To set the **active.products** property in **probe.properties** from the startup command, you concatenate the **D** and **probe** from the property file name, and then tack on the name of the property that you are specifying, that is, **active.products**, as follows:

```
-Dprobe.active.products=AD,AM
```

- To set the **registrar.url** property in **dispatcher.properties** from the startup command, you concatenate the **D** and **dispatcher** from the property file name, and then tack on the name of the property that you are specifying, that is, **registrar.url**, as follows:

```
-Ddispatcher.registrar.url=http://host01.company.com:2006/registrar
```

Controlling Probe Logging

You can control the level of the messages that the probe logs and change the location where the log messages are posted using the probe properties.

Controlling the Log Message Level

The level of messages from the probe that are logged to the standard output is controlled by the **lowest_printing_level** property in the property file `<probe_install_dir>/etc/logging.properties`. The default setting for this property is OFF which prevents almost all probe messages from being logged to the console.

You can adjust the logging level dynamically by changing the value assigned to the **lowest_printing_level** property. The level of messages that are logged changes as soon as you save the property file.

The valid values for the **lowest_printing_level** property are:

Property Value	Description
OFF	No messages are logged.
DEBUG	All messages are logged.
INFO	Info, Severe, and Warning messages are logged.
WARN	Warning and Severe messages are logged.
SEVERE	Severe messages are logged.

Changing the Log Directory for a Probe

The default location for the log directory for a probe is `<probe_install_dir>/log`. When you have more than one probe on the same host, you can change the location of the log directory for each probe using the `log.dir` property. This property can be set in two ways:

- ▶ The value of the `log.dir` property can be set in the property file `<probe_install_dir>/etc/probe.property`.
- ▶ The value of the `log.dir` property can be specified on the startup command-line for the application server as a JAVA system property as shown in the following example:

```
-Dprobe.log.dir=/path/to/log
```

For more information on specifying the `log.dir` property on the startup command line, see “Configuring a Probe for a Proxy Server” on page 407.

Setting the Probe Host Machine Name

The probe’s host name is used to register the probe with the Diagnostics Server in Commander mode. The Diagnostics Server in Commander mode uses the probe’s host name to communicate with the probe and displays it along with the system metrics for the server that the probe is monitoring in the Diagnostics views.

The probe normally can detect the host name of the machine that is its host. In some situations, the server configuration is faulty and the probe cannot detect the correct host name. In situations where a firewall or NAT is in place or where your probe host machine has been configured as a multi-homed device, it may not be possible for the probe to properly detect its host.

If the probe cannot detect its host name, you can instruct the probe to get the host name via a reverse DNS lookup based on the socket connection or you can specify the host name using a probe property.

Instructing the Probe to Use Reverse DNS Lookup

If the configuration of the probe's host prevents the probe from detecting the host name, you can instruct the probe to detect the host name using a reverse-DNS lookup by setting the `server.host.name.lookup` property. This property is located in the `<probe_install_dir>/etc/dispatcher.properties` file.

The default value for the `server.host.name.lookup` property is 'false'. This tells the probe to do the lookup without using reverse-DNS. Set this property to 'true' to instruct the probe to use reverse-DNS lookup.

Manually Specifying the Probe Host Name

The `registered_hostname` property allows you to manually set host machine name for the probe and stop the probe from doing the automatic lookup.

To set default host machine name for a probe, set the `registered_hostname` property, located in the J2EE Probe property file, `<probe_install_dir>/etc/dispatcher.properties`, to a machine name or IP address.

When you set the `registered_hostname` property the automatic lookup of the host name is disabled.

Note: Setting the `registered_hostname` property because of a NAT or firewall is only an issue for a test environment where you are using LoadRunner, Performance Center, or Diagnostics Standalone.

However, if you should set the `registered_hostname` in a production environment where you are using Mercury Business Availability Center or Diagnostics Standalone, the name that you specify will be shown as the host name in System Health.

Controlling Automatic Method Trimming on the Probe

The default configuration for the probe includes settings that control the trimming of methods. Trimming can be controlled based upon how long the method takes to execute, which is known as *latency*, and by the *stack depth* of the method call. The default configuration instructs the probe to trim both by latency and by depth.

You may want to reduce the level of trimming, or turn off trimming completely, for certain Diagnostics situations. You can control the trimming using the **minimum.method.latency** and **maximum.stack.depth** properties in `<probe_install_dir>/etc/capture.properties`.

Controlling Latency Trimming

Methods that complete with latency greater than or equal to the value of the **minimum.method.latency** property are captured, and those that complete with latency less than this limit are trimmed to avoid incurring the overhead for methods that are less likely to be of interest.

Note: There are a couple of situations where a method latency is not trimmed when its latency is less than the trimming property.

- ▶ Methods that are the root for a call tree are not trimmed.
- ▶ Methods that threw an exception are not trimmed, unless the probe is facing a severe throttling situation due to an inability to send events to the Diagnostics Server in Mediator mode.

Because of threading and buffering behavior, partial information about a method that was trimmed may be saved in the Profiler buffers (Development mode) or transmitted to the Diagnostics Server (Production and Test mode). When the Diagnostics Server detects that it only received partial information for a method, it issues a warning message. You should ignore this message unless your run requires that the information for all methods be captured.

If the information for all methods must be captured, lower the value of the **minimum.method.latency** property or set it to zero.

Consider the following when setting the **minimum.method.latency** property:

- ▶ The lower the value of the **minimum.method.latency** property, the greater the chance that the performance of your application will be adversely impacted.
- ▶ Depending on your platform, and whether native timestamps are being used (**use.native.timestamps** = false), it may not be useful to specify this value in increments of less than 10 ms.
- ▶ For Mercury Business Availability Center, LoadRunner, and Performance Center capture, the throttle mechanism will automatically increase the effective minimum method latency value if the Diagnostics Server is unable to keep up with the number of events being sent.

Controlling Depth Trimming

Methods that are called at a stack depth less than or equal to the value of the **maximum.stack.depth** property will be captured. Those that are called at a stack depth greater than this limit will be trimmed to avoid incurring the overhead for methods that are less likely to be of interest.

For example:

- ▶ If **maximum.stack.depth** is 3
- ▶ /login.do calls a() calls b() calls c()
- ▶ Only /login.do, a, and b will be captured.

The following should be considered when setting the **maximum.stack.depth** property.

- ▶ Setting a low **maximum.stack.depth** can significantly reduce the overhead of capture.

- In AM mode, it is not useful to have a **maximum.stack.depth** configured higher than the Diagnostics Server's depth trim. The Diagnostics Server's depth trimming is set using **trimming.type=depth** in **<Diagnostics Server_install_dir>/etc/server.properties**. This property is disabled by default.

Controlling Probe Throttling

About Throttling

The default configuration for the J2EE Probe is set to minimize the performance impact of collecting Diagnostics information. In addition, the probe is able to scale back automatically on the amount of information that it captures when it detects that its processing has started to have a negative impact on the performance of your application. The process for scaling back the amount of information that the probe captures is called *throttling*.

The probe throttles the amount of Diagnostics information that it collects by skipping over method calls that occur relatively quickly. The probe determines exactly what “relatively quickly” means based upon the amount of data that it is collecting. The skipped method calls can have a duration that starts at the configured minimum latency trim value, which defaults to 51 milliseconds. As load increases, the minimum latency trim value is increased. When the load decreases, the trim value is reduced to the configured minimum value.

The probe properties that are used to control throttling are listed below. These properties are found in **<probe_install_dir>/etc/capture.properties**.

- **gentle.reserve.buffer.count**

The gentle reserve buffers are temporarily allocated for workload spikes, while the load throttle measures are deploying. By default, the **gentle.reserver.buffer.count** property is set to the same value as the **maximum.private.buffer.count** property.

► **hard.reserve.buffer.count**

The hard reserve buffers are allocated for workload spikes when it is determined that the gentle reserve buffering cannot keep up. By default, the **hard.reserve.buffer.count** is set to the same value as the **maximum.private.buffer.count** property.

► **buffer.wait.time**

The **buffer.wait.time** property is used to control the length of time that the probe will wait for an event to be buffered. This property tells the probe what to do when all throttle attempts are exhausted and an event cannot be buffered:

- -1 = indicates that the processing should wait for as long as it takes
- 0 = indicates that the event should be dropped immediately
- # = indicates that the processing should wait for # milliseconds before dropping the event

Turning Off Throttling

When you are using the probe to monitor a production system, the preferred behavior is to allow the probe to throttle automatically. However, when diagnosing some performance issues, you may want to sacrifice the performance of your application so that you can be sure that every diagnostic event is captured.

To configure the probe so that it will not throttle, you must edit the following properties in the `<probe_install_dir>/etc/capture.properties` file:

- Set **gentle.reserve.buffer.count** to 0
- Set **hard.reserve.buffer.count** to 0
- Set **buffer.wait.time** to -1

Note: When you are turning off throttling, be sure to set all three of the properties as instructed.

Tuning Throttling

By reviewing the probe logs, you can determine when throttling is being started and stopped. If throttling is being engaged more than you would like you should check to make sure that the probe thread configuration is synchronized with the number of threads that the application server has been configured to allow. If the application server has more threads than the probe can handle, throttling is engaged sooner and may not be disengaged.

You can check the maximum number of threads that can generate events into the probe by checking the log messages after the probe is started. A message similar to the following message should be logged:

```
2006-02-01 12:37:30,203 INFO class com.mercury.opal.capture.util.BufferPool [main]
BufferPool ready: buffer size=6600000 (max 66 threads), # events in throttle
overflow=4000000
```

In this example the maximum number of threads is 66.

The maximum number of threads that can generate events can be influenced using the **maximum.private.buffer.count** property, in property file `<probe_install_dir>/etc/capture.properties`.

If the maximum number of threads indicated by the log message is less than the number of threads that your application server can allocate, you can increase the value of the property. Be careful when increasing this value because this causes the memory overhead of the probe to increase. If the memory overhead is of concern, the value of the **maximum.buffer.size** and **minimum.buffer.size** properties may need to be decreased by an amount proportionate to the increase in the maximum buffer count.

Configuring a Probe for a Proxy Server

Important! This section only applies if you are using the probe with a Diagnostics Server.

There are two properties that are used to tell the probe the URL of the Diagnostics Server in Commander mode. The property that you set depends upon whether there is a proxy present or not.

► **registrar.url in dispatcher.properties**

The **registrar.url** property in `<probe_install_dir>\etc\dispatcher.properties` is set when you install the probe. When there is a direct connection between the probe and the URL of the Diagnostics Server in Commander mode, the value of this property is used by the probe.

► **registrar.url in webserver.properties**

In the presence of a proxy, you must set the **registrar.url** property in the `<probe_install_dir>\etc\webserver.properties` file to indicate the URL of the Diagnostics Server in Commander mode.

Configuring Reverse HTTP for a Probe in MMS

When you install a J2EE Probe that is to be used in a Mercury Managed Services (MMS) environment you are encouraged to disable the reverse http (rhttp) communication between the probe and the Diagnostics Server in Mediator mode. This configuration prevents other customers in the MMS environment from inadvertently getting the metrics from your applications.

The following instructions have been provided so that you can disable or enable rhttp after the probe has been installed.

Disabling Reverse HTTP

If you neglected to disable rhttp when the probe was installed you can disable it manually using the following instructions.

To disable rhttp, comment out the **dispatcher.objects** property value in `<probe_install_dir>/etc/modules.properties` as shown in the following example:

```
#####
## Dispatcher Module properties
#####
dispatcher.objects=\
  com.mercury.diagnostics.common.net.InternalCommSecurityManager \
  com.mercury.diagnostics.common.modules.HostNameResolver, \
# com.mercury.opal.capture.dispatcher.RhttpClient
```

Enabling Reverse HTTP

If you inadvertently disabled rhttp for a probe, you can enable rhttp manually using the following instructions.

To enable rhttp, uncomment the **dispatcher.objects** property value in `<probe_install_dir>/etc/modules.properties` as shown in the following example:

```
#####
## Dispatcher Module properties
#####
dispatcher.objects=\
  com.mercury.diagnostics.common.net.InternalCommSecurityManager \
  com.mercury.diagnostics.common.modules.HostNameResolver, \
com.mercury.opal.capture.dispatcher.RhttpClient
```

Diagnostics Probe Administration Page

You can use the Diagnostics Probe Administration page to configure J2EE Probe and Profiler settings. You access the Diagnostics Probe administration page directly from your browser.

Accessing the Diagnostics Probe Administration Page

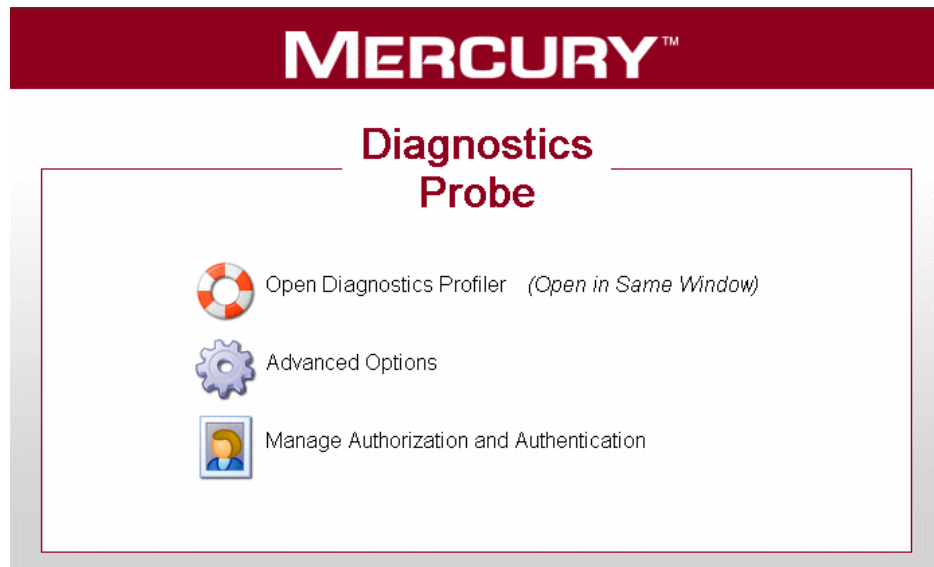
You open the Diagnostics Probe administration page inside your browser.

To access the Diagnostics Probe administration page:

- 1 In your browser, navigate to http://<probe_host>:<probeport>.

A probe is assigned to the first available port, beginning at 35000.

The Administration page opens.



- 2 Select the menu option for the activity that you want to perform.
 - **Open Diagnostics Profiler.** Opens the J2EE Diagnostics Profiler.
 - **Advanced Options.** Opens the Components pages. For more information, see “Diagnostics Probe Components Page” on page 410.
 - **Manage Authorization and Authentication.** Depending on how your probe is configured, you will access a different pages from this option
 - If your probe is configured to work with a Diagnostics Server, the probe (Profiler) authorization and authentication settings are managed from the Diagnostics Server in Commander mode to which this probe is connected. In this case, when you click this option, you are redirected to that Diagnostics Server in Commander mode. For more information, see Appendix B, “User Authentication and Authorization.”
 - If your probe is configured to work as a Profiler only and is not connected to any Diagnostics Server, this option opens the User Administration page, where you can create, edit and delete users and change their privileges. For more information, see “Authentication and Authorization for J2EE Profilers in Standalone Mode” on page 412.

Diagnostics Probe Components Page

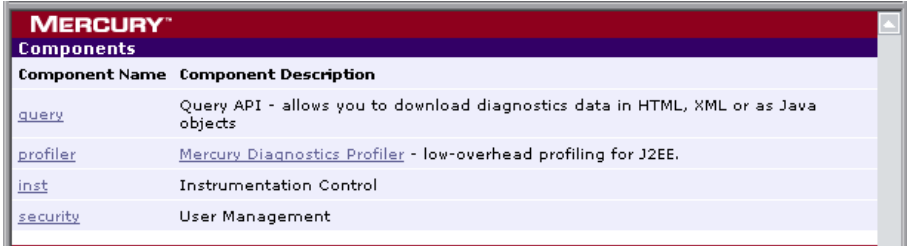
From the Components page you can open the J2EE Diagnostics Profiler, and access the User Administration page.

To access the Components page:

- 1 Open the Diagnostics Probe Administration page as described in “Accessing the Diagnostics Probe Administration Page” on page 409.
- 2 Click **Advanced Options**.

- 3 If prompted, enter your user name and password.

The Components page opens.



MERCURY™	
Components	
Component Name	Component Description
query	Query API - allows you to download diagnostics data in HTML, XML or as Java objects
profiler	Mercury Diagnostics Profiler - low-overhead profiling for J2EE.
inst	Instrumentation Control
security	User Management

- 4 Click one of the listed options, as follows:

- **query.** For internal use by developers.
- **profiler.** Opens the J2EE Diagnostics Profiler application.
- **inst.** Includes various instrumentation options. For more information about probe instrumentation, see “Instrumenting an Application” on page 237.
- **security.** Depending on how your probe is configured, you will access a different pages from this option
 - If your probe is configured to work with a Diagnostics Server, the probe (Profiler) authorization and authentication settings are managed from the Diagnostics Server in Commander mode to which this probe is connected. In this case, when you click this option, you are redirected to that Diagnostics Server in Commander mode. For more information, see “User Authentication and Authorization” on page 489.
 - If your probe is configured to work as a Profiler only and is not connected to any Diagnostics Server, this option opens the User Administration page, where you can create, edit and delete users and change their privileges. For more information, see “Authentication and Authorization for J2EE Profilers in Standalone Mode” on page 412.

Authentication and Authorization for J2EE Profilers in Standalone Mode

When you install the J2EE Probe as a Profiler only (not connected to any Diagnostics Server), you manage the authentication and authorization of users of the Profiler from the Diagnostics Probe User Administration page.

Note: If the J2EE Probe is configured to work with a Diagnostics Server, the probe (Profiler) authorization and authentication settings are managed from the Diagnostics Server in Commander mode to which this probe is connected. For more information, see “User Authentication and Authorization” on page 489.

To manage authentication and authorization for users of the standalone J2EE Diagnostics Profiler:

1 Access the Diagnostics Probe administration page

- In your browser, navigate to http://<probe_host>:<probeport>. A probe is assigned to the first available port, beginning at 35000.

The Diagnostics Probe administration page opens.

2 Select Manage Authorization and Authentication to open the User Administration page.

The screenshot shows the MERCURY User Administration interface. It features a table with columns for user management and a section for password changes.

Delete	User Name	Privileges					Change Password	
		View	Execute	Change	System	RhttpOut	Password	Confirm Password
<input checked="" type="checkbox"/>	admin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	mercury	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>	<input type="text"/>

Below the table, there is a field labeled "Password for admin:" with an adjacent input box.

At the bottom of the page, there are three buttons: "Save Changes", "Create User", and "Cancel".

On the User Administration page, you can create new users, assign privileges to users, change passwords of existing users, and delete users.

To create a new user:

- 1** Click **Create User**, enter a user name in the **New User Name** box, and click **OK**. The new user appears in the list of user names.
- 2** In the row representing the new user, type a password in the **Password** box and confirm it by retyping it in the **Confirm Password** box.
- 3** Type the password of the user currently logged on, in the **Password for <current user>** box and click **Save Changes**.

To assign privileges to a user:

- 1** Go to the row representing the relevant user and select the appropriate check boxes representing the different privileges.

The following privilege levels can be assigned to J2EE Diagnostics Profiler users:

Privilege	Description
View	The user can view Profiler data from the UI.
Execute	The user can perform garbage collection and clear the performance data held by the Profiler.
Change	The user can run potentially risky operations, such as taking a heap-dump or changing instrumentation.

Note: The privilege levels, **rhttpout** and **system** are for internal purposes only.


Each privilege level stands alone. There is no inheritance of privileges from one level to the next. You must grant a user all of the privilege levels that are necessary to perform the functions that they need to perform.

- 2** Type the password of the user currently logged on, in the **Password for <current user>** box and click **Save Changes**.

To change the password of an existing user:

- 1 Go to the row representing the relevant user, type a password in the **Password** box, and confirm it by retyping it in the **Confirm Password** box.
- 2 Type the password of the user currently logged on, in the **Password for <current user>** box and click **Save Changes**.

To delete a user:

- 1 Type the password of the user currently logged on, in the **Password for <current user>** box.
- 2  Click the **Delete user** button corresponding to the user you want to delete.
A message box opens asking if you want to delete the selected user.
- 3 Click **OK** to delete the user.

Using BEA JRockit Mission Control

If the application that you are monitoring is running on the BEA JRockit JVM and if the J2EE Probe has been configured to work with a Diagnostics Server, then you have access to a BEA JRockit Mission Control license that allows you to use Mission Control to monitor your applications performance.

When the probe that is monitoring an application running on the BEA JRockit JVM starts up and connects with the Diagnostics Server, it will ask the Diagnostics Server for the BEA license file if it does not already have one. The probe stores the BEA license file to enable you to license and begin using BEA JRockit Mission Control.

Part VII

Advanced Component Communication

22

Enabling HTTPS Between Diagnostics Components

This chapter describes the configuration steps that you must perform to enable HTTPS communications between the Mercury Diagnostics components.

This chapter describes:	On page:
About Configuring HTTPS Communications	418
Enabling Incoming HTTPS Communication for Diagnostics Components	419
Enabling Outgoing HTTPS Communication from Diagnostics Components	424
Enabling HTTPS Communications for the Mercury Business Availability Center Server	428

Note: The configuration instructions described in this chapter are intended for experienced users with in-depth knowledge of Mercury Diagnostics. Please use caution when modifying any configuration settings for the Diagnostics components.

About Configuring HTTPS Communications

The instructions for configuring each type of component contain details of the following main steps:

- Generate a keystore on the component.
- Export the certificate from the keystore
- Obfuscate passwords that provide access to the keystore.
- Copy the component's certificate to the Diagnostics components that will initiate communication.
- Configure the component's security properties to enable SSL and provide the passwords necessary for HTTPS communication to take place.

Note: As you review this information it will be useful to reference the Component Communication Diagram in Appendix E, “Diagnostics Component Configuration and Communication Diagrams”.

Enabling Incoming HTTPS Communication for Diagnostics Components

The following instructions provide you with the steps to configure the Diagnostics Server or J2EE Probe to receive incoming HTTPS communications. The HTTPS communications can come from other Diagnostics components, from when the Diagnostics component is accessed using a Web browser, or when the component is accessed by other external application.

Notes:

- ▶ Enabling SSL and HTTPS Communications for the J2EE Probe is supported on SUTs with the Sun, IBM, and JRockit JVMs. However, if you are using a JVM version prior to 1.4, you must download and install the Sun JSSE Optional Package onto the SUT server before you can enable SSL.

Other JSSE implementation, such as IBM's are not supported.

- ▶ To avoid issues with DNS and host name resolution, the Commander URL for the Diagnostics Server in Commander mode should be configured as "localhost" This can be accomplished by setting the `commander.url` property in `<server_install_dir>/etc/server.properties` to `http://localhost:2006`.
-

To configure the Diagnostics Server or J2EE Probe for incoming HTTPS connections:

- 1 Generate a keystore in the `<diagnostics_server_install_dir>/etc` directory or `<J2EE_probe_install_dir>/etc` directory using the following command:

► **Command for Diagnostics Server:**

```
<diagnostics_server_install_dir>/_jvm/bin/keytool -keystore  
<diagnostics_server_install_dir>/etc/keystore -storepass <password> -alias SERVER  
-genkey -keyalg RSA -keypass <password> -dname  
"CN=<diagnostics_server_hostname>, OU=Diagnostics, O=Mercury, L=Mountain  
View, S=CA, C=USA" -validity 3650
```

To use this command example:

- Replace `<diagnostics_server_install_dir>` with the path to the installation directory for the Diagnostics Server or replace `<J2EE_probe_install_dir>` with the path to the installation directory for the J2EE Probe.
- Replace `<diagnostics_server_hostname>` with the machine name for the host of the Mercury Diagnostics Server or replace `<probe_hostname>` with the machine name for the host of the J2EE Probe. This value cannot be the server's IP address.
- Replace each occurrence of `<password>` with the same password string. If you want, you may assign different passwords to the `storepass` and the `keypass`.

After you execute this command, a keystore is created in `<diagnostics_server_install_dir>/etc/keystore` with an entry called **SERVER** for the host of the Diagnostics Server.

► **Command for J2EE Probe:**

```
<probe_install_dir>/_jvm/bin/keytool -keystore <probe_install_dir>/etc/keystore -
storepass <password> -alias PROBE -genkey -keyalg RSA -keypass <password> -
dname "CN=<probe_hostname>, OU=Diagnostics, O=Mercury, L=Mountain View,
S=CA, C=USA" -validity 3650
```

After you execute this command, a keystore is created in `<probe_install_dir>/etc/keystore` with an entry called **PROBE** for the host of the J2EE Probe.

- 2** Export the certificate for the **SERVER** or **PROBE** entry in the keystore using the following command.

► **Command for Diagnostics Server:**

```
<diagnositcs_server_install_dir>/_jvm/bin/keytool -keystore
<diagnositcs_server_install_dir>/etc/keystore -storepass <password> -alias
SERVER -export -rfc -file
<diagnositcs_server_install_dir>/etc/<server_certificate_name>.cer
```

To use this command:

- Replace `<diagnostics_server_install_dir>` with the path to the installation directory for the Diagnostics Server, or replace `<probe_install_dir>` with the path to the installation directory for the J2EE Probe.
- Replace `<password>` with the string that you assigned as the **storepass** password when you created the keystore.
- Replace `<server_certificate_name>` or `<probe_certificate_name>` with the name that you would like to assign to the certificate file. It is recommended that you assign a certificate name that will make it easy to recognize the component for which the certificate was created.

For Diagnostics Servers, use `diag_server_commander.cer` or `diag_server_mediator.cer`.

For J2EE Probes, include the type of the probe and the host name for the probe so that it will be easy to recognize the component for which the certificate was created. For example: `j2EE_probe_<probe_hostname>`.

After this command runs, a certificate file with the name assigned in `<server_certificate_name>` is created in the `<diagnostics_server_install_dir>/etc` directory for the Diagnostics Server, for example, `diag_server_commander.cer`.

► **Command for J2EE Probe:**

```
<probe_install_dir>/_jvm/bin/keytool -keystore <probe_install_dir>/etc/keystore -  
storepass <password> -alias PROBE -export -rfc -file  
<probe_install_dir>/etc/<probe_certificate_name>.cer
```

After this command runs, a certificate file called `j2ee_probe_<probe_hostname>.cer` is created in the `<probe_install_dir>/etc` directory for the J2EE Probe.

Note: The certificate file must be imported to the host machines for each of the Diagnostics components that are expected to initiate communications with the Diagnostics Server or J2EE Probe. The instructions for importing the certificate file to each Diagnostics component are provided below.

- 3 Using the command in the following example, generate an obfuscated version of the **storepass** and the **keypass** passwords that you assigned when you created the keystore.
 - a Replace `<diagnostics_server_install_dir>` with the path to the installation directory for the Diagnostics Server, or replace `<probe_install_dir>` with the path to the installation directory for the J2EE Probe.
 - b Replace `<password>` with the string that you assigned as the password when you created the keystore.

```
<diagnostics_server_install_dir>/_jvm/bin/java -cp  
<diagnostics_server_install_dir>/lib/ThirdPartyLibs.jar org.mortbay.util.Password  
<password>
```

The output from the obfuscation is shown in the following example. In this example, the password string was “testpass”. The output consists of three lines. The original string that was to be obfuscated and two lines depicting the obfuscated password. Only the line that begins with “OBF” is used to set the properties in the following step of this process.

```
testpass
OBF:1ytc1vu91v2p1y831y7v1v1p1vw11yta
MD5:179ad45c6ce2cb97cf1029e212046e81
```

Note: If you did not use the same password for **keypass** and **storepass**, you must run this command twice to create an obfuscated version for each password.

- 4** Change the following properties in the file `<diagnostics_server_install_dir>/etc/security.properties` for the Diagnostics Server in Commander mode, or `<probe_install_dir>/etc/security.properties` for the Mercury Diagnostics Probe for J2EE.
 - a** Set `enableSSL=true`.
 - b** Set `keyStorePassword=<obfuscated_password>`
 - c** Set `keyPassword=<obfuscated_password>`

Note: The value entered for `<obfuscated_password>` must include the entire “OBF” line that was output from the command in the previous step. For example:

```
keyStorePassword=OBF:1ytc1vu91v2p1y831y7v1v1p1vw11yta
```

Enabling Outgoing HTTPS Communication from Diagnostics Components

The following instructions provide you with the steps necessary to configure the Diagnostics Server and J2EE Probe to send outgoing HTTPS communications to the other Diagnostics components.

To enable the Diagnostics Server in Commander mode for outgoing communication to the Diagnostics Server in Mediator mode via HTTPS.

- 1 Copy the certificate file from `<diagnostics_server_install_dir>/etc/diag_server_mediator.cer` on the Diagnostics Server to `<diagnostics_server_install_dir>/etc/diag_server_mediator.cer` on the Diagnostics Server in Commander mode.
- 2 Change the value of the `trusted.certificate` property in the file `<diagnostics_server_install_dir>/etc/security.properties` for the Diagnostics Server in Commander mode.

Set `trusted.certificate=diag_server_mediator.cer`. If there are already other certificate files included in the value of this property, add the certificate file to the end of the list separated from the preceding value by a comma.

- 3 For incoming Diagnostics Server communication, indicate the URL for the Diagnostics Server by updating the following property in the file `<diagnostics_server_inst_dir>/etc/server.properties`.

Set `commander.url` to `https://<diagserver_hostname>:8443`

To enable the Diagnostics Server in Mediator mode for outgoing communication to the Diagnostics Server in Commander mode via HTTPS:

- 1 Copy the certificate file from `<diagnostics_server_install_dir>/etc/diag_server_commander.cer` on the Diagnostics Server in Commander mode to `<diagnostics_server_install_dir>/etc/diag_server_commander.cer` on the Diagnostics Server in Mediator mode.

- 2 Change the value of the **trusted.certificate** property in the file `<diagnostics_server_install_dir>/etc/security.properties` for the Diagnostics Server in Mediator mode.

Set **trusted.certificate=diag_server_commander.cer**. If there are already other certificate files included in the value of this property, add the certificate file to the end of the list separated from the preceding value by a comma.

- 3 For incoming Diagnostics Server communication, indicate the URL for the Diagnostics Server by updating the following property in the file `<diagnostics_server_inst_dir>/etc/server.properties`.

Set **commander.url** to **https://<diagserver_hostname>:8443**

To enable the Diagnostics Server in Mediator mode for outgoing communications to the probes via HTTPS:

- 1 Copy the certificate file from `<probe_install_dir>/etc/j2ee_probe_<probe_host>.cer` for each probe to `<diagnostics_server_install_dir>/etc/j2ee_probe_<probe_host>.cer` on the Diagnostics Server in Mediator mode.
- 2 Change the value of the **trusted.certificate** property in the file `<diagnostics_server_install_dir>/etc/security.properties` for the Diagnostics Server in Mediator mode.

Set **trusted.certificate=j2ee_probe_<probe_host>.cer**. If there are already other certificate files included in the value of this property, add the certificate file to the end of the list separated from the preceding value by a comma.

To enable the J2EE Probe for outgoing communications to the Diagnostics Server in Mediator mode via HTTPS:

- 1 Copy the certificate file from `<diagnostics_server_install_dir>/etc/diag_server_mediator.cer` on the Diagnostics Server in Mediator mode to `<probe_install_dir>/etc/diag_server_mediator.cer` on the J2EE Probe.
- 2 Change the value of the `trusted.certificate` property in the file `<probe_install_dir>/etc/security.properties` for the J2EE Probe.

Set `trusted.certificate=diag_server_mediator.cer`. If there are already other certificate files included in the value of this property, add the certificate file to the end of the list separated from the preceding value by a comma.

- 3 For incoming J2EE Probe communication, Indicate the URL for the Diagnostics Server in Mediator mode by updating the following property in the file `<probe_inst_dir>/etc/dispatcher.properties`.

Set `registrar.url` to

`https://<diagserv_mediatormode_hostname>:8443/commander/registrar/`

Configuring a .NET probe to Communicate via HTTPS with the Commander

To enable the .NET Probe for outgoing communications to the Diagnostics Server in Commander mode via HTTPS:

- 1 Copy the keystore for the Diagnostics Server in Commander mode to the host for the .NET Probe. The keystore was generated when the Diagnostics Server in Commander mode was configured to receive HTTPS. See “Enabling Incoming HTTPS Communication for Diagnostics Components” on page 419 for instructions to configure the Diagnostics Server in Commander mode to receive HTTPS. If you followed the instructions in the referenced section, the keystore can be found in `<diagnostics_server_install_dir>/etc/keystore`.
- 2 Edit the `<probe_install_dir>/etc/probe_config.xml` and change the `url` attribute for the `<registrar>` tag to specify the HTTPS URL for the Diagnostics Server in Commander mode. For example:

```
<registrar url="https://<diagnostics_server_host>:8443/registrar"/>
```

- 3 On the Windows Taskbar, click **Start > Run**.

- 4** Run the Microsoft Management Console by typing `mmc`, and then clicking **OK**.
- 5** On the Microsoft Management Console menu, click `File > Add/Remove Snap-in` to display the Add/Remove Snap-in dialog.
- 6** Click **Add** on the Add/Remove Snap-in dialog.
- 7** Select **Certificates** from the Available Standalone Snap-in list and click **Add**.
- 8** On the Certificates Snap-in dialog select **Computer account**, and click **Next**.
- 9** On the Select Computer dialog select **Local Computer: (the computer this console is running on)**, and then click **Finish**.
- 10** Click **Close** on the Add Standalone Snap-in.
- 11** Click **OK** on the Add/Remove Snap-in dialog.
- 12** On the Microsoft Management Console expand the listing for **Certificates (Local Computer)** in the left pane of the Console Root dialog.
- 13** Under **Certificates (Local Computer)**, expand **Trusted Root Certification Authorities**.
- 14** Under **Trusted Root Certification Authorities**, right-click **Certificates** and select **All Tasks > Import** to start the Certificate Import Wizard.
- 15** Click **Next** to move past the Welcome dialog box of the Certificate Import Wizard.
- 16** Click **Browse** to navigate to the public keystore for the Diagnostics Server in Commander mode.
 - a** Select **All Files (*.*)** in Files of type:
 - b** Navigate to the directory where the keystore for the Diagnostics Server in Commander mode was copied in step 1 and click **Open**. This should be `<diagnostics_server_install_dir>/etc/keystore`
- 17** Click **Next** to import the file.
- 18** Click **Next** to accept the default Certificate Store location of “Trusted Root Certification Authorities”.
- 19** Click **Finish** on Completing the Certificate Import Wizard.
- 20** Click **OK** on the Certificate Import Wizard confirmation dialog.

- 21 Restart IIS. For instructions on restarting IIS see “Restarting IIS” on page 334.

To verify that you have successfully configured the .NET probe for HTTPS communication with the Diagnostics Server in Commander mode:

- 1 Browse to your .NET application to activate the .NET Probe.
- 2 Verify that the .NET Probe node is shown in System Health.

Enabling HTTPS Communications for the Mercury Business Availability Center Server

The following instructions will guide you through the process of configuring Mercury Business Availability Center for HTTPS communication with Diagnostics.

To enable HTTPS communications between the Diagnostics Server in Commander mode and Mercury Business Availability Center:

- 1 Copy the Diagnostics certificate file, **diag_server_commander.cer**, from the Diagnostics Server in Commander mode installation directory, `<diagnostics_server_install_dir>/etc/`, to the Mercury Business Availability Center host.
- 2 Import the copied certificate, **diag_server_commander.cer**, into the Mercury Business Availability Center server cacert keystore by running the following command on the Mercury Business Availability Center host:

```
<BAC_server_install_dir>/_jvm/bin/keytool -import -file  
<copied_diag_certificate_directory>/diag_server_commander.cer -keystore  
<BAC_server_install_dir>/jre/lib/security/cacerts -alias SERVER
```

- Replace `<BAC_server_install_dir>` with the path to the installation directory for Mercury Business Availability Center.
- Replace `<copied_diag_certificate_directory>` with the path to the copied Diagnostics certificate file.

Type `changeit` when you are prompted to enter the keystore password.

Type `yes`, instead of the default `no` when you are asked if the certificate should be trusted.

- 3 Copy the Mercury Business Availability Center certificate file, `<BAC_certificate_file.cer>`, to the Diagnostics Server host.
- 4 Import the copied certificate into the Diagnostics Server cacert keystore by running the following command on the Diagnostics Server host.

```
<diagnostics_server_install_dir>/_jvm/bin/keytool -import -file
<copied_BAC_certificate_directory>/<BAC_certificate_file.cer> -keystore
<diagnostics_server_install_dir>/JRE/lib/security/cacerts
```

- Replace `<diagnostics_server_install_dir>` with the path to the installation directory of the Diagnostics Server in Commander mode.
- Replace `<copied_BAC_certificate_directory>` with the path to the copied Mercury Business Availability Center certificate file.

When you are prompted to enter the keystore password, type the string that you assigned as the **storepass** password when you created the keystore.

Type **yes**, instead of the default **no**, when you are asked if the certificate should be trusted:

- 5 Point the Mercury Business Availability Center server to the HTTPS port on the Diagnostics Server in Commander mode.
 - a Open the Diagnostics Administration in Mercury Business Availability Center by selecting **ADMIN > Diagnostics**.
 - b Click the **Registration** tab.
 - c Locate the Diagnostics Server details section.
 - d Provide the following information in the appropriate fields:
 - Enter the host name for the Diagnostics Server in Commander mode exactly as it was specified in the **CN** parameter when you created the keystore for the Diagnostics Server in Commander mode. See “Enabling Incoming HTTPS Communication for Diagnostics Components” on page 419.
 - Enter HTTPS for the protocol
 - Enter 8443 for the Web port of the Diagnostics Server.
 - e Click **Submit Configuration**.

23

Configuring Diagnostics Components for HTTP Proxy

This chapter describes the configuration steps that you must perform to enable HTTP proxy communications between the Mercury Diagnostics components.

This chapter describes:	On page:
Enabling HTTP Proxy Communications for the Diagnostics Servers	432
Enabling HTTP Proxy Communications for the J2EE Probe	433
Enabling HTTP Proxy Communications for a .NET Probe	433

Note: The configuration instructions described in this chapter are intended for experienced users with in-depth knowledge of Mercury Diagnostics. Please use caution when modifying any configuration settings for the Diagnostics components.

Enabling HTTP Proxy Communications for the Diagnostics Servers

The following section describes how to configure the Diagnostics Servers in Commander mode and Diagnostics Servers in Mediator mode to communicate with each other through an HTTP proxy.

To configure the Diagnostics Servers for HTTP proxy communications:

- 1** Set the following properties in `<diagnostics_server_install_dir>/etc/server.properties`:
 - Set **proxy.host** to the host name of the proxy server.
 - Set **proxy.port** to the port of the proxy server.
 - Set **proxy.protocol** to the protocol to use for the proxy server (http).
 - Set **proxy.user** to the user used to authenticate the proxy server.
 - Set **proxy.password** to the password used to authenticate the proxy server.
 - For a Diagnostics Server in Commander mode that is to run over the proxy, set **commander.url** so that the host name is the real host name and not a localhost.
- 2** Restart the Diagnostics Server. For instructions, see “Starting and Stopping the Diagnostics Server” on page 46.

Enabling HTTP Proxy Communications for the J2EE Probe

The following section describes how to configure the J2EE Probe to communicate with Diagnostics Server in Commander mode through an HTTP proxy:

To configure the J2EE Probe for HTTP proxy communications:

- 1** Set the following properties in `<J2ee_install_dir>/etc/dispatcher.properties`:
 - ▶ Set **proxy.host** to the host name of the proxy server.
 - ▶ Set **proxy.port** to the port of the proxy server.
 - ▶ Set **proxy.protocol** to the protocol to use for the proxy server (http).
 - ▶ Set **proxy.user** to the user used to authenticate the proxy server.
 - ▶ Set **proxy.password** to the password used to authenticate the proxy server.
- 2** Restart the instrumented application VM.

Enabling HTTP Proxy Communications for a .NET Probe

The following section describes how to configure the .NET Probe to communicate with the Diagnostics Server in Commander mode through an HTTP proxy:

To configure the .NET Probe for HTTP proxy communications:

- 1** Set the following **proxy** properties in the `<probe_install_dir>/etc/probe_config.xml` file to point to the Diagnostics Server host:
 - ▶ Set **uri** to the host for the Diagnostics Server.
 - ▶ Set **proxy** to the proxy url.
 - ▶ Set **proxy.user** to the user used to authenticate the proxy server.
 - ▶ Set **proxy.password** to the password used to authenticate the proxy server.

The following example shows how this would look in the **probe_config.xml** file:

```
<diagnosticsserver url="http://<diagserver_host_name>:2006/registrar/"  
proxy="http://proxy:8080" proxyuser= "<username>" proxypassword="<password>"/>
```

- 2** Set the following **proxy** properties in the **<probe_install_dir>/etc/metrcis.config** file to configure system metrics to use a proxy:
 - ▶ Set **proxy.uri** to the proxy url
 - ▶ Set **proxy.user** to the user used to authenticate the proxy server.
 - ▶ Set **proxy.password** to the password used to authenticate the proxy server.
- 3** Restart the instrumented application process.

24

Configuring Diagnostics Components to Work with a Firewall

This chapter describes the configuration steps that you must perform to enable Mercury Diagnostics to work correctly in an environment where a firewall is present. This additional configuration is required when the firewall separates the probes from the other Diagnostics components or the components of LoadRunner, Performance Center, or Mercury Business Availability Center.

This chapter describes:	On page:
Overview of Configuring Diagnostics for a Firewall	436
Collating Offline Analysis Files over a Firewall	439
Installing and Configuring the Mercury MI Listener	440
Configuring the Diagnostics Server in Mediator mode to Work with a Firewall	441
Configuring LoadRunner and Performance Center to Work with Diagnostics Firewalls	448

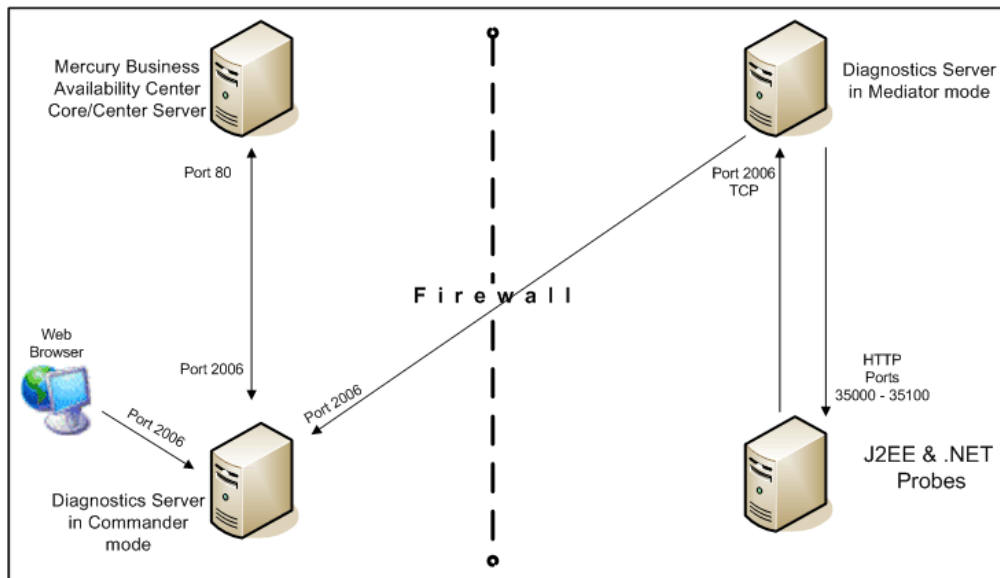
Note: The configuration instructions described in this chapter should be used only by experienced users with in-depth knowledge of Mercury Diagnostics. Please use caution when modifying any configuration settings for the Diagnostics components.

Overview of Configuring Diagnostics for a Firewall

Configuring Diagnostics for a firewall differs depending on which Mercury Product is part of the Diagnostics integration.

Mercury Business Availability Center

The diagram below shows a typical Diagnostics topology where a firewall separates the probe from the other Diagnostics and Mercury Business Availability Center components.



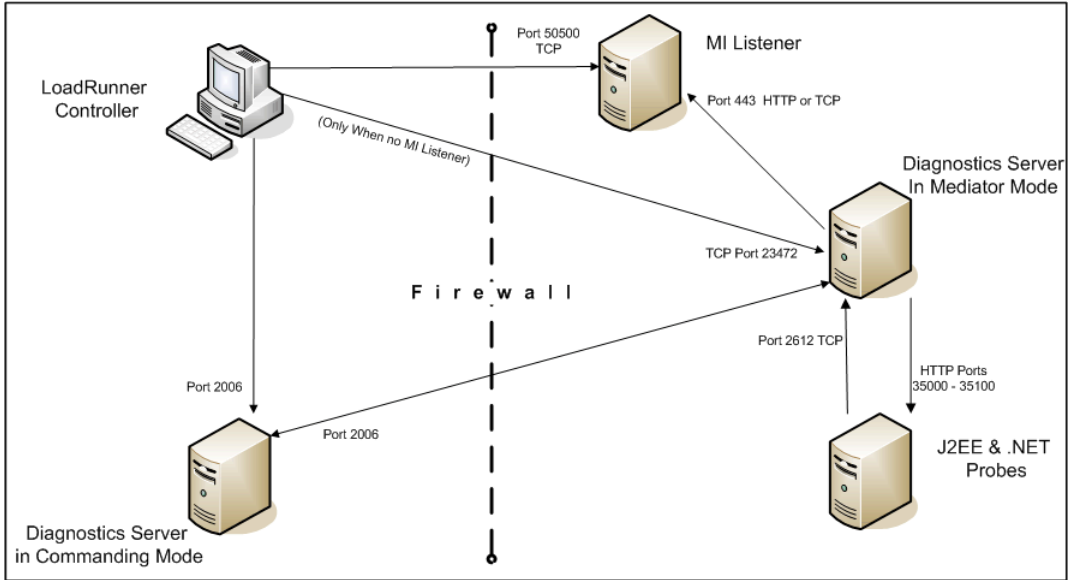
To configure the firewall to enable the communications between the Diagnostics components, open the ports that will allow:

- ▶ HTTP requests from the Mercury Business Availability Center Core/Center server(s) to the Diagnostics Server in Commander mode, on port 2006.
- ▶ HTTP requests from the Diagnostics Server in Commander mode to Mercury Business Availability Center core server on port 80.
- ▶ HTTP requests from the Diagnostics Server in Mediator mode to ports 35000-35100 of probe.

- ▶ HTTP requests from the Web Browser Client machine to the Diagnostics Server in Commander mode on port 2006.
- ▶ HTTP requests from the Diagnostics Server in Mediator mode to the Diagnostics Server in Commander mode on port 2006.
- ▶ TCP requests from the probe to the Diagnostics Server in Mediator mode on port 2612.
- ▶ HTTP requests from the probe to the Diagnostics Server in Mediator mode on port 2006.
- ▶ HTTP requests from the Diagnostics Server in Commander mode to the probes on port 35000-35100. The actual ports on which you must allow communications will depend upon the port numbers that you enabled when you configured the probe and the number of instrumented VMs. For information on setting the probe port range, see “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.

LoadRunner and Performance Center

The diagram below shows a typical Diagnostics topology where a firewall separates the probe from the other Diagnostics and LoadRunner components.



Note: LoadRunner is used in this diagram for illustrative purposes. The same information would apply to Performance Center.

You must configure the firewall to allow the Diagnostics components to communicate with each other.

To configure the firewall to enable the communications between the Diagnostics components, open the ports that will allow:

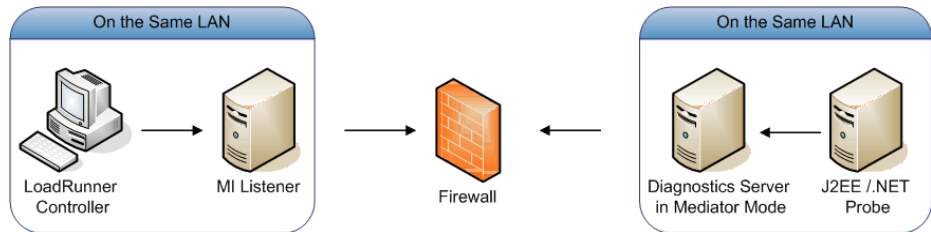
- ▶ HTTP requests from the Diagnostics Server in Mediator mode to the Diagnostics Server in Commander mode on ports 2612 and 2006.
- ▶ TCP requests from the probe to the Diagnostics Server in Mediator mode on port 2612.
- ▶ HTTP requests from the probe to the Diagnostics Server in Mediator mode on port 2006.
- ▶ HTTP requests from the Diagnostics Server in Commander mode to the probes on port 35000-35100. The actual ports on which you must allow communications will depend upon the port numbers that you enabled when you configured the probe and the number of instrumented VMs. For information on setting the probe port range, see “Configuring the Probes for Multiple Application Server JVM Instances” on page 393.

Note: In addition to the above topology, if you are using the LoadRunner Analysis Tool to view offline J2EE results, see “Collating Offline Analysis Files over a Firewall” to properly configure the Controller and the Diagnostics Servers in Mediator mode for offline file retrieval.

Collating Offline Analysis Files over a Firewall

During a LoadRunner / Performance Center load test, the Diagnostics Servers that have probes reporting to them generate an offline analysis file on their host machine. The offline analysis files is retrieved by LoadRunner / Performance Center when it collates the results of a load test.

If there is a firewall between the LoadRunner / Performance Center Controller and the Diagnostics Server involved in a load test, you must configure the Controller and the Diagnostics Server to use the MI Listener utility to enable the transfer of the offline analysis file. The MI Listener utility comes with LoadRunner / Performance Center and should be installed on a machine inside your firewall as shown in the following diagram.



To configure the Controller to access Diagnostics Servers that are behind a firewall:

- ▶ Install and configure the Mercury MI Listener.
- ▶ Configure the Diagnostics Servers to work with a firewall.
- ▶ Configure LoadRunner / Performance Center to work with a firewall.

Installing and Configuring the Mercury MI Listener

The Mercury MI Listener component is the same component that is used to serve Load Generators that are outside of a firewall. For more information about how to configure the MI Listener for LoadRunner, refer to the Mercury LoadRunner Controller User's Guide. For more information about how to configure the MI Listener for Performance Center, refer to the Mercury Performance Center System Configuration and Installation Guide.

Configuring the Diagnostics Server in Mediator mode to Work with a Firewall

To configure the Diagnostics Server in Mediator mode so that it can work across a firewall, you must complete the following additional configuration steps. If you have not yet installed and configured the Diagnostics Server in Mediator mode you must do so prior to attempting these steps. For instructions on installing the Diagnostics Server in Mediator mode, see Chapter 2, “Installing the Mercury Diagnostics Server.”

To configure the Diagnostics Server in Mediator mode for a firewall on a Windows machine:

- 1 Modify the `<diagnostics_server_install_dir>\nanny\windows\`.
- 2 Modify the `<diagnostics_server_install_dir>\nanny\windows\dat\nanny\server.nanny` file.

- ▶ Locate the `start_nt` line:

```
start_nt="C:\MercuryDiagnostics\Server\jre\bin\javaw.exe" -server
-Xmx512m ....
```

- ▶ Remove the quotes that surround the `java.exe` path and add the caret (^) after `javaw.exe` as shown below:

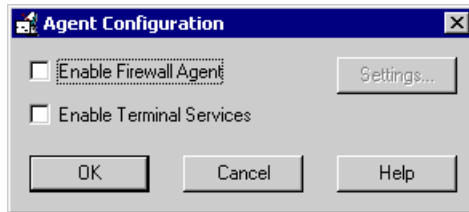
```
start_nt=C:\MercuryDiagnostics\Server\jre\bin\javaw.exe^ -server
-Xmx512m...
```

- 3 Modify the `<diagnostics_server_install_dir>/etc/server.properties` file.
 - ▶ Uncomment the `dispatcher.offline.locationprefix` property.
 - ▶ Set the value of the property to `C:/Documents and Settings/Default User/Local Settings/Temp/MOfflineFiles` as shown in the following example. Make sure that you use the correct drive letter and path for your installation.

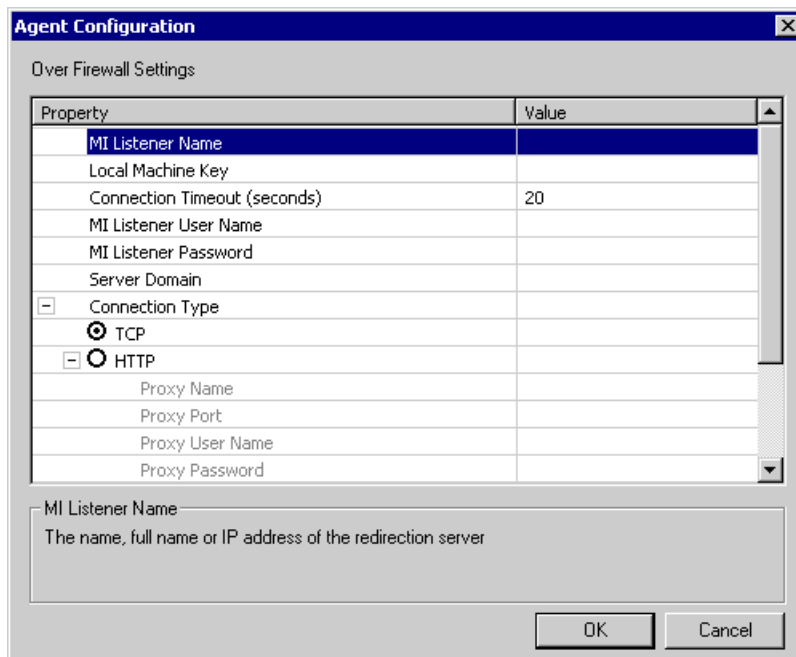
```
dispatcher.offline.locationprefix= C:/Documents and Settings/Default User/Local
Settings/Temp/MOfflineFiles
```

- 4 Launch the Agent Configuration by running `<diagnostics_server_install_dir>/nanny/windows/bin/AgentsConfig.exe`.

The Agent Configuration process opens the following dialog box opens:



- 5 Select **Enable Firewall Agent**. The **Settings** button becomes enabled.
- 6 Click **Settings**. The Agent Configuration process opens the Agent Configuration Settings dialog box.
- 7 In Value column of the MI Listener Name property, enter the host name or IP address of the machine where the MI Listener was installed.

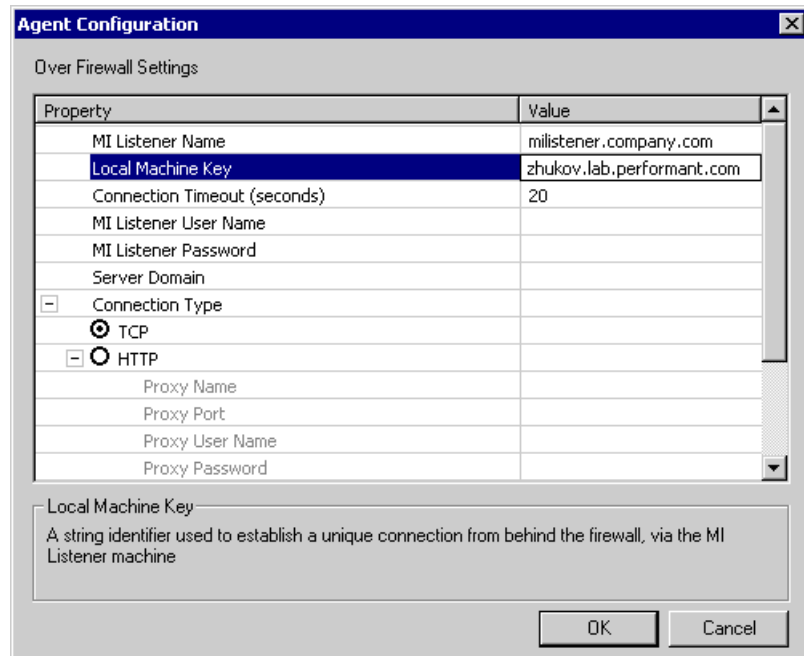


- 8 For the **Local Machine Key** property, enter the machine name of the host of the Diagnostics Server in Mediator mode.

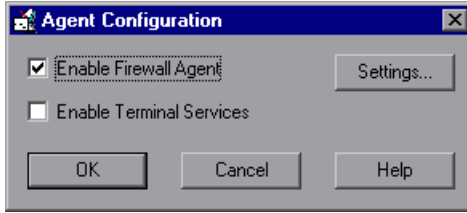
Important: When entering the host name of a Diagnostics component you must use the fully qualified host name, that is, the machine name and the domain name.

Use the System Health Monitor to determine the machine name for the host of the Diagnostics Server in Mediator mode. For more information on the System Health Monitor, see Appendix C, “Using the System Health Monitor.”

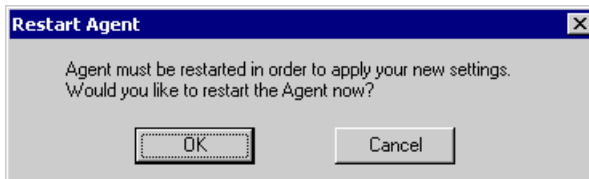
- 9 Click OK to close the setting dialog box.



- 10 Click **OK** again to close the Agent Configuration dialog box.



- 11 The Restart Agent dialog box opens. Click **OK** to restart the Agent.



- 12 Restart the Mercury Diagnostics Server service.

To configure the Diagnostics Server in Mediator mode for a firewall on a Solaris machine:

- 1 Modify the `<diagnostics_server_install_dir>/nanny/solaris/merc_asl/process.asl` file by adding the following lines:

```
nanny.exe=  
nanny=  
java=
```

- 2 Modify the `<diagnostics_server_install_dir>/nanny/solaris/dat/nanny/server.nanny` file:

- Locate the `start_solv4` line:

```
start_solv4= "/opt/Diagnostics/Server/jre/bin/java" -server -Xmx512m...
```

- Remove the quotes that surround the java path and add the caret (^) after java as shown below:

```
start_solv4= /opt/Diagnostics/Server/jre/bin/java^-server -Xmx512m...
```

- 3** Modify the `<diagnostics_server_install_dir>/nanny/solaris/dat/br_lnc_server.cfg` file.

Change the value of the `FireWallServiceActive` property to 1.

- 4** Modify the `<diagnostics_server_install_dir>/etc/server.properties` file.

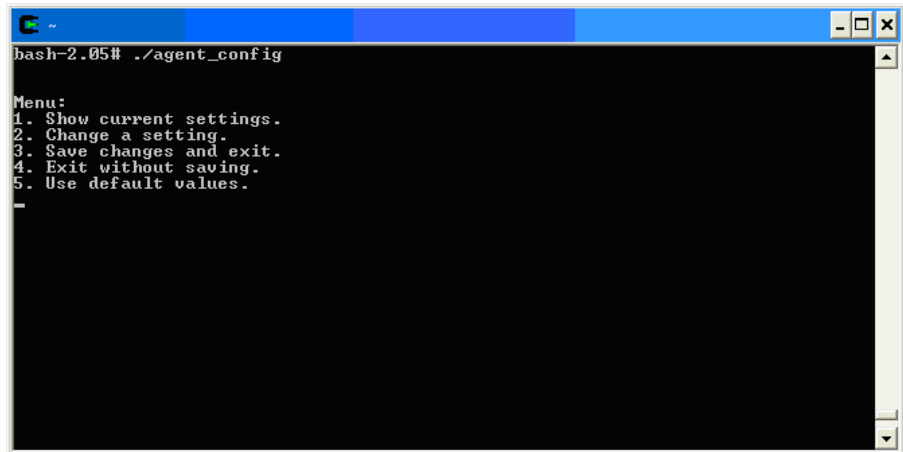
- ▶ Uncomment the `dispatcher.offline.locationprefix` property.
- ▶ Set the value of the property file to `/tmp/MOfflineFiles` as shown in the following example. The letter for your drive may be different but the path to the directory must be the same.

```
dispatcher.offline.locationprefix= /tmp/MOfflineFiles
```

- 5** Run the following commands to launch the Agent Configuration utility:

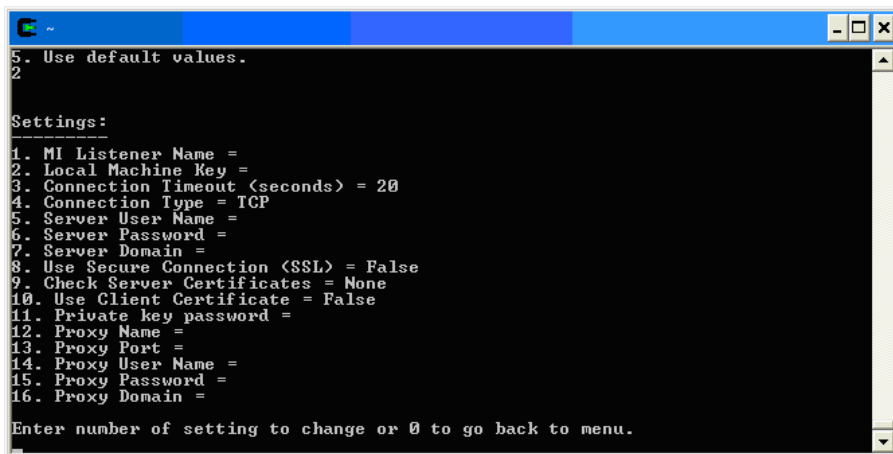
```
export LD_LIBRARY_PATH=.
export M_LROOT=<diagnostics_server_install_dir>
cd $M_LROOT/nanny/solaris/bin
./agent_config
```

- 6** In the Agent Configuration Utility window, press **2**, **Change a Setting**.



7 A list of settings appears.

Press **1** to select **MI Listener Name**, and enter the machine name or IP address of the MI Listener host.

A screenshot of a terminal window with a blue title bar. The terminal displays a settings menu. At the top, it says "5. Use default values." followed by a "2" on the next line. Below that is the heading "Settings:" followed by a list of 16 numbered settings, each followed by an equals sign. The settings are: 1. MI Listener Name =, 2. Local Machine Key =, 3. Connection Timeout (seconds) = 20, 4. Connection Type = TCP, 5. Server User Name =, 6. Server Password =, 7. Server Domain =, 8. Use Secure Connection (SSL) = False, 9. Check Server Certificates = None, 10. Use Client Certificate = False, 11. Private key password =, 12. Proxy Name =, 13. Proxy Port =, 14. Proxy User Name =, 15. Proxy Password =, 16. Proxy Domain =. At the bottom of the terminal, it says "Enter number of setting to change or 0 to go back to menu." The terminal has a scrollbar on the right side.

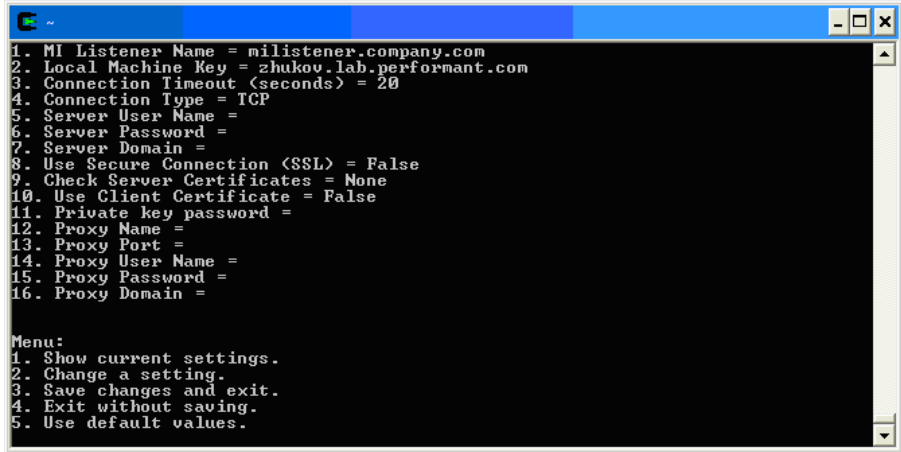
```
5. Use default values.
2

Settings:
1. MI Listener Name =
2. Local Machine Key =
3. Connection Timeout (seconds) = 20
4. Connection Type = TCP
5. Server User Name =
6. Server Password =
7. Server Domain =
8. Use Secure Connection (SSL) = False
9. Check Server Certificates = None
10. Use Client Certificate = False
11. Private key password =
12. Proxy Name =
13. Proxy Port =
14. Proxy User Name =
15. Proxy Password =
16. Proxy Domain =

Enter number of setting to change or 0 to go back to menu.
```

- 8** Press **2** to select **Change a Setting** and enter the machine name of the host of the Diagnostics Server in Mediator mode as displayed in the Host: field on the System Health Monitor.

Use the System Health Monitor to determine the machine name for the host of the Diagnostics Server in Mediator mode. For more information on the System Health Monitor, see Appendix C, “Using the System Health Monitor.”



```

1. MI Listener Name = milistener.company.com
2. Local Machine Key = zhukov.lab.performant.com
3. Connection Timeout (seconds) = 20
4. Connection Type = TCP
5. Server User Name =
6. Server Password =
7. Server Domain =
8. Use Secure Connection (SSL) = False
9. Check Server Certificates = None
10. Use Client Certificate = False
11. Private key password =
12. Proxy Name =
13. Proxy Port =
14. Proxy User Name =
15. Proxy Password =
16. Proxy Domain =

Menu:
1. Show current settings.
2. Change a setting.
3. Save changes and exit.
4. Exit without saving.
5. Use default values.

```

- 9** Press **3**, **Save changes and exit**, to complete the updates.
- 10** Restart the Diagnostics Server in Mediator mode.

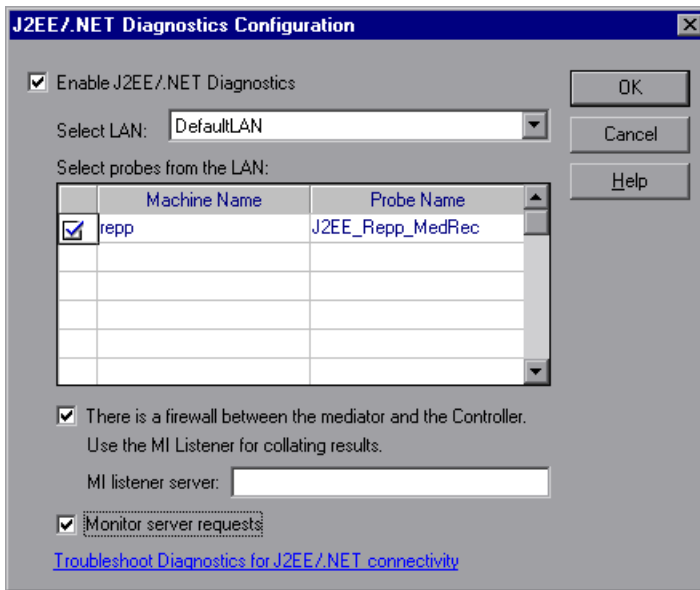
For more information about restarting the Diagnostics Server in Mediator mode, see “Starting and Stopping the Diagnostics Server” on page 46.

Configuring LoadRunner and Performance Center to Work with Diagnostics Firewalls

After the MI Listener has been installed and your Mediator machines have been configured, you must update the Diagnostics Configuration in LoadRunner / Performance Center so that the application knows to use the MI Listener when it is transferring the offline data from a Mediator that is outside of a firewall.

To configure LoadRunner to work with a Diagnostics firewall:

- 1 From the LoadRunner Controller menu bar, select **Diagnostics > Configuration** to open the Diagnostics Distribution dialog box.
- 2 Ensure that the **Enable the following diagnostics** check box is selected.
- 3 In the Online and Offline Diagnostics section, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.



- 4 Select **There is a firewall between the mediator and the Controller**.

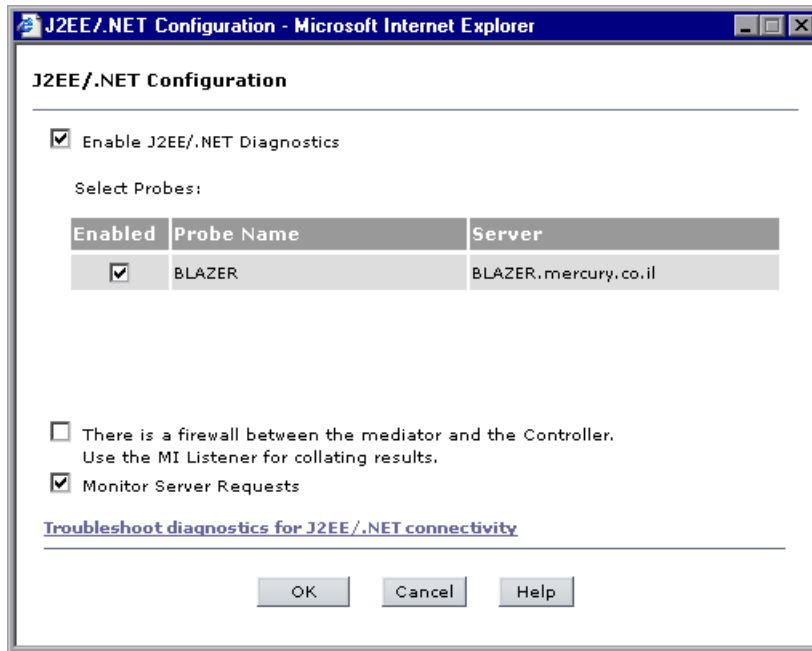
- 5 Specify the machine name of the MI Listener host in the **MI listener server** text box.

See Chapter 12, “Setting Up LoadRunner 8.1 to use Mercury Diagnostics,” for more information about configuring LoadRunner to use the Diagnostics components.

To configure Performance Center to work with a Diagnostics firewall:

- 1 In the Performance Center Administration Site, in the **General Settings** page, go to the **Firewall** section and ensure that you have specified the IP address of the MI Listener machine in the **Firewall Diagnostics Communicator** box.
- 2 In the Performance Center User site, choose **Load Tests > Create/Edit** from the left navigation menu to open the Load Test Configuration page.
- 3 Click an existing test that you want to configure in the **Name (# of Runs)** column or click **New Load Test**.
- 4 Click the **Diagnostics** tab, and ensure that the **Enable Diagnostics** check box is selected.

- 5 In the **Offline and Online Diagnostics** section, click **Configure** to open the J2EE/.NET Diagnostics Configuration dialog box.



- 6 Select **There is a firewall between the mediator and the Controller**.

See Chapter 13, “Setting Up Performance Center 8.1 to Use Mercury Diagnostics,” for more information about configuring Performance Center to use the Diagnostics components.

Part VIII

Configuring Diagnostics Metrics Collectors

25

Overview of Diagnostics Metrics Collectors

This chapter describes metrics capture and how to configure the metric collectors.

This chapter describes:	On page:
About Metrics Capture	453
Configuring Metric Collector Entries	454

About Metrics Capture

The metric collectors are installed with the probe. The collectors gather system metrics from the probe host and JMX metrics from the application server. The metric collectors are configurable so that you can control which metrics are collected.

Configuring Metric Collector Entries

The metric collectors for your probe installation are defined in the metrics configuration file. The properties and entries in the metrics configuration file, `<probe_install_dir>/etc/metrics.config`, allow you to control the metric collectors.

Note: When you update the metrics configuration file, the metric collectors automatically restart so the changes will take effect.

Understanding Metric Collector Entries

Metric Collector entries instruct the metric collectors to gather specific metrics. The parameters on the left hand side of the entry control how the probe gathers the metric from the host or the JVM, and the parameters on the right hand side of the entry define how the collected metrics are processed in Diagnostics and displayed in the user interface.

The entries can have one of the following layouts:

```
<collector_name>/<metric_config>=<metric_id>|<metric_units>|<category_id>
```

or

```
<collector_name>/<metric_config>=
RATE<rate_multiplier>(<metric_id>|<metric_units>|<category_id>)
```

where:

- **<collector_name>** indicates the name of the Diagnostics metric collector. The collectors are defined in **metrics.config**.

For system metrics the value of this parameter is **system**. For JMX metrics the value of this parameter is usually defined as the name of the application server type and the version, such as **WebSphere5**.

- ▶ **<metric_config>** identifies the metric that is to be monitored on the host system or on the JVM for the application server. The format of this parameter varies depending on whether you are creating an entry for a system metric or a JMX metric. For information on formatting the **metric_config** property for the system metric collector, see “Capturing Custom System Metrics” on page 462. For information on formatting the **metric_config** property for JMX metrics, see “Capturing JMX Metrics” on page 474.

- ▶ **RATE(...)** indicates that metric values are converted to a rate (units per second) during sampling.

For example, when the **Rate** parameter is used with the metric **total servlet requests since startup**, the value of the collected metric is converted from a *count of servlet requests* to the *number of servlet requests per second*.

When **Rate** is not used, omit the parenthesis as shown in the first example above.

Note: This parameter should only be used for metrics with non-decreasing values.

- ▶ **<rate_multiplier>** is an optional parameter that indicates that the rate is to be adjusted by multiplying the it by the **<rate_multiplier>**.

For example, when the **Rate** parameter and the **rate_multiplier** are used with the metric **total gc time** (in ms), the value of the metric collected is converted from *the total time for gc* to the *percent time spent in gc*.

- ▶ **<metric_id>** indicates the name that represents the metric in the UI. The **metric_id** must be unique in the **metrics.config** file. If the value of the **metric_id** is the same as one of the default metrics, Diagnostics replaces the **metric_id** in the entry with a standard name to be used to reference the metric in the UI. If the value of the **metric_id** is not the same as one of the default metrics, then the **metric_id** is used as the name of the metric in the UI exactly as shown in the entry.

- **<metric_units>** indicates the units of measure in which the metric is reported. This is a required parameter and it must contain one of the following units of measure:
 - microseconds, milliseconds, seconds, minutes, hours, days
 - bytes, kilobytes, megabytes, gigabytes
 - percent, fraction_percent
 - count
 - load
- **<category_id>** groups a set of metrics together under the same heading in the tree in the side bar of the Metrics tab in the J2EE Profiler. This parameter has no impact on the data displayed in the Details Table in the Diagnostics views.

Note: After you have created the metric collector entry, you must add the escape character “\” before each occurrence of a back-slash '\', space ' ', or colon ':'. This is a requirement for Java properties loaded from a file.

Example

The following example shows how to create the metric collector entry for a system metric. To create an entry for a system metric called CPU on a host platform, you would enter the following:

```
system/CPU = CPU|percent
```

where:

- **system** indicates that the metric is to be collected by the system metric collector
- the first **CPU** indicates that the metric known as **CPU** on the platform is being monitored

- ▶ the second **CPU** is the name that is to be used in the UI to label the metric
- ▶ **percent** indicates the units in which the metric is measured on the host, and reported in the UI

Starting Capture for a Metric

To gather information for an additional metric, add an entry for the metric to the appropriate metric collector in the **metrics.config** file using the template described in “Understanding Metric Collector Entries” on page 454.

Modifying a Metric that is Already Being Captured

You can update both the default and the custom metrics entries in the metric collectors in the **metrics.config** file.

Stopping Capture of a Metric

To stop a metric collector from collecting a metric listed in **metrics.config**, you can either delete the metric entry or make the metric entry a comment line by adding a **#** to the beginning.

26

Configuring System Metrics Capture

This chapter describes the process for capturing system metrics and how to configure the system metric collector to capture them.

This chapter describes:	On page:
About System Metrics	459
System Metrics Captured by Default	460
Configuring System Metric Collector	461
Capturing Custom System Metrics	462
Enabling z/OS System Metrics Capture	468

About System Metrics

The system metric collector is installed with the probe. The collector gathers system level metrics, such as CPU usage and memory usage, from the probe's host. The system metric collector is configurable so that you can control which of the system metrics are collected.

Only one instance of the system metric collector is run on a given host, no matter how many instances of the probe have been started on the host. When an instance of the probe is started, it attempts to connect to the UDP port specified in the metrics properties. If a connection is established, the system metric collector instance is started. If a connection cannot be made, then a system metric collector instance has already been started on the host by another instance of the probe and a new instance cannot be started.

Each probe periodically attempts to connect to the port to make sure that a system metric collector is always running. If the probe that started the systems metric collector is stopped, one of the other instances of the probe will start a new instance of the systems metric collector when it finds that the port is available.

System Metrics Captured by Default

The following are the system metrics that the metric collector collects by default for all supported platforms (excluding z/OS):

- CPU
- MemoryUsage
- VirtualMemoryUsage
- ContextSwitchesPerSec
- DiskBytesPerSec
- DiskIOPerSec
- NetworkBytesPerSec
- NetworkIOPerSec
- PageInsPerSec
- PageOutsPerSec

You can control which of the default system metrics the system metric collector gathers and you can add other platform specific metrics so that the collector gathers the information for them as well. See “Configuring System Metric Collector” on page 461 for more information. For certain platforms, such as Windows, Solaris, and Linux, you may create custom system metrics which can be gathered by the system metric collector. For details, see “Capturing Custom System Metrics” on page 462.

For information on z/OS system metrics see “Enabling z/OS System Metrics Capture” on page 468.

Configuring System Metric Collector

You can configure the system metrics capture process to run in your environment, and to collect and report the system metrics that are of interest to you, by modifying the properties in the metrics configuration file, `<probe_install_dir>/etc/metrics.config`.

Note: If you update the metrics configuration file, the systems metric collector automatically restarts so that your changes can take effect.

Modifying the Default Port

The default port for the metric collectors is 35000. This value can be modified using the `system.udp.port` property if the configuration for your probe host requires that another port be used.

To modify the default port:

- 1** Locate the `system.udp.port` property in `metrics.config`.
- 2** Change the value of the `system.udp.port` property to the number of the port that you want to be used by the system metric collector. The default port is 35000.

Note: The port assigned to the system metric collector is not related to the port for the probe's Web server.

For information on configuring the entries in the metric collectors see “Configuring Metric Collector Entries” on page 454.

Disabling System Metrics Collection

To disable the collection of system metrics so that they will not be collected or displayed in the UI, set the value of the `system.udp.port` property to -1.

Capturing Custom System Metrics

The Windows, Solaris, and Linux platforms allow you to create custom system metrics. The custom metrics can be monitored by the system metric collector.

The following sections provide instructions for creating the metrics and updating the entries in the system metric collector so that the custom metrics can be monitored.

Capturing Custom System Metrics on Windows Hosts

Using the features of Windows System Monitor, you can add counters to represent the performance of specific aspects of a system or service. The counters are tracked and reported in the Windows System Monitor, and can be monitored by the Diagnostics system metric collector.

To add counters using the Windows System Monitor:

1 Start the Windows Performance Monitor:

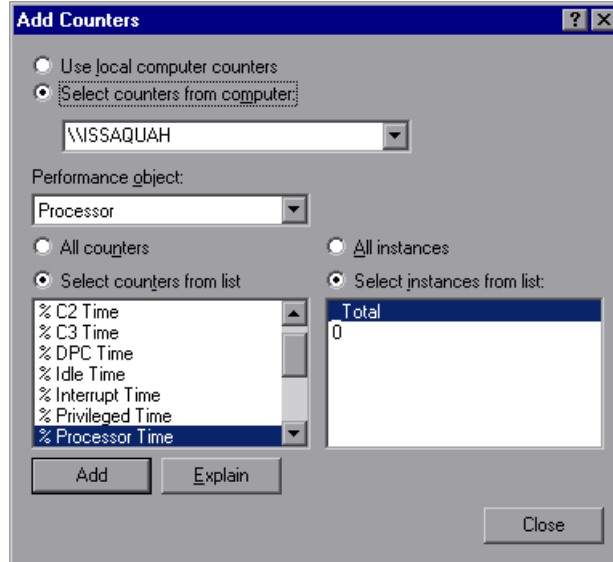
- a** Select **Start > Run** from the Start menu.
- b** In the **Open** box on the **Run** dialog box type 'perfmon'.

The **Performance** dialog box opens showing the **System Monitor** graph with a table of the current counters beneath the graph.

2 Display the Add Counters dialog box:

- a** Right-click the **System Monitor** graph and select **Add Counters...** from the pop-up menu.

Windows displays the **Add Counters** dialog box:



- 3** Ensure that the host computer is selected from **Select counters from computer** list.
- 4** In the **Performance object** list, select the object that the counter belongs to.
- 5** Choose **Select counters from list**, and select a counter from the list of counters that follows.
- 6** Choose **Select instances from list**, and select an instance from the list of instances that follows.
- 7** Click **Add**.

Once a counter has been added to the Systems Monitor, the system metric collector can be configured to gather the metrics for the counter. The following instructions will guide you through the steps to create an entry for the **metrics.config** based upon the following template:

```
<collector_name>/<metric_config>= <metric_id>|<metric_units>
```

This template is described in “Understanding Metric Collector Entries” on page 454.

To collect metrics for a Windows System Monitor Counter:

- 1 Open `<probe_install_dir>/etc/metrics.config`.
- 2 Create the `<metric_config>` part of the entry using the following template, type the entry for the counter:

```
\<performance_object>(<instance>)\<counter>
```

In the example shown in the preceding screen image:

- ▶ the selected Performance Object is `%Processor`
- ▶ the selected Instance is `_Total`
- ▶ the selected Counter is `Processor Time`

The `<metric_config>` portion of the entry that would be created for this example would be:

```
\Processor(_Total)\% Processor Time
```

- 3 Fill in the rest of the system metric entry template as shown in the following example:

```
system\Processor(_Total)\% Processor Time = ProcessorTime|percent
```

- 4 Format the initial entry by prepending a back-slash `'\'` before each occurrence of back-slash `'\'`, space `' '`, or colon `':'` in the initial entry.

Following this step, the initial entry in the previous step becomes:

```
system\\Processor(_Total)\\% Processor Time = ProcessorTime|percent
```

This is the correctly formatted entry for **metrics.config** to enable the system metric collector to gather the metrics for a Windows System Monitor counter.

```
system\\\\\\RemoteMachine\\Processor(_TOTAL)\\% Processor Time= Processor Time(Remote Machine)|percent
```

Note: Assuming **perfmon** is setup properly on a remote machine, you can use it to get metrics from remote machines by adding **\\MachineName** before the Performance object name as shown in the following example:

```
system\\\\RemoteMachine\\Processor(_TOTAL)\\%\\ Processor\\
Time=Processor\\ Time(Remote Machine)|percent
```

Capturing Custom System Metrics on Solaris Hosts

The Solaris system metrics that can be monitored by the system metric collector are found using the **kstat** command. Only a subset of the metrics found using the **kstat** command can be monitored by the system metric collector.

To collect metrics for a Solaris system metric:

- 1 Execute the **kstat** command and identify the metric that you want to monitor.

A Solaris system metric has the following format:

```
module:instance:name:statistic
```

For example:

```
vmem:35:ptms_minor:free
```

- 2 To cause the metric collector to gather the metrics for an additional system metric, add an entry for the metric to the system metric collector in the **metrics.config** file using the following template:

```
<collector_name>/<metric_config>= <metric_id>|<metric_units>
```

This template is described in “Understanding Metric Collector Entries” on page 454.

Using this template, the example from the previous step would initially appear as follows:

```
system/vmem:35:ptms_minor:free = Virtual Memory (35) Free | count
```

- 3 Format the initial entry by prepending a back-slash '\' before every back-slash '\', space ' ', or colon ':'.

Following this step the initial entry in the previous step becomes:

```
system/vmem\:35\:ptms_minor\:free = Virtual\ Memory\ (35)\ Free | count
```

This is the correctly formatted entry for **metrics.config** to enable the system metric collector to gather the metrics for a Solaris systems metric.

Capturing Custom System Metrics on Linux Hosts

The Linux system metrics that can be monitored by the system metric collector are found in the **/proc** file system. To configure the system metric collector to gather custom Linux metrics you must scan the **/proc** file system to locate the desired metric, and then create the system metric collector entry for the metric in **metrics.config** based upon the where the metric information is located.

To collect metrics for a Linux system metric:

- 1 Scan the **/proc** file system to locate the metric that you would like the Diagnostics system metric collector to monitor.

To create the system metrics configuration entry in **metrics.config** for the Linux metric, you must explicitly specify where the value for the system metric is located. The location is specified using the following values:

- **File name.** The name of the file where the metric information is located, including the path from the **/proc** directory.
- **Line offset.** A count of the number of lines in the file to the line where the system metric is located. The first line is counted as line 0.
- **Word offset.** A count of the number of words that the metric value is offset into the line in the file. The first word in the line is counted as line 0. The value at the specified offset must be an unsigned integer.

For example, if you wanted the system metric collector to monitor the SwapFree system metric so that you can see it displayed in the Diagnostics views, you would scan the `/proc` directory to locate the metric, and you would discover that the metric is located in the `meminfo` file. The layout of this file is as follows:

```
MemTotal: 515548 kB
MemFree: 1552 kB
Buffers: 41616 kB
Cached: 152084 kB
SwapCached: 46064 kB
Active: 402720 kB
Inactive: 75328 kB
HighTotal: 0 kB
HighFree: 0 kB
LowTotal: 515548 kB
LowFree: 1552 kB
SwapTotal: 1048568 kB
SwapFree: 779192 kB
Dirty: 4544 kB
Writeback: 0 kB
Mapped: 300056 kB
Slab: 28764 kB
Committed_AS: 801364 kB
PageTables: 3184 kB
VmallocTotal: 499704 kB
VmallocUsed: 2184 kB
VmallocChunk: 497324 kB
HugePages_Total: 0
HugePages_Free: 0
Hugepagesize: 4096 kB
```

The location of the SwapFree metric in this file would lead to the following values:

- **File name:** meminfo
- **Line offset:** 12
- **Word offset:** 1

- 2 To gather the metrics for an additional system metric, add an entry for the metric to the system metric collector in the **metrics.config** file using the following template:

```
<collector_name>/<line>:<word>:<file>= <metric_id>|<metric_units>
```

This template is a version of the template described in “Understanding Metric Collector Entries” on page 454. The `<metric_config>` property has been replaced with the properties `<line>:<word>:<file>`.

Using this template, the example from the previous step would initially appear as follows:

```
system/12:1:meminfo = Swap Free | kilobytes
```

- 3 Format the initial entry by prepending a back-slash `'\'` before every back-slash `'\'`, space `' '`, or colon `':'`.

Following this step the initial entry in the previous step becomes:

```
system/12\:1\:meminfo = Swap\ Free | kilobytes
```

This is the correctly formatted entry for **metrics.config** to enable the system metric collector to gather the metrics for a Solaris systems metric.

Enabling z/OS System Metrics Capture

The following system metrics can be collected for the z/OS platform:

- CPU
- DiskIOPerSec
- DiskBytesPerSec

System metrics are not captured by default, as this requires some system configuration changes. You must perform the following configuration steps in order to enable capture of z/OS system metrics.

To enable z/OS system metrics capture:

- 1** Change the permissions for the directory `<probe_install_dir>/bin/` to recursively allow execution. This can be done using a command like the following example:

```
chmod -R 770...
```

- 2** Change the permissions for the directory `<probe_install_dir>/bin/390-zos/systemmetrics` to allow execution. This can be done using a command as illustrated in the following example:

```
chmod -R 0+x ...
```

- 3** Start the RMF Monitor III and ensure that SMF record 70-79 is collecting.
- 4** Start the RMF Data Buffer on one or more systems in the sysplex.
- 5** Check the list of system names passed to the ERBDSQRY service.
- 6** Ensure that the system is collecting SMF record 92 with subtype 5.

27

Configuring JMX Metric Capture

This chapter describes the process for capturing JMX metrics and how to configure metric collectors to capture them.

This chapter describes:	On page:
About JMX Metrics	471
Configuring WebSphere for JMX Metric Collection	472
Configuring JMX Metric Collectors	474
Capturing Custom JMX Metrics	474

About JMX Metrics

The J2EE Probe comes with pre-defined JMX metric collectors that access the JMX metrics from the following application servers:

- ▶ IBM WebSphere 5.x, 6.x
- ▶ BEA WebLogic 6.x, 7.x, 8.x, and 9.x
- ▶ SAP NetWeaver 6.4
- ▶ Oracle AS 10g

The J2EE Probe can also collect JMX data from any J2EE server that supports the JMX standard. To collect JMX metrics from a different server, refer to Chapter 27, “Configuring JMX Metric Capture.”

The J2EE Probe runs the JMX metric collectors periodically to collect the metrics from the application server. The collected metrics are displayed on the user interfaces in both Mercury Diagnostics and J2EE Diagnostics Profiler.

Configuring WebSphere for JMX Metric Collection

You may need to configure the Performance Monitoring Service on the WebSphere server in order to start receiving JMX metrics. Instructions are provided below for both WebSphere 5.x and WebSphere 6.x.

The following are the instructions for configuring WebSphere 5.x for JMX metrics collection.

To configure WebSphere 5.x server for JMX metrics collection:

- 1** Open the WebSphere Administrative Console.
- 2** In the Console navigation tree, select **Servers > Application Servers**. The console displays a table of the application servers.
- 3** Click name of the application server that you want to configure from the Application Servers Table. The console displays the **Runtime** and the **Configuration** tabs for the selected application server.
- 4** Click the **Configuration** tab.
- 5** In the **Configuration** tab:
 - Click **Performance Monitoring Service**.
 - Select the **Startup** check box under General Properties.
 - Set the **Initial Specification Level** to **standard**.
 - Click **Save**.
- 6** Click **Apply** or **OK**.
- 7** If Java 2 Security is enabled on the application server, open the server policy file. (<WebSphere 5.x Installation Directory>/profiles/server.policy) and add the following security permission to the file, to allow JMX collection:

```
grant codeBase "file:<probe_install_dir>/lib/probe-jmx.jar"  
{ permission java.security.AllPermission; }
```

8 Restart the application server.

The following are the instructions for configuring WebSphere 6.x for JMX metrics collection.

To configure WebSphere 6.x server for JMX metrics collection:

- 1** Open the WebSphere Administrative Console.
- 2** In the Console navigation tree, select **Servers > Application Servers**. The console displays a table of the application servers.
- 3** Click name of the application server that you want to configure from the Application Servers Table. The console displays the **Runtime** and the **Configuration** tabs for the selected application server.
- 4** Click the **Configuration** tab.
- 5** In the **Configuration** tab:
 - ▶ Under the Performance heading, click **Performance Monitoring Infrastructure (PMI)**.
 - ▶ Under the General Properties heading, select the **Enable Performance Monitoring Infrastructure (PMI)** check box.
 - ▶ Under the Currently monitored statistic set heading select **Extended**.
- 6** Click **Apply** or **OK**.
- 7** If Java 2 Security is enabled on the application server, open the server policy file. (<WebSphere 6.x Installation Directory>/work/tools/ibm-6.0/websphere/appserver/profiles/default/properties/server.policy) and add the following security permission to the file, to allow JMX collection:


```
grant codeBase "file:/<probe_install_dir>/lib/probe-jmx.jar"
{ permission java.security.AllPermission; }
```
- 8** Restart the application server.

Configuring JMX Metric Collectors

The JMX metric collectors are configurable so that you can control which of the JMX metrics are collected. The JMX metric collectors are defined in the `<probe_install_dir>/etc/metrics.config` file. Typically a separate collector is defined for each major version of each application server.

For information on configuring the metric collectors see “Configuring Metric Collector Entries” on page 454.

Capturing Custom JMX Metrics

The J2EE Probe is installed with a number of predefined JMX metric collectors for the application servers listed in “About JMX Metrics” on page 471. You may modify or remove any of the JMX metric entries from the predefined collectors by following the instructions in “Configuring Metric Collector Entries” on page 454. You may also create entries in the existing metric collectors and even create new collectors if there are additional JMX metrics that you would like Diagnostics to monitor.

The following sections provide instructions creating new entries in the JMX metric collectors so that additional JMX metrics can be monitored.

Capturing JMX Metrics

The metric collectors installed with the J2EE Probe include entries for many of the JMX metrics that are available for each application server. However, there may be other JMX metrics that you would like to monitor or new metrics may be exposed by the application server vendor. The following instructions will guide you through the process of creating the JMX metric entries based upon the following template:

```
<collector_name>/<metric_config>= <metric_id>|<metric_units>
```

This template is described in “Understanding Metric Collector Entries” on page 454.

To capture JMX metrics:

- 1** Open `<probe_install_dir>/etc/metrics.config`, and locate the JMX metric collector that is appropriate for the application that is being monitored by the J2EE Probe.
- 2** The `<collector_name>` parameter is the same as the rest of the entries in the collector. If you were creating an entry for WebLogic, the value of this parameter would be WebLogic.
- 3** Create the `<metric_config>` parameter using the following template:

```
<MBean object name pattern>.<attribute name>
```

For JMX metrics the `<metric_config>` parameter is a pattern that the collector uses to find a matching MBean. The pattern consists of two components, separated by the '.' character:

- ▶ `<MBean object name pattern>` is the string representation of the object name of an MBean
- ▶ `<attribute_name>` is the name of the attribute that represents the metric.

For example, for a WebLogic application server, the `<metric_config>` parameter for the throughput of all **Execute Queues** is configured as:

```
*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount
```

For an explanation of metric patterns see “Understanding Metric Patterns” on page 476.

- 4** Fill in the rest of the JMX metric entry template as shown in the following example:

```
WebLogic/*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount =  
RATE(Execute Queues Requests / sec|count|Execute Queues)
```

- 5 Format the initial entry by prepending a back-slash '\' before every back-slash '\', space ' ', equals (=), or colon ':'.
Following this step the initial entry in the previous step becomes:

```
WebLogic/*:Type\=ExecuteQueueRuntime,*.ServicedRequestTotalCount =  
RATE(Execute Queues Requests / sec|count|Execute Queues)
```

This is the correctly formatted entry for a JMX metric collector to enable the collector to gather a WebLogic JMX metrics.

Understanding Metric Patterns

For JMX metrics the <metric_config> parameter is a pattern that the collector uses to find a matching MBean. For example:

```
*:Type=ExecuteQueueRuntime,*.ServicedRequestTotalCount
```

In the example above, the object name is ***:Type=ExecuteQueueRuntime,***, which could actually resolve to many MBeans whose names have the **Type** component equal to **ExecuteQueueRuntime**. **ServicedRequestTotalCount** is an attribute name for which metric values will be collected by the JMX metric collector.

Note: Current implementation of the JMX collector only supports attributes that are numeric in type (i.e., long, integer, etc.).

The JMX metric collector first uses MBeanServer's query mechanism to find the matching MBeans for each object name provided in the configuration. For JMX metrics the object names are a pattern that the collector uses to find a matching MBean. For more details around the object names, see <http://java.sun.com/j2ee/1.4/docs/api/javax/management/ObjectName.html>.

Since MBean object names are patterns that can resolve into multiple MBeans, the JMX collector will validate all of the attribute names in the entry against all MBeans that match the pattern, and will aggregate the attribute values over the set of those matching MBeans. Of course, it is not always the case that the object name resolves into multiple MBeans. For example, the following object name resolves to a single MBean (on a WebLogic application server):

```
*\:Name\=weblogic.kernel.Default,Type\=ExecuteQueueRuntime,  
*.ServicedRequestTotalCount
```


Part IX

Appendixes

A

Diagnostics Server Administration Page

This appendix describes how to access the Diagnostics Server Administration page, where you can configure Diagnostics properties. It also describes how to view the Configuration pages, and modify the properties within these pages.

This appendix describes:	On page:
Accessing the Diagnostics Server Administration Page	481
Configuring Diagnostics Using the Diagnostics Server Configuration Pages	483

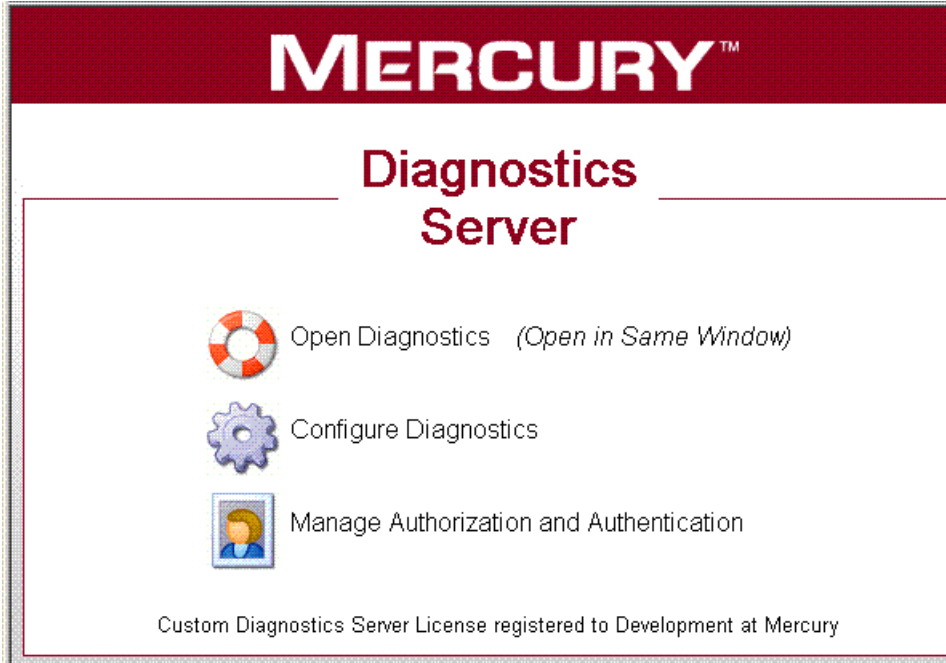
Accessing the Diagnostics Server Administration Page

You can set the user privileges, configure Diagnostics settings, and open Diagnostics directly from the administration page of the Diagnostics Server.

To use the Diagnostics Server administration page:

- 1 Open the administration page by navigating to http://<diagnostics_server_host>:2006 in your browser, or by selecting **Start > Programs > Mercury Diagnostics Server > Administration**. The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.

The Diagnostics Server administration page opens in your browser.



2 Click the option for the activity that you want to perform.

- ▶ **Open Diagnostics.** Opens the Diagnostics Web pages where you can view the performance metrics collected by the probes that are reporting to the Diagnostics Server. The performance metrics are displayed in the standard Diagnostics views.

For more information about opening Diagnostics directly from the Diagnostics Server in Commander mode, see the *Mercury Diagnostics User's Guide*.

- ▶ **Configure Diagnostics.** Opens the Components page, which has a link to the Diagnostics Server Configuration page.

For more information about configuring the Diagnostics properties, see “Configuring Diagnostics Using the Diagnostics Server Configuration Pages” on page 483.

- **Manage Diagnostics Users.** Opens the User Administration page where you can add and maintain security information, and user privileges for specific users.

For more information about security and user privileges, see “User Authentication and Authorization” on page 489.

Configuring Diagnostics Using the Diagnostics Server Configuration Pages

In the Diagnostics Server Configuration pages, you set the property values that control how the Diagnostics Server communicates with the other Diagnostics components, and how it processes the data that it receives from the probes.

Note: To ensure that you are entering valid property values, it is recommended that you use the configuration pages to modify the Diagnostics Server properties, rather than editing the property files directly.

Accessing Diagnostics Server Configuration Pages

You access the Diagnostics Server Configuration pages from the Components page.

To access the Diagnostics Server configuration pages:

- 1** Open the administration page by navigating to http://<diagnostics_server_host>:2006 in your browser, or by selecting **Start > Programs > Mercury Diagnostics Server > Administration**. The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.

The Diagnostics Server administration page opens in your browser.

- 2** Click **Configure Diagnostics**.

- 3 If you have not already signed into the Diagnostics Server, you are prompted for a user name and password. This must be a valid user name, and must have both **View** and **Change** privileges. For information about valid user names and privileges, see Appendix B, “User Authentication and Authorization.”

Note: Diagnostics continues to prompt for a user name and password until valid credentials are entered.

If you click **Cancel**, the following error message is displayed in your browser: **Access denied. You must specify a valid user name and password.**

If you entered a valid user name and password, but do not have the proper privileges, the following error message is displayed in your browser: **Access denied. You do not have the required permission to view this screen.**

To log on as a different user to the one you are currently logged on as, you need to close your browser and re-open it.

After you have logged on, the Diagnostics Server Components page opens:

Component Name	Component Description
registrar	Central list of all Diagnostics component deployments
security	User Management
logging	Configure log files and logging details.
monitor	Current Metrics
configuration	Configuration
license	License Management

Mercury Diagnostics Server "server-square", version 4.1.48.646

4 Click **configuration** to access the Diagnostics Server Configuration page.

Configuration	
Name	Description
Customer Information	Properties that are used to configure the customer information for the Server.
Alert Properties	Properties that are used to configure the alert settings for the Server. All changes take effect dynamically, without server restart.
Component Communications	Properties that are used to configure how this Diagnostics Component communicates with the other diagnostic components.
Memory Diagnostics	Properties that configure the way that the Server will handle memory diagnostics
Online Cache	Properties that are used to configure the Server's Online Cache.
Logging	Configure log files and logging details.
Show Advanced Options	
Mercury Diagnostics Server	

Click the link to the page whose properties you want to update. You can configure:

- Customer information
- Alert properties
- Component Communication properties
- Memory Diagnostics properties
- Online cache properties
- Logging level details

Viewing Advanced Diagnostics Server Configuration Options

The configuration options displayed on the Configuration page are the commonly configured options. The more advanced configuration options are hidden by default.

Important: Do not manipulate the advanced options without the guidance of your Mercury Customer Support representative.

To display the advanced options:

At the bottom of the page, click **Show Advanced Options**.

The list of options on the page is updated to include the advanced configuration options, and the link changes to **Hide Advanced Options**.

To hide the advanced options:

At the bottom of the page, click **Hide Advanced Options**.

The list of options on the page is updated so that the advanced configuration options are no longer visible, and the link changes to **Show Advanced Options**.

Modifying Diagnostics Server Properties

You modify the properties of the Diagnostics Server from the Configuration page.

To modify the Diagnostics Server properties:

- 1 Access the Diagnostics Server Configuration page, as described in “Accessing Diagnostics Server Configuration Pages” on page 483.
- 2 Click the link to the properties (for example, **Component Communications**) that you want to update.
- 3 Review the properties that are displayed and revise any values that should be updated.

MERCURY			
Component Communications			
Name	Value	Description	Default Value
Commander URL	<input type="text" value="http://localhost:2006"/>	The web address for the Diagnostics Commander.	http://localhost:2006
Registered Host Name	<input type="text"/>	The hostname to register with in the Diagnostics Commander Registrar. This hostname will be used by the Commanding Server to reconnect back to the Distributed Server when used in a Load Runner environment. If no hostname is specified, the Server will register with the default system hostname.	
Diagnostics Commander Proxy Host	<input type="text"/>	The hostname or IP address of the proxy that should be used to communicate with the Diagnostics Commander. Leave blank for no proxy.	
Server/Commander Proxy Port	<input type="text"/>	The port of the proxy that should be used to communicate with the Diagnostics Commander.	80
Diagnostics Commander Proxy Protocol	<input type="text"/>	The protocol (either HTTP or HTTPS) to connect to the proxy for the Diagnostics Commander.	http
Probe Data Port	<input type="text" value="2612"/>	The Server port where it receives event data from the probe.	2612
Server Webserver Listen Address	<input type="text" value="0.0.0.0"/>	The local address the Local Server listens on for HTTP requests from the Commanding Server. The default is set to listen on all local addresses.	0.0.0.0
Server Webserver Port	<input type="text" value="2006"/>	The port where the Local Server listens on for HTTP requests from the Commanding Server.	2006

- 4 When you are satisfied with your changes, click **Submit** to save them. Otherwise click **Reset All** to cancel the changes you made.
- 5 A message appears at the top of the page to indicate that your changes were saved.

Note:

- For most properties that you update, a message is displayed reminding you to restart the Diagnostics Server. The property changes will not take effect until you restart the Diagnostics Server.

If you want make other changes to the Diagnostics Server properties, you should finish making all of your changes before restarting the Diagnostics Server.

Restarting the server will result in a small loss of data (up to 6 minutes). You should therefore schedule restarts at a time that is convenient.

- Modifying the logging level details does not require restarting the Diagnostics Server; however, it may take up to a minute for your changes to be applied.
-

B

User Authentication and Authorization

This appendix introduces the Mercury Diagnostics authentication and authorization process and describes how to create and maintain user security permissions.

This appendix describes:	On page:
About User Authentication and Authorization	490
Understanding User Privileges	491
Accessing Diagnostics Using Default User Names	492
Understanding the Diagnostics Server Permissions Page	493
Creating, Editing and Deleting Users	496
Assigning Privileges Across the Diagnostics Deployment	498
Assigning Privileges for Probe Groups	499
Authentication and Authorization for Users of Integrated Mercury Products	502
Tracking User Administration Activity	503

About User Authentication and Authorization

User authentication and authorization settings for all the Diagnostics components are configured in the Diagnostics Server in Commander mode.

You manage authentication and authorization by creating and editing user names and granting the users privileges so that users are able to perform the functions within the application for which they are responsible.

User permissions and privileges for the Profilers (.NET Diagnostics Profiler or J2EE Diagnostics Profiler) of the probes connected to a particular Diagnostics Server are also defined in the Diagnostics Server in Commander mode. You can assign users one set of permissions for accessing Profilers in a particular probe group and a different set of permissions for accessing the Diagnostics Server.

You manage users and assign user privileges in the Diagnostics Server Permissions page.

Important: When you install the probe as a profiler only (not connected to any Diagnostics Server), you manage the authentication and authorization of users of the Profiler in the probe itself.

For information about managing authentication and authorization for the J2EE Probe installed as a profiler only, see “Authentication and Authorization for J2EE Profilers in Standalone Mode” on page 412.

For information about managing authentication and authorization for the .NET Probe installed as a profiler only, see “Authentication and Authorization for .NET Profilers in Standalone Mode” on page 355.

Before you can view any Diagnostics data, or make any changes to the Diagnostics configuration or user privileges, you must log on to the Diagnostics Server in Commander mode using a user name that has valid security access with the appropriate privileges.

After logging on to the Diagnostics Server in a particular browser session, the user name remains in effect until the browser session ends. When you are finished using Diagnostics, be sure to close your browser to prevent others from accessing Diagnostics using your privileges.

Understanding User Privileges

The following privilege levels can be assigned to Diagnostics users:

Privilege	Description
View	The user can view Diagnostics data from the UI.
Execute	The user can make changes to the settings on the UI, such as changing thresholds or adding comments. On the Profiler, this privilege gives permission to perform garbage collection and clear the performance data held by the Profiler.
Change	The user can access the Configure Diagnostics menu to alter component configuration, view system health, and maintain user information. On the profiler, this gives permission to run potentially risky operations, such as taking a heap-dump or changing instrumentation.

Note: The privilege levels, **rhttpout** and **system** are for internal purposes only. **rhttpout** is used to grant the user access to the rhttp/out URL for doing remote management of distributed servers.

system is an internal permission generally granted only to the **mercury** special user. It is the permission that allows Diagnostics components to talk to one another (for example, the permission required for a probe to register with the Diagnostics Server).

Each privilege level stands alone. There is no inheritance of privileges from one level to the next. You must grant a user all of the privilege levels that are necessary to perform the functions that they need to perform.

For example, a user must be granted both **View** and **Execute** privileges in order to be able to make changes to thresholds. A user name that has been granted only **Execute** privileges would not be useful, as it would not allow the user to see the UI on which they have permission to make changes.

For information about assigning privileges to users, see “Assigning Privileges Across the Diagnostics Deployment” on page 498.

Accessing Diagnostics Using Default User Names

The following default user names have been defined for Diagnostics:

Default User Names	Privileges	Description
user	View	Can only view the data from the UI.
superuser	View, Execute	Can view data, change thresholds, and create alerts and comments from the UI.
admin	View, Change, Execute, System	Can view data, change thresholds, and create alerts and comments from the UI. Can configure components and maintain user information.

You can use these default user names to access Diagnostics functionality.

The passwords for the default user names are the same as the user names. For example, for the user name **admin**, the password is **admin**.

You can modify the password or privileges for the default user names to suit your needs. You can also define new user names in order to control user access to Diagnostics.

Important: There are two default users, **mercury** and **bac**, that are used for internal purposes and should never be modified. These users are for internal communication between components.

Understanding the Diagnostics Server Permissions Page

You manage users and assign user privileges in the Permissions page.

This section includes:

- ▶ “Accessing the Permissions Page” on page 493
- ▶ “The Permissions Page at a Glance” on page 495

Accessing the Permissions Page

In the Permissions page, you can create, edit, and delete Diagnostics users and manage the permissions for specific user names.

You can access the Permissions page from the Diagnostics Server Administration page or from the Maintenance page.

To access the Permissions page from the Diagnostics Server Administration page:

- 1** In your browser, open the Diagnostics Server Administration page by navigating to http://<commanding_diagnostics_server_host>:2006, or by selecting **Start > Programs > Mercury Diagnostics Server > Administration**.

The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.

- 2** In the Diagnostics Server Administration page, click **Manage Authorization and Authentication** to open the Permissions page. opens, where you can create, edit, and delete Diagnostics users and manage the permissions for specific user names.

To access the Permissions page from the Diagnostics UI:

- 1** Click **Maintenance** at the top the Diagnostics views. The Maintenance page opens.
- 2** Click **security** to open the Permissions page.

When the Permissions page opens, if you have not already signed into the Diagnostics Server, you are prompted for a user name and password. You must have at least **View** privileges in order to view your privileges, and modify your password. To add or delete users, or update user privileges, you must have both **View** and **Change** privileges.

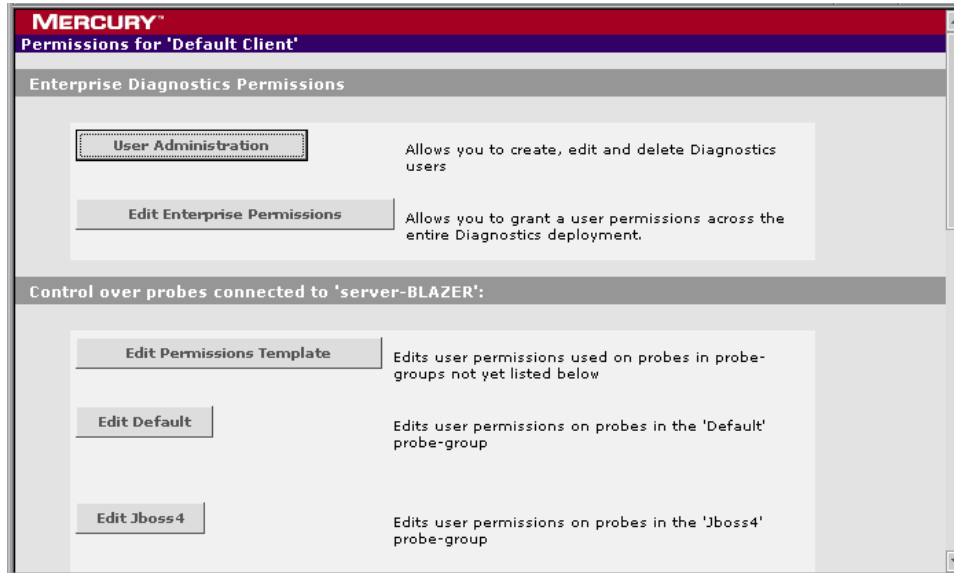
Note: Diagnostics continues to prompt for a user name and password until valid details are entered.

If you click **Cancel**, the following error message is displayed in your browser: **Access denied. You must specify a valid username and password.**

If you entered a valid user name and password, but do not have the proper privileges, the following error message is displayed in your browser: **Access denied. You do not have the required permission to view this screen.**

The Permissions Page at a Glance

The following screen is an example of the Diagnostics Server Permissions page:



The Permissions page is divided into the following two sections:

- **Enterprise Diagnostics Permissions.** In this section you manage Diagnostics users and you assign privileges across the whole Diagnostics deployment, including the Diagnostics Servers and probes.

By default, if users are authorized to access a particular Diagnostics Server, they also have the same authorization (and privileges) to access all probes connected to that server.

- **Control over probes connected to <commanding_Diagnostics_server>.** In this section you assign privileges for users accessing the probe Profilers. You can assign users one set of permissions for accessing Profilers in a particular probe group and a different set of permissions for accessing the Diagnostics Server.

Creating, Editing and Deleting Users


Users with both **View** and **Change** privileges can create new users, edit the password for an existing user, or delete users. Users with only **View** privileges can maintain their own password.

To create a new user:

- 1 Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2 On the Permissions page, click **User Administration** to open the User Administration page.
- 3 On the User Administration page, click **Create User**.
- 4 In the **New User Name** box, type a user name for the new user and click **OK**. The new user appears in the list of user names.
- 5 Under **Change Password**, in the **Password** box, type a password for the new user, and confirm it by retyping it in the **Confirm Password** box.
- 6 In the **Password for <current user>** box, type the password of the user currently logged on.
- 7 Click **Save Changes**.

By default the new user, has **view** privileges. For information about changing the privileges assigned to the user, see “Assigning Privileges Across the Diagnostics Deployment” on page 498.

To delete a user:

- 1 Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2 On the Permissions page, click **User Administration** to open the User Administration page.
- 3 On the User Administration page, in the **Password for <current user>** box, type the password of the user currently logged on.
- 4  Click the **Delete user** button corresponding to the user you want to delete.
- 5 A message box opens asking if you want to delete the selected user. Click **OK** to delete the user.

To change a user's password if you have View and Change privileges:

- 1** Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2** On the Permissions page, click **User Administration** to open the User Administration page.
- 3** On the User Administration page, in the row representing the relevant user, type the new password in the **Password** and **Confirm Password** boxes.
- 4** In the **Password for <current user>** box, type the password of the user currently logged on.
- 5** Click **Save Changes** to save all the changes you made to the different user names.

To change your own password if you only have View privileges:

- 1** Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2** On the Permissions page, click **User Administration** to open the User Administration page.
- 3** On the User Administration page, type the new password in the **Password** and **Confirm Password** boxes.
- 4** In the **Old Password** box, type your old password.
- 5** Click **Save Changes**.

Assigning Privileges Across the Diagnostics Deployment

Users with both **View** and **Change** privileges can grant users privileges across the entire Diagnostics deployment.

Note: For a description of the user privileges that you can assign to Diagnostics users, see “Understanding User Privileges” on page 491

To assign user privileges across the entire Diagnostics deployment:

- 1 Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2 On the Permissions page, click **Edit Enterprise Permissions** to open the Editing Enterprise Permissions page.

The Editing Enterprise Permissions page is an editable page which allows you to modify user privileges.

- 3 Locate the name of the user, whose privileges you want to modify.
-

Important: You add users on the User Administration page, as described in “Creating, Editing and Deleting Users” on page 496.

- 4 Add the privileges to the username as comma separated values.

For example, if you have defined a user by the name of **newuser** and you want to assign this user with view and execute privileges you would locate **newuser** and edit the line so that it appears as follows:

```
newuser = view,execute
```

The Editing Enterprise Permissions page also includes a set of default users. These users are described in “Accessing Diagnostics Using Default User Names” on page 492. You can modify the privileges of these default users.

Assigning Privileges for Probe Groups

Users with both **View** and **Change** privileges can grant users privileges for accessing the probe Profilers belonging to particular probe groups.

By default, if users are authorized to access a particular Diagnostics Server, they also have the same authorization (and privileges) to access all probe Profilers connected to that server.

However, you can assign users a different set of permissions for different probe groups than what they have for the Diagnostics Servers themselves.

Note: For a description of the user privileges that you can assign to Diagnostics users, see “Understanding User Privileges” on page 491.

You can modify user privileges for each probe group individually and you can also modify a Permissions template that defines the user privilege settings for all future probe groups added to your system.

For each probe group, there are three default user groups of users with certain privileges. You can choose to comment out these groups or to modify their privileges. The following groups of users are defined by default in all the probe groups:

user group	permissions
any_diagnostics_admin	This group refers to any user with administration (change) privileges on the Diagnostics Server. By default, any user who falls into this category and does not have any other predefined permission settings has administration permissions for all probes connected to that server.
any_diagnostics_superuser	This group refers to any user with superuser (execute) privileges on the Diagnostics Server. By default, any user who falls into this category and does not have any other predefined permission settings has execute permissions for all probes connected to that server.
any_diagnostics_user	This group refers to any user with user (view) privileges on the Diagnostics Server. By default, any user who falls into this category and does not have any other predefined permission settings has view permissions for all probes connected to that server.

To assign user privileges for accessing a particular probe group:

- 1** Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2** In the **Control over probes connected to <commanding_Diagnostics_server>** section of the permissions page, click **Edit <name of probe group>**.
The Editing Permissions page opens. This is an editable page which allows you to modify user privileges.

- 3 Enter the username to which you want to assign unique privileges and add the privileges to the username as comma separated values.

For example, if you have defined a user by the name of **newuser** and you want to assign this user with view and execute privileges on this particular probe group, you enter the following line:

```
newuser = view,execute
```

To assign user privileges using the Permissions template:

- 1 Access the Diagnostics Server Permissions page as described in “Accessing the Permissions Page” on page 493.
- 2 In the **Control over probes connected to <commanding_Diagnostics_server>** section of the permissions page, click **Edit Permissions Template**.

The Editing Template Permissions page opens. This is an editable page which allows you to modify user privileges.

- 3 Enter the username to which you want to assign unique privileges and add the privileges to the username as comma separated values.

For example, if you have defined a user by the name of **newuser** and you want to assign this user with view and execute privileges on this particular probe group, you enter the following line:

```
newuser = view,execute
```

You could also modify or comment out one of the user group settings defined in the template.

Important: All future probe groups that are connected to your Diagnostics Server will inherit the user privilege settings from this Permissions template.

Authentication and Authorization for Users of Integrated Mercury Products

Diagnostics can be integrated with other Mercury applications (Mercury Business Availability Center, Performance Center, or LoadRunner). Privileges for users. This section describes how authentication and authorization works for users of these integrated products and includes the following sections:

- ▶ “Authentication and Authorization for Mercury Business Availability Center Users” on page 502.
- ▶ “Authentication and Authorization for Performance Center and LoadRunner Users” on page 503

Authentication and Authorization for Mercury Business Availability Center Users

In Mercury Business Availability Center, you can define user permissions for Diagnostics. For more information, see “Assigning Permissions for Diagnostics Users” on page 219.

When an existing or new Mercury Business Availability Center user opens Diagnostics from Mercury Business Availability Center, their permissions are picked up from the Mercury Business Availability Center session and copied into the Diagnostics permissions system (under the MMS customer, if relevant).

Updates to Mercury Business Availability Center user permissions are only picked up when the user opens Diagnostics. (If Diagnostics is already open, changes will not be detected until it is closed and re-opened).

Mercury Business Availability Center passwords are never sent to Diagnostics—Diagnostics trusts a successful Mercury Business Availability Center login.

If privileges of Mercury Business Availability Center users change, the changes are not picked up until that user re-opens Diagnostics.

If a Mercury Business Availability Center user is deleted, it is recommended that you manually remove their permissions from Diagnostics. For more information, see “Creating, Editing and Deleting Users” on page 496.

Note: The Diagnostics Server can take up to five minutes to detect permission changes to users.

Authentication and Authorization for Performance Center and LoadRunner Users

When you set up both LoadRunner or Performance Center for integration with Diagnostics, you specify the Diagnostics Server details within LoadRunner / Performance Center. These details include the username and password with which you log on to Mercury Diagnostics.

When you access Diagnostics from LoadRunner or Performance Center, you are logged into Diagnostics with that same username and password that you specified during the integration setup.

Users accessing Diagnostics from within LoadRunner or Performance Center, will therefore have the privileges that are associated with the username that was specified during the integration setup.

Tracking User Administration Activity

Each time a user enters the Diagnostics Server User Administration page, all activity that takes place is logged in the following log file:
<diagnostics_server_install_dir>\log\server-useradmin.log.

The data logged in the file includes the date and time of each action performed, a description of the action, and the name of the user performing the action.

To view the log file:

- 1** Open the Diagnostics Server administration page in one of the following ways:
 - ▶ By selecting **Start > Programs > Mercury Diagnostics Server > Administration**.
 - ▶ By navigating to http://<diagnostics_server_host>:2006 in your browser. The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.

The Diagnostics Server administration page opens.

- 2** Click **Configure Diagnostics**.
- 3** If you have not already signed into the Diagnostics Server, you are prompted for a user name and password. This must be a valid user name, and must have both **View** and **Change** privileges. For information about valid user names and privileges, see “Understanding User Privileges” on page 491.

Note: Diagnostics continues to prompt for a user name and password until valid credentials are entered.

If you click **Cancel**, the following error message is displayed in your browser: **Access denied. You must specify a valid user name and password.**

If you entered a valid user name and password, but do not have the proper privileges, the following error message is displayed in your browser: **Access denied. You do not have the required permission to view this screen.**

The Diagnostics Server Components page opens.

- 4** Click **logging**. The logging page opens.
- 5** Click **View Log Files**. A list of log files appears.
- 6** Click the `<diagnostics_server_install_dir>\log\server-useradmin.log` link.

The log file is displayed at the bottom the page.

C

Using the System Health Monitor

This appendix describes how to use the System Health Monitor to review the configuration of the Mercury Diagnostics components and verify that they are working properly.

This appendix describes:	On page:
Introducing the System Health Monitor	506
Accessing the System Health Monitor	506
Understanding the System Health Monitor	509
Viewing Component Status and Host Configuration Tooltips	514
Viewing Detailed Component Information	516
Viewing Troubleshooting Tips	520
Viewing Log Information for the Whole System	520
Customizing the System Health Monitor Display	521
Filtering the System Health Monitor Component Map by Customer	523
Controlling the Refresh Rate of the System Health Monitor	524
Creating and Using System Health Monitor Snapshots	525

Introducing the System Health Monitor

The System Health Monitor provides you with a map of all of the components of your Mercury Diagnostics deployment, and gives you real-time information and health status for each component. At a glance, you can determine which components are experiencing problems, the load on each component, and the amount of data flowing between components.

By drilling down to the System Health Monitor component map, you can reveal the details about the configuration, performance metrics, and processing logs for the components. You can also find troubleshooting tips for handling many of the issues that are revealed in the System Health Monitor component map. In many cases, the System Health Monitor will be your first and only stop when you need to know information about the components in your Diagnostics deployment and the machines that host them.

You can also capture a snapshot of the System Health Monitor information in an .xml file so that you can share it with others who may be able to help you diagnose the issues that you see on the map.

Accessing the System Health Monitor

You can access the System Health Monitor directly from your Web browser, from Performance Center, from the LoadRunner Controller, or from Mercury Business Availability Center.

To access the System Health Monitor from your Web browser:

- 1 Enter the following URL into your browser:

```
http://<Diagnostics Server in Commander mode host>:<Diagnostics Server in  
Commander mode port>/registrar/health
```

The default port for the Diagnostics Server in Commander mode is 2006. If you have configured the Diagnostics Server to use an alternate port, use that port number in the URL.

- 2 Enter your user name and password. For more information about user names and passwords, see Appendix B, “User Authentication and Authorization.”

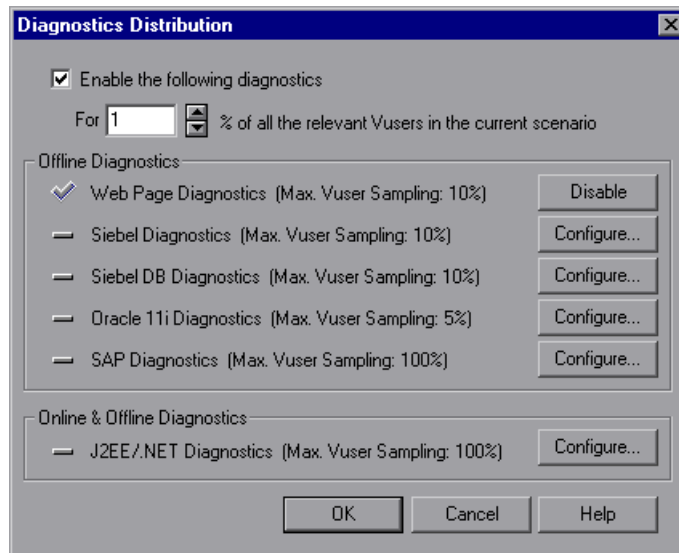
The System Health Monitor opens in your browser.

To access the System Health Monitor from Mercury Business Availability Center:

- 1 Select **Admin > Diagnostics**.
- 2 Click the **System Health** tab. The System Health information is displayed.

To access the System Health Monitor from LoadRunner Controller:

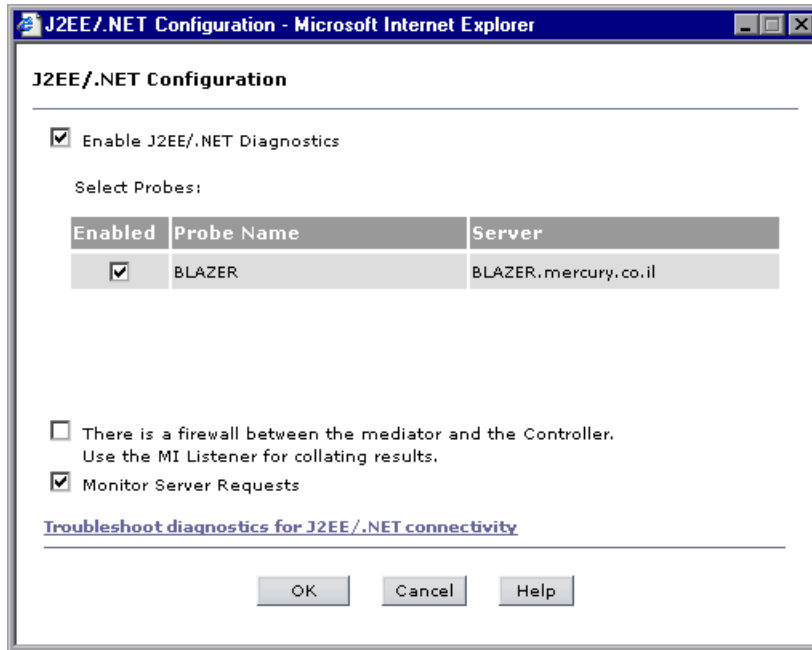
- 1 Choose **Diagnostics > Configuration** from the menu bar in the Controller Window. The Diagnostics Distribution dialog box opens.
- 2 In the Online and Offline Diagnostics section of the Diagnostics Distribution dialog box, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.



- 3 Click the **Troubleshoot Diagnostics for J2EE/.NET connectivity** link. The System Health Monitor is displayed in a new browser window.

To access the System Health Monitor from Performance Center:

- 1 In the Load Tests page, click the **Diagnostics** tab, and select the **Enable Diagnostics** check box to enable Diagnostics.
- 2 In the **Offline and Online Diagnostics** section, click **Configure** to open the J2EE/.NET Configuration dialog box.



- 3 Click the **Troubleshoot Diagnostics for J2EE/.NET connectivity** link. The System Health Monitor is displayed in a new browser window.

Troubleshooting Accessing the System Health Monitor

If you are unable to access the System Health Monitor verify that:

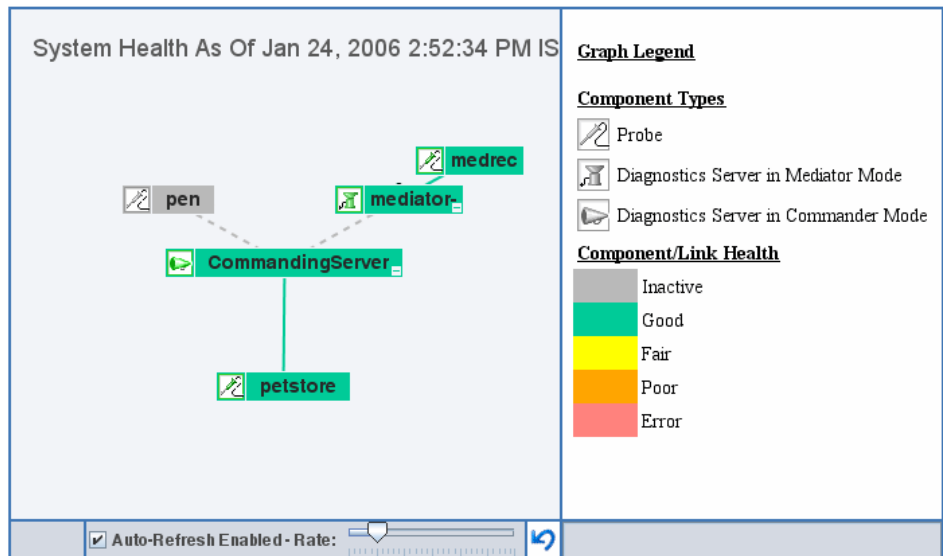
- the machine name for the **Diagnostics Server** host is correct.
- the port number for the **Diagnostics Server** is correct. The default port is **2006**.
- the **Diagnostics Server** started successfully, and is running on the host.

Understanding the System Health Monitor

The System Health Monitor is comprised of two panes:

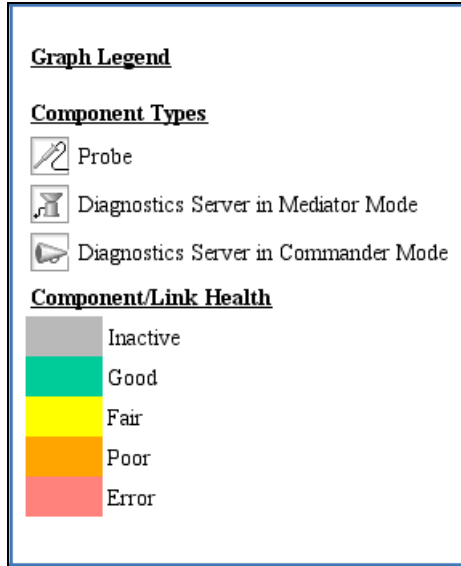
- **The graph legend** (in the right pane)
- **The System Health Monitor component map** (in the left pane)

When the System Health Monitor opens you see a deployment map of the Mercury Diagnostics components, accompanied by the graph legend.



The Graph Legend

The graph legend, to the right of the component map, helps you to interpret the information displayed in the component map.



The System Health Monitor Component Map

The System Health Monitor component map displays high-level information about your Diagnostics deployment and how well the components are performing. The System Health Monitor component map displays the following:

- Icons that represent each of the Diagnostics components that you have correctly configured as part of your Diagnostics environment.
- The color of the component icon that indicates the health of the component.
- An indicator of the amount of load that the component is processing.

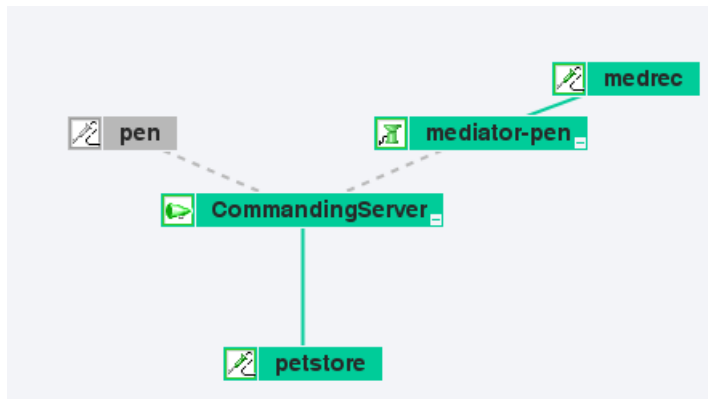
- ▶ Links between the component icons, which indicate the state of the communication links between components.

The color of the link indicates the status of the link.

- ▶ A solid line means data is flowing across the link.
- ▶ A dashed line means there is currently no data flowing.

Component Icons in the System Health Monitor Component Map

In the following image of a Diagnostics deployment, one Diagnostics Server is deployed in Commander mode and one Diagnostics Server is deployed in Mediator mode. There are Diagnostics probes reporting to both Diagnostics Servers.



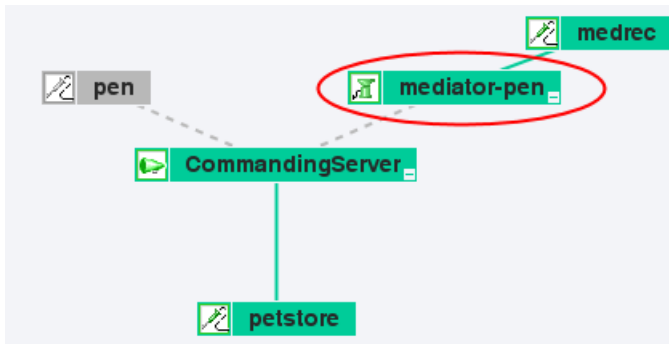
Diagnostics Server in Commander mode

When a Diagnostics Server is deployed in Commander mode, it has both commanding and mediating responsibilities. A Diagnostics Server in Commander mode is represented in the System Health Monitor component map by an icon labelled **CommandingServer**.



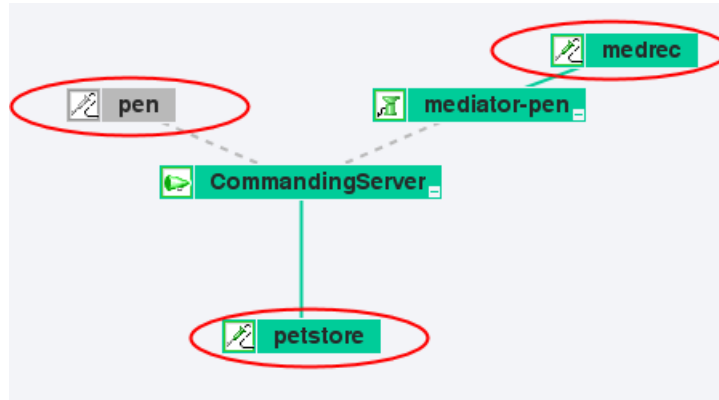
Diagnostics Server in Mediator mode

When a Diagnostics Server is deployed in Mediator mode, it only has mediating responsibilities. A Diagnostics Server in Mediator mode is represented by an icon labelled **mediator-<name of host>**.



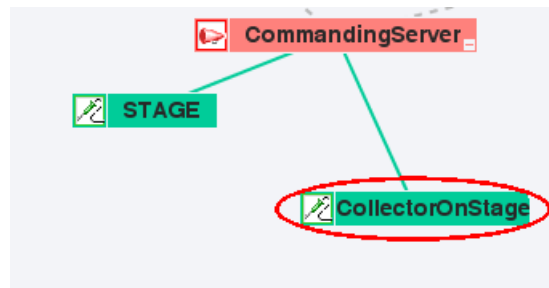
Diagnostics Probes

The System Health Monitor component map displays all of the probes that are reporting to each particular Diagnostics Server. A probe is represented by an icon labelled <name of probe>. The name of the probe is assigned during the installation procedure.



Diagnostics Collector

The System Health Monitor component map displays all of the Diagnostics Collectors that are reporting to each particular Diagnostics Server. A Collector is represented by an icon labelled <name of Collector>. The name of the collector is assigned during the installation procedure.



Viewing Component Status and Host Configuration Tooltips

You can view tooltips for each component on the System Health Monitor component map. The tooltips provide you with basic status and host configuration information.

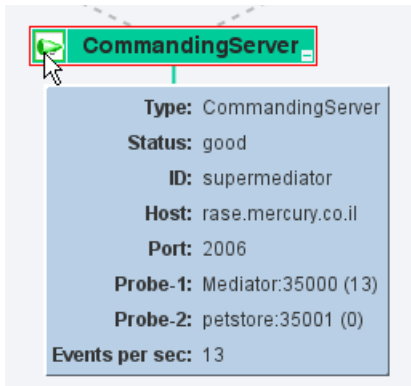
To view component status and host configuration tooltips:

Hold the mouse pointer over the component until a tooltip is displayed.

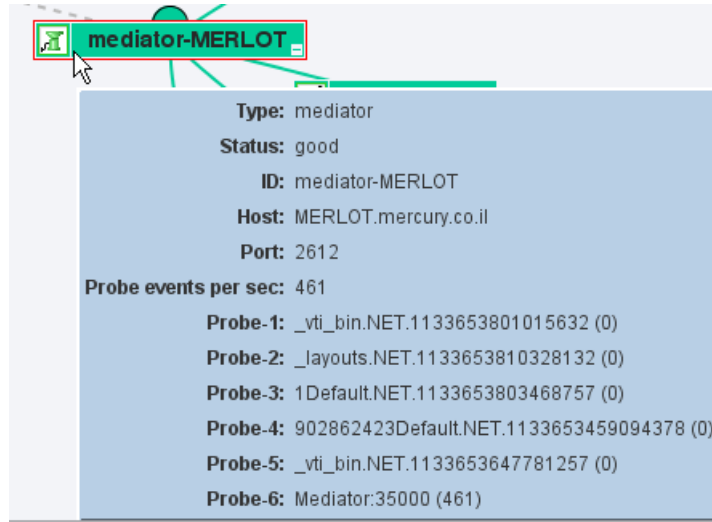
The examples that follow, show tooltips displaying component status and host configuration information for each different component.

The information in the tooltip for the Diagnostics Server components includes a list of the probes that have been configured to work with the Diagnostics Server.

► Diagnostics Server in Commander mode



► Diagnostics Server in Mediator mode

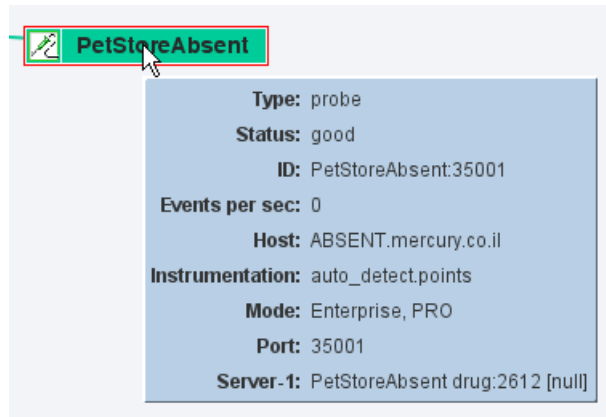


The screenshot shows a tooltip for a component named 'mediator-MERLOT'. The tooltip is a light blue box with a white border, containing the following information:

- Type:** mediator
- Status:** good
- ID:** mediator-MERLOT
- Host:** MERLOT.mercury.co.il
- Port:** 2612
- Probe events per sec:** 461
- Probe-1:** _vti_bin.NET.1133653801015632 (0)
- Probe-2:** _layouts.NET.1133653810328132 (0)
- Probe-3:** 1Default.NET.1133653803468757 (0)
- Probe-4:** 902862423Default.NET.1133653459094378 (0)
- Probe-5:** _vti_bin.NET.1133653647781257 (0)
- Probe-6:** Mediator:35000 (461)

► Diagnostics Probe

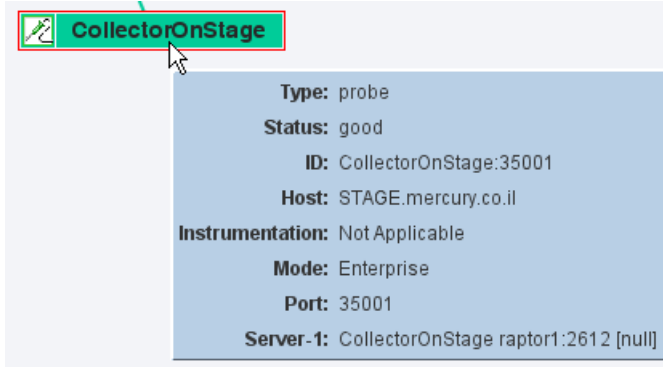
The information in the tooltip for a probe component includes the name of the Diagnostics Server that the probe is working with (**Server** field), and the mode that the probe has been configured to work in (**Mode** field). The tooltip also directs you to the relevant instrumentation points file.



The screenshot shows a tooltip for a component named 'PetStoreAbsent'. The tooltip is a light blue box with a white border, containing the following information:

- Type:** probe
- Status:** good
- ID:** PetStoreAbsent:35001
- Events per sec:** 0
- Host:** ABSENT.mercury.co.il
- Instrumentation:** auto_detect.points
- Mode:** Enterprise, PRO
- Port:** 35001
- Server-1:** PetStoreAbsent drug:2612 [null]

► **Diagnostics Collector**

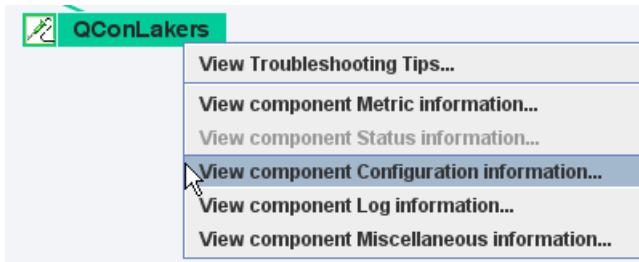


Viewing Detailed Component Information

You can view detailed information about the configuration and performance of each component in the System Health Monitor component map. The information is displayed in the Component Monitor detail table.

To view detailed component information:

Right-click the component and choose the relevant item from the menu.



Alternatively, double-click the component icon and scroll to the relevant section.

You can view the following information:

- ▶ **Component Metric information:** Lists the processing metrics for the selected component.
- ▶ **Component Configuration information:** Lists the component configuration for the selected component.
- ▶ **Component Log information:** Lists the log messages for the selected component.
- ▶ **Component Status Information:** Provides further information about the performance status of the component, where relevant.
- ▶ **Component Miscellaneous information:** Provides miscellaneous information about the component.

Component Metric Information

The following image shows a section of the Component Metric information for the Diagnostics Server in Commander mode:

Component Monitor Update As Of Dec 7, 2005 2:21:12 PM GMT+02:00	
Metric	
Events since launch	<unknown>
Last update (commander time)	Wed Dec 07 04:22:35 PST 2005
Up time	1 day(s) 0 hour(s) 4 minute(s) 53 second(s)
WDE bytes relayed	166836
WDE bytes relayed since launch	92396096
WDE samples failed	75
WDE samples failed since launch	58870
WDE samples relayed	171
WDE samples relayed since launch	94676

Component Configuration Information

The following image shows the Component Configuration information for the .NET Probe:

Component Monitor Update As Of Dec 7, 2005 2:27:12 PM GMT+02:00	
⊗ Configuration	
Host	ABSENT.mercury.co.il
IP Address	10.168.4.112
Install dir	D:\dotnet_1702\
Instrumentation	ASP.NET.points,MSPetShop.points
LAN ID	DotNetRealProbes
Lightweight Memory Diagnostics	ON
Mode	AM,PRO
Port	35001
System	ABSENT (Microsoft Windows NT 5.2.3790.0)
VM	.NET 1.1.4322.2032
Version	4.0.114.102

Component Log information

The following image shows the Component Log information for the Diagnostics Server in Mediator mode:

Component Monitor Update As Of Dec 7, 2005 2:34:02 PM GMT+02:00	
⊗ Log	
Wed Dec 07 04:06:30 PST 2005	WARNING: real user fragment 31289557 timed out
Wed Dec 07 03:09:02 PST 2005	WARNING: real user fragment 32930633 timed out
Wed Dec 07 03:08:49 PST 2005	WARNING: real user fragment 32687513 timed out
Wed Dec 07 01:08:38 PST 2005	WARNING: real user fragment 8834978 timed out
Tue Dec 06 23:08:41 PST 2005	WARNING: real user fragment 12640068 timed out
Tue Dec 06 05:51:44 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.I
Tue Dec 06 05:51:44 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.I
Tue Dec 06 05:46:01 PST 2005	SEVERE: out of order com.mercury.diagnostics.server.persistence.impl.I

Component Status Information

Below is an example of Component Status information for the J2EE Probe. This information is available when the status of the component drops below **Good**.

Component Monitor Update As Of Dec 7, 2005 3:13:20 PM GMT+02:00	
⊗ Status	
Error	Probe reporting communication problems with mediator: myDomain6 alol:2612 [null]

Component Miscellaneous Information

The following image shows the Component Miscellaneous information for the J2EE Probe:

Component Monitor Update As Of Dec 7, 2005 2:42:42 PM GMT+02:00	
⊗ Miscellaneous	
ProbeProtocolVersion	3

Viewing Troubleshooting Tips

Troubleshooting information is available for each component in the System Health Monitor component map.

To view tips for troubleshooting Diagnostics components:

Right-click the relevant component and choose **View Troubleshooting Tips**. Troubleshooting information for the selected component is displayed.

Scan the information displayed for the symptoms that you are investigating, to see if there is a recommended solution documented in the troubleshooting tips.

Viewing Log Information for the Whole System

You can view the log messages for all of the components in the System Health Monitor on one screen.

To view log information for all the components:

Right-click anywhere in the System Health Monitor component map (except on a component icon) and choose **View Log History**. The log for all component activity is displayed.

The screenshot shows a window titled "Log History" with a subtitle "Component Monitor Update As Of Dec 7, 2005 3:32:32 PM GMT+02:00". The window contains a table with three columns: Time, Component Name, and Message. The table lists several warning messages from various components like mediator-MERLOT, QConLakers, and Raptor1Medrec.

Time	Component Name	Message
Wed Dec 07 05:27:44 PST 2005	mediator-MERLOT	WARNING: real user fragment 32317
Wed Dec 07 05:27:44 PST 2005	mediator-MERLOT	WARNING: real user fragment 16040
Wed Dec 07 05:26:13 PST 2005	mediator-MERLOT	WARNING: WARN!! [RangeSocketL
Wed Dec 07 05:25:53 PST 2005	mediator-MERLOT	WARNING: WARN!! [RangeSocketL
Wed Dec 07 05:18:00 PST 2005	QConLakers:35000	WARNING: [redirect] PdhGetFormatt
Wed Dec 07 05:11:39 PST 2005	Raptor1Medrec:35000	WARNING: WARN!! [RangeSocketL
Wed Dec 07 05:09:43 PST 2005	Raptor1Medrec:35000	WARNING: WARN!! [RangeSocketL
Wed Dec 07 05:09:37 PST 2005	Raptor1Medrec:35000	WARNING: WARN!! [RangeSocketL

Customizing the System Health Monitor Display

You can customize the way information is displayed on the System Health Monitor.

Manipulating the Appearance of the System Health Monitor Component Map

You can manipulate the System Health Monitor component map to display any area of the tree, to display expanded branches or collapsed branches, and to change the overall panorama of the image.

To manipulate the System Health Monitor component map display:

- ▶ Click a component to move it to the center of the map.
- ▶ Click anywhere in the component map to change the emphasis of the map. The components will center around the point where the mouse was clicked.
- ▶ Click and drag anywhere in the component map to move and rotate the components around that point.
- ▶ Expand and collapse the branches of a sub-tree by clicking the expand (+) or collapse (-) symbols, displayed on the lower-right corner of the component icon.
- ▶ Increase or reduce the gap between each branch by holding down the ALT button on the keyboard and dragging in the component map.

Displaying the Graph Legend

You can choose whether or not to display the graph legend on the System Health Monitor.

To show the graph legend:

If the graph legend is not displayed, right-click anywhere in the System Health Monitor component map (except on a component icon) and choose **Show Graph Legend**.

To hide the graph legend:

If the graph legend is displayed, right-click anywhere in the System Health Monitor component map (except on a component icon) and choose **Hide Graph Legend**.

Displaying Load

You can choose the way in which load is displayed on the System Health Monitor. You can choose between displaying the load indicator as a circle that appears behind the component icon, or as a bar scale that appears below the component icon.

To display load as a circle:

When load is displayed as a bar scale, right-click anywhere in the System Health Monitor component map (except on a component icon) and choose **Show Load Using Circles**.

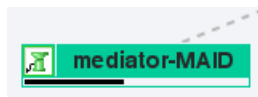
In the following example, the circle behind the component represents the amount of load.



To display load as a bar scale:

When load is displayed as circles, right-click anywhere in the System Health Monitor component map (except on a component icon) and choose **Show Load Using Scale**.

In the following example, the bar scale below the component represents the amount of load.

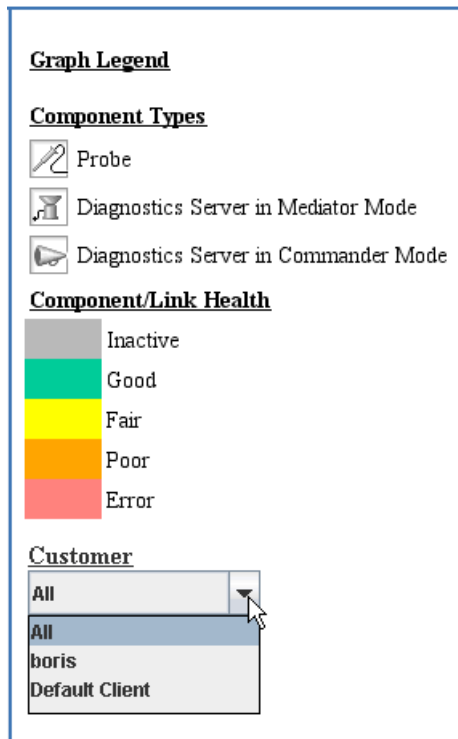


Filtering the System Health Monitor Component Map by Customer

When you are running Diagnostics in an Mercury Managed Services environment with multiple customers, you can filter the information that is displayed in the System Health Monitor so that only the information for a selected customer is displayed in the graph.

By default, the System Health Monitor will display the components for all of the customers.

When more than one customer is represented in the Diagnostics deployment displayed in the System Health Monitor, a list of customers is displayed in the Graph Legend box.



Select the customer whose components you would like to see displayed in the System Health Monitor, or select **All** to display the components for all customers at once.

Controlling the Refresh Rate of the System Health Monitor

By default, the System Health Monitor is configured to automatically refresh the information displayed. You can change the auto-refresh frequency, or completely disable the auto-refresh feature. You can also request a manual refresh at any time.

The controls at the bottom of the System Health Monitor that are used to control the refresh of the information displayed, are shown below.



To disable the automatic refresh feature:

Clear the **Auto-Refresh Enabled** check box.

To enable the automatic refresh feature:

Select the **Auto-Refresh Enabled** check box.

To adjust the auto-refresh rate:

Slide the **Rate** slide control to the right to decrease the auto-refresh frequency and to the left to increase the frequency.

To refresh the display manually:

Click the **Refresh** button located to the right of the slide bar.

Creating and Using System Health Monitor Snapshots

You can take snapshots of the System Health Monitor to share the information with others. You can also import snapshots that others have taken of their System Health Monitors to learn information about their systems.

Exporting a Snapshot of the System Health Monitor

You can export the information displayed on System Health Monitor to an .xml formatted file so that you can share the information with others. This is especially useful when you need help diagnosing a problem. The information in the System Health .xml file can be viewed in .xml format or can be imported into any working copy of the System Health Monitor.

Note: Remember! People who have access to the Diagnostics Server in Commander mode host can view the System Health Monitor directly using their Web browser if you give them the URL.

To export a System Health Monitor snapshot as an .xml file:

- 1 Enter the following URL in your Web browser:

http://<Diagnostics Server in Commander mode host>:<Diagnostics Server in Commander mode port>/registrar/xml

The default port is 2006. If you have configured the Diagnostics Server to use an alternate port, use that port number in the URL

- 2 Using the **Save** menu option in your Web browser, save the Web page to a file. Be sure to give the file a name that will help you remember why you saved the snapshot. For example, to remember that you took a snapshot of a System Health Monitor with a poor performing Diagnostics Server in Mediator mode, you may want to name the file:

sys_health_mediator_yyyymmdd.xml

Importing a Snapshot of a System Health Monitor

If someone gives you an .xml snapshot of a System Health Monitor, you can view it from an existing System Health Monitor.

To import a System Health Monitor snapshot:

- 1 Copy the System Health snapshot file to a directory on the Diagnostics Server in Commander mode host machine.
- 2 Enter the following URL in your Web browser:

`http://<Diagnostics Server in Commander mode host>:<Diagnostics Server in Commander mode port>/registrar/health?xml=<XML_file>`

where <XML_file> is the path to the .xml file on the Diagnostics Server in Commander mode host machine.

For example, if you copied `sys_health_mediator_yyyymmdd.xml` to the hard drive (C:) on the Diagnostics Server in Commander mode host machine, whose name is `jake`, the URL might look like this:

`http://jake:2006/registrar/health?xml=c:\sys_health_mediator_yyyymmdd.xml`

The <Diagnostics Server in Commander mode port>, by default, should be 2006. If you have configured the Diagnostics Server to use an alternate port, use that port number in the URL.

When the System Health Monitor is displayed in your Web browser, the information is from the imported .xml file.

D

Diagnostics Data Management

This appendix provides a detailed description of how Mercury Diagnostics data is managed and stored.

This appendix describes:	On page:
About Diagnostics Data	528
Custom View Data	528
Performance History Data	530
Data Storage Size & Data Retention	533
Pre-Installation Data Management Considerations	535
Backing Up Diagnostics Data	536
Handling Diagnostics Data when Upgrading Diagnostics	539

About Diagnostics Data

There are two main types of Diagnostics data. The first consists of the custom views that each user has created. The second consists of the Diagnostics data collected by the probes and aggregated by the Diagnostics Servers. This data is stored in a time-series database on the Diagnostics Servers. Each Diagnostics Server stores the data that is collected by the probes that report to it. In addition, the Diagnostics Server in Commander mode stores the virtual transactions' data both for AD and AM. The organization and maintenance of the data files that make up the data base are described in the following discussion.

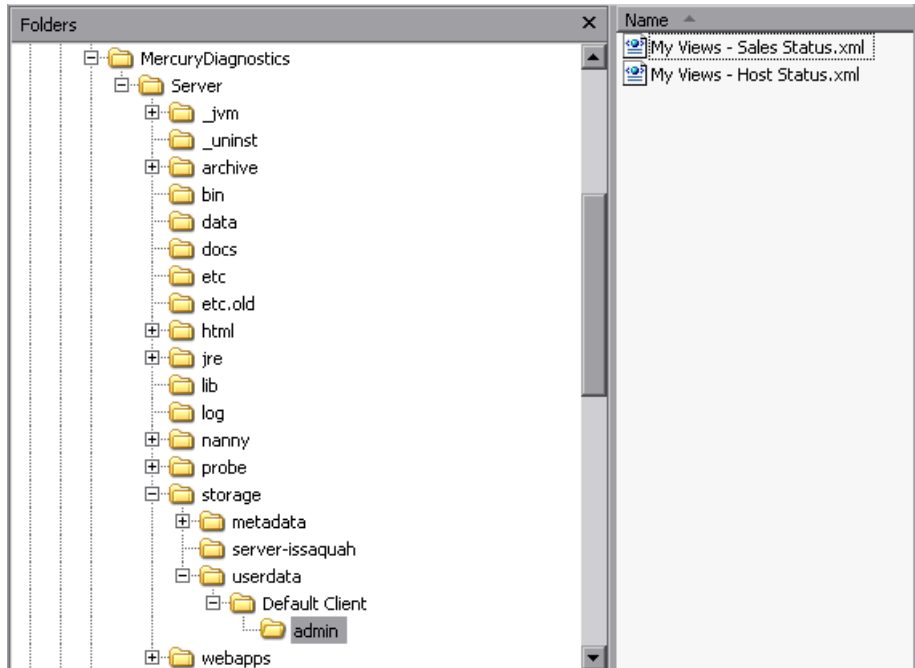
Custom View Data

Diagnostics users can create and save customized views as described in the chapter, "Customizing Diagnostics Views," in the *Mercury Diagnostics User's Guide*. Diagnostics stores the customized views as XML files on the host for the Diagnostics Server in Commander mode.

Custom View Data Organization

The user defined custom views are stored as XML files in the `<diagnostics_server_install_dir>/storage/userdata` directory on the host for the Diagnostics Server in Commander mode. The custom view files are relatively small with a maximum size of approximately 100 K each.

Each user that has defined a custom view has their own custom view sub-directory in the **userdata** directory. For example, if the 'admin' user created two custom views, Sales Status and Host Status, the two views would be stored as separate xml files in the `<diagnostics_server_install_dir>/storage/userdata/Default Client/admin` directory on the Diagnostics Server in Commander mode as shown in the following example.



Performance History Data

Diagnostics stores the historical performance data in a time series database on the Diagnostics Server in Mediator mode. If the Diagnostics Server has numerous probes reporting to it, the stored historical performance data can grow to many gigabytes of data. Although the amount of data collected for each application can vary in size, it is recommended that you plan for approximately 3 GB of data for each virtual machine that you are monitoring. For more information, see “Data Storage Size & Data Retention” on page 533.

Performance History Data Organization

The Diagnostics performance history data is in the `<diagnostics_server_install_dir>/archive/mediator-<host_name>/persistence/<customer_name>_` directory of the Diagnostics Server in Mediator mode where:

- ▶ `<host_name>` is the name of the host for the Diagnostics Server in Mediator mode
- ▶ `<customer_name>` is the customer name that you entered when you installed the Diagnostics Server in Mediator mode. Note that the name of this directory is the customer name with an appended underscore.

Note: Unless you are an MMS customer, the customer name should always be Default Client.

Performance History Data File Types

The Diagnostics performance history data is stored in five types of files:

Symbol Table Files

The symbol tables contain string-to-integer mappings for small and fast data encoding of the other data files. For example, `/login.do` might be encoded as `1347854`.

The symbol tables are stored in two files in the `<diagnostics_server_install_dir>/archive/mediator-<host_name>/symboltable` directory:

Per-time-Period Performance Summary Files

In the Diagnostics views, data can be viewed in yearly, quarterly, monthly, weekly, daily, 6-hourly, 1-hour, 20-minute, and 5-minute periods. Each viewable time period is stored in separate *summary* files. For example, over the course of one week, the following summary files would be created:

- ▶ 1 weekly summary file
- ▶ 7 daily summary files
- ▶ 28 six-hourly summary files
- ▶ 168 hourly summary files
- ▶ 504 twenty-minute summary files
- ▶ 2,016 five-minute summary files

Each summary file is named according to the minute (in GMT) at which Diagnostics started storing summary data in it. For example, a summary file that contained data beginning at midnight (GMT) on November 23, 2006 would be named **2006_11_23_0.summary**.

The summary files are stored in sub-directories based upon the duration of the data that they contain. The subdirectories are **Weeks/7d**, **Days/1d**, **Days/6h**, **Hours/1h**, **Hours/20m**, **Hours/5m**. For example, a summary file for one of the daily time periods would be stored in the sub-directory `<diagnostics_server_install_dir>/archive/mediator-<host_name>/persistence/<customer_name>_ /Days/1d`.

The data from the summary files are used to create the graphs and tables shown in the Diagnostics views.

Per-Major-Time-Period Performance Trend Files

Diagnostics stores the charted trend data for each major time period in *trend* files. The major time periods are months, weeks, days, and hours. The trend files are named according to the minute (in GMT) at which the trended data that they contain was captured. For example, a file that contained hourly trend data starting at midnight November 23, 2006 would be named as follows: **Hours/1h/ 2006_11_23_0.trend**.

The trend files are accessed when the user selects an element to be charted on the UI. To retrieve a trend for a minor time period, such as 5 minutes, only a small portion of the data in a major time period trend file is read.

Per-Major-Time-Period Instance Tree Files

The *instance tree* files are similar to the trend files; there is a corresponding dump of the collected instance call trees for each major time period. For example, **Hours/1h/2005_11_23_0.tree**.

Instance tree files are accessed when the user drills down to an instance call tree on the Diagnostics UI.

Work Journal File

Diagnostics stores a work journal in `<diagnostics_server_install_dir>/archive/mediator-<host_name>/persistence/<customer_name>_<run_name>/aggregationMarks`. The *journal* contains the name of summary files that have been rolled up into the summary trend and instance tree files for larger time periods.

Data Storage Size & Data Retention

Diagnostics uses data compression and data retention strategies that allow it to optimize its use of disk storage.

Data File Compression

Diagnostics performance data for an application is stored in multiple files that represent different time periods. However, Diagnostics only *writes* new diagnostics data to the files that represent the *current* time period. This means that most of the Diagnostics data files for an application can be stored as read-only files

Since the historical data files are quite large they are automatically compressed after each time period is complete in order to save disk space. Compressed files have the same files names with a **.zip** extension.

Data File Retention

In order to make optimum use of disk space, historical performance data is stored in tiers with the data in each tier retained based upon the diagnostics data retention policy. The data in the tiers with lower resolution data points is kept for longer periods of time to assist with such activities as capacity planning. The data in the tiers with high resolution data points is kept for shorter periods of time to assist with such activities as performance diagnostics. For this retention policy, measurements have shown that Diagnostics uses approximately 3 GB for each probed virtual machine.

The Diagnostics data retention policy is illustrated in the table below:

Directory	Data is kept for ...	Viewable Periods	Trend Resolution
Years	5 Years	Year, Quarter	1 Day
Months	24 Months	Month	6 Hours
Weeks	52 Weeks	Week	1 Hour
Days	93 Days	Day & 6 Hours	5 minutes
Hours	72 Hours	Hour, 20 minutes & 5 minutes	5 seconds

Data File Retention and Trended Data in the Diagnostics Views

When Diagnostics displays trended metrics in the Diagnostics views, it attempts to show approximately sixty data points for each viewable period so that the trend that is presented will be meaningful and easy to understand. To arrive at the data points needed for each viewable period, the Diagnostics Server averages the data from the appropriate tier in the data files. For example, when you are looking at trended data for the last hour, one data point per minute is shown in the graph. These data points would have been created by averaging raw data points for twelve 5-second time periods.

Pre-Installation Data Management Considerations

When preparing to install and configure a large Diagnostics Server, you should consider the following performance tuning recommendations:

- ▶ For maximum performance, the Diagnostics Server should be installed onto an empty or recently defragmented disk. The archive directory should be stored on that same disk. Alternatively, the archive directory could be mounted on an empty or recently defragmented disk.

Note: It is recommended that the disk used for diagnostics time series database that is stored in the `<diagnostics_server_install_dir>/archive` not be used for other disk activity (i.e. don't mount the `<diagnostics_server_install_dir>/archive` directory on the same disk that is used for your system files, temporary files etc...)

- ▶ To reduce fragmentation over time and increase system performance a separate disk (or partition) dedicated to the archive directory is recommended.
- ▶ Intensive background disk processes (such as disk de-fragmentation or virus scans) should be disabled on the disk where the archive directory is stored.
- ▶ Network file systems such as NFS or Samba should not be used.

Note: The better the raw performance of the disk that you dedicate to the archive directory of the Diagnostics Server, the more load the Diagnostics Server can handle. Ask your system administrator to make sure the disk mounted for the archive directory is a high-performance disk or array.

Backing Up Diagnostics Data

It is recommended that you back up the Diagnostics data regularly so that it can be restored in the case of a disk or system failure.

Backing up Data Remotely

Remote backup is possible by downloading the Diagnostics data files over HTTP to a local directory, and then backing up that directory using your normal backup procedures.

The Diagnostics Server supports the HTTP If-Modified-Since and Request-Range headers ("rget") to allow standard HTTP mirroring software to download or incrementally update these files.

Diagnostics is installed with a remote backup script stored at `<diagnostics_server_install_dir/bin/remote-backup.sh`. The remote backup script is a Unix script that uses the wget utility (<http://www.gnu.org/software/wget/wget.html>) to download over HTTP incrementally. On Windows, Cygwin (<http://www.cygwin.com/>) can be used to run this script.

The following table lists the `remote-backup.sh` parameters:

Parameter	Description
<code>-h</code>	The host (or IP address) to download from
<code>-o</code>	The directory to store the backup in
<code>-u</code>	The HTTP username to use Default: admin
<code>-p</code>	The HTTP password to use Default: admin
<code>-P</code>	The HTTP port number to use (optional) Default: 2006
<code>-r</code>	The ID of the Diagnostics Server in Mediator mode being read from (for rhttp backups)(optional)

For example, to backup a Diagnostics Server running on the dragonfly machine into the **dragonfly-backup** directory:

```
% mkdir dragonfly-backup
% bin/remote-backup.sh -u admin -p secret -h dragonfly -o dragonfly-backup
```

The data is backed up in the following directories:

Data	Backup Directory
Server configuration	etc/
User custom views	storage/userdata
Raw performance history data	archive/.../persistence/
Symbol table	archive/.../symboltable/

Restoring Data After a Failure

The files in the backup directory are stored in the structure used by the Diagnostics Server.

To restore the time series database from the backup:

- 1** Install a clean Diagnostics Server. The Diagnostics Server is started automatically after the installation completes.
- 2** Shut down the Diagnostics Server.
- 3** Ensure that the Diagnostics Server has been shut down by verifying that there are no java/javaw processes in your process list. On Windows systems, you can use the Task Manager to do this and on UNIX systems, you can use ps.
- 4** Delete the `<diagnostics_server_install_dir>/archive` directory from Diagnostics Server.
- 5** Copy the database backup to replace the `<diagnostics_server_install_dir>/archive`.

- 6 If the host name for the Diagnostics Server has changed since the backup was taken you must update the directory name that is based upon the Diagnostics Server host name to reflect the new host name.

Rename `<diagnostics_server_install_dir>/archive/mediator-<host-name>` so that `<host-name>` reflects the new Diagnostics Server host name. For example, if host name in the backup was `oldhost` and the new host name is `newhost` you would change `<diagnostics_server_install_dir>/archive/mediator-oldhost` to `<diagnostics_server_install_dir>/archive/mediator-newhost`

Index Regeneration

When a restored Diagnostics Server is first started, the indexed data, which was not backed up, must be regenerated. Index regeneration is started automatically in the background and could take several hours to complete. While the indexes are regenerated the Diagnostics Server is able to receive events from probes; but some historical data cannot be displayed in the Diagnostics views until the restoration is complete.

Known Limitation

In Mercury Diagnostics version 6.5, binary data is written in the native byte order. This means that a Diagnostics data backup from a Solaris machine (Big Endian) cannot be restored and used on a x86 machine (Windows/Linux -- Little Endian).

Note: The backup script supplied with the Diagnostics Server backs up the data in a specific order. Failing to back up files in the correct order causes the restored backup to be unusable. It is therefore recommended to always use the supplied script to create data backups.

Handling Diagnostics Data when Upgrading Diagnostics

When you upgrade your Diagnostics Server from 4.0 or 4.1 to 4.2 you must follow the following data upgrade instructions in order to be able to continue to use the custom views and the Diagnostics data that was stored on the old Diagnostics Server.

Upgrading Performance History Data

The following instructions enable you to use the performance history data from the 4.0 or 4.1 version of the Diagnostics Server in the 4.2 Diagnostics Server.

To move your Diagnostics data from an older Diagnostics Server:

- 1** Shut down the current Diagnostics Server.
- 2** Ensure that the Diagnostics Server has been shut down by verifying that there are no java/javaw processes in your process list. On Windows systems, you can use the Task Manager to do this and on UNIX systems, you can use ps.
- 3** Create a backup copy of the <diagnostics_server_install_dir>/archive directory and the <diagnostics_server_install_dir>/storage/userdata directory in a location where they will not be impacted by the uninstall of the existing Diagnostics Server or the installation of the new Diagnostics Server.
- 4** Uninstall the existing Diagnostics Server as instructed in “Uninstalling the Diagnostics Server” on page 50.
- 5** Install the new Diagnostics Server. The 4.2 Diagnostics Server is started automatically after the installation has finished.
- 6** Shut down the 4.2 Diagnostics Server.
- 7** Ensure that the Diagnostics Server has been shut down by verifying that there are no java/javaw processes in your process list. On Windows systems, you can use the Task Manager to do this and on UNIX systems, you can use ps.
- 8** Delete the <diagnostics_server_install_dir>/archive directory for the 4.2 Diagnostics Server.

- 9 Copy the backup copy of the old `<diagnostics_server_install_dir>/archive` that you created into the 4.2 `<diagnostics_server_install_dir>/archive`.
- 10 If the host name for the 4.2 Diagnostics Server is different than the host name for the old Diagnostics Server, you must update the directory name that is based upon the Diagnostics Server host name to reflect the new host name.

Rename `<diagnostics_server_install_dir>/archive/mediator-<host-name>` so that `<host-name>` reflects the 4.2 Diagnostics Server host name. For example, if your old host name was `oldhost` and the 4.2 host name is `newhost` you would change

`<diagnostics_server_install_dir>/archive/mediator-oldhost` to
`<diagnostics_server_install_dir>/archive/mediator-newhost`

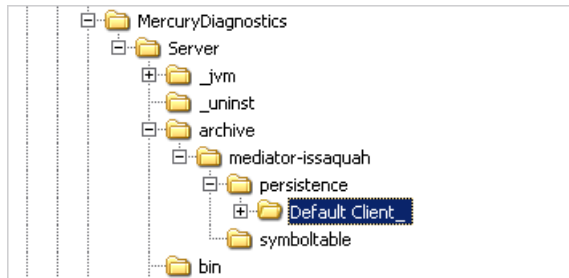
- 11 Move all of the contents of the `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence` directory to a safe back up directory called `PERCISTENCE_BACKUP`.
- 12 Create the customer name directory inside the `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence` directory.

The customer name directory was new for the 4.1 Diagnostics Server. Create a new directory that is named based on the customer name that you defined when you installed the Diagnostics Server. The new directory name must be the customer name with an underscore appended to it as follows:

`<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence/CustomerName_`.

Note: Unless you are an MMS customer, the customer name should always be `Default Client`.

When you have created the new directory, the directory structure must look as follows:

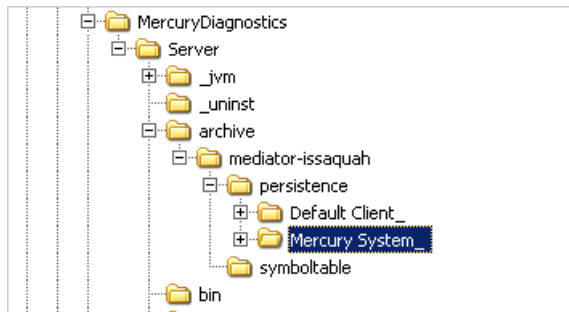


In this example the customer name was Default Client.

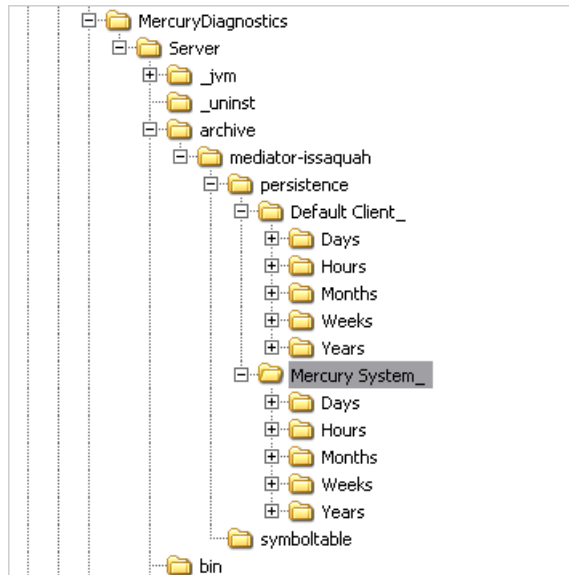
- 13** Create the Mercury System directory inside the `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence` directory.

The Mercury System directory is new for the 4.1 version of the Diagnostics Server. Create a new directory that is named Mercury System_. Be sure to include the underscore at the end of the name. The new directory structure will be as follows: `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence/Mercury System_`.

When you have created the new directory, the directory structure must look as follows:



- 14 Copy the contents of the PERSISTENCE_BACKUP that you created from `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence` in the earlier step into both of the new directories that you just created. Copy the directories in PERSISTENCE_BACKUP to `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence/CustomerName_` and to `<diagnostics_server_install_dir>/archive/mediator-<host-name>/persistence/Mercury System_`. Each of the new directories should contain the same information as shown in the following image:



- 15 Delete the `<diagnostics_server_install_dir>/storage/userdata/<customer_name>` directory for the 4.2 Diagnostics Server.

- 16** Copy the backup copy of the old `<diagnostics_server_install_dir>/storage/userdata/<customer_name>` that you created and paste it to the 4.2 `<diagnostics_server_install_dir>/storage/userdata/<customer_name>`.
-

Note: When you open the custom views that were created in Diagnostics 4.0 or 4.1 for the first time in Diagnostics 4.2, Diagnostics will upgrade the view for any changes that are necessary because of changes that were made to the functionality of Diagnostics. When Diagnostics changes your custom views is issues a message is displayed to let you know that your custom view has been modified.

- 17** Start the 4.2 Diagnostics Server. For instructions see “Starting and Stopping the Diagnostics Server” on page 46.

E

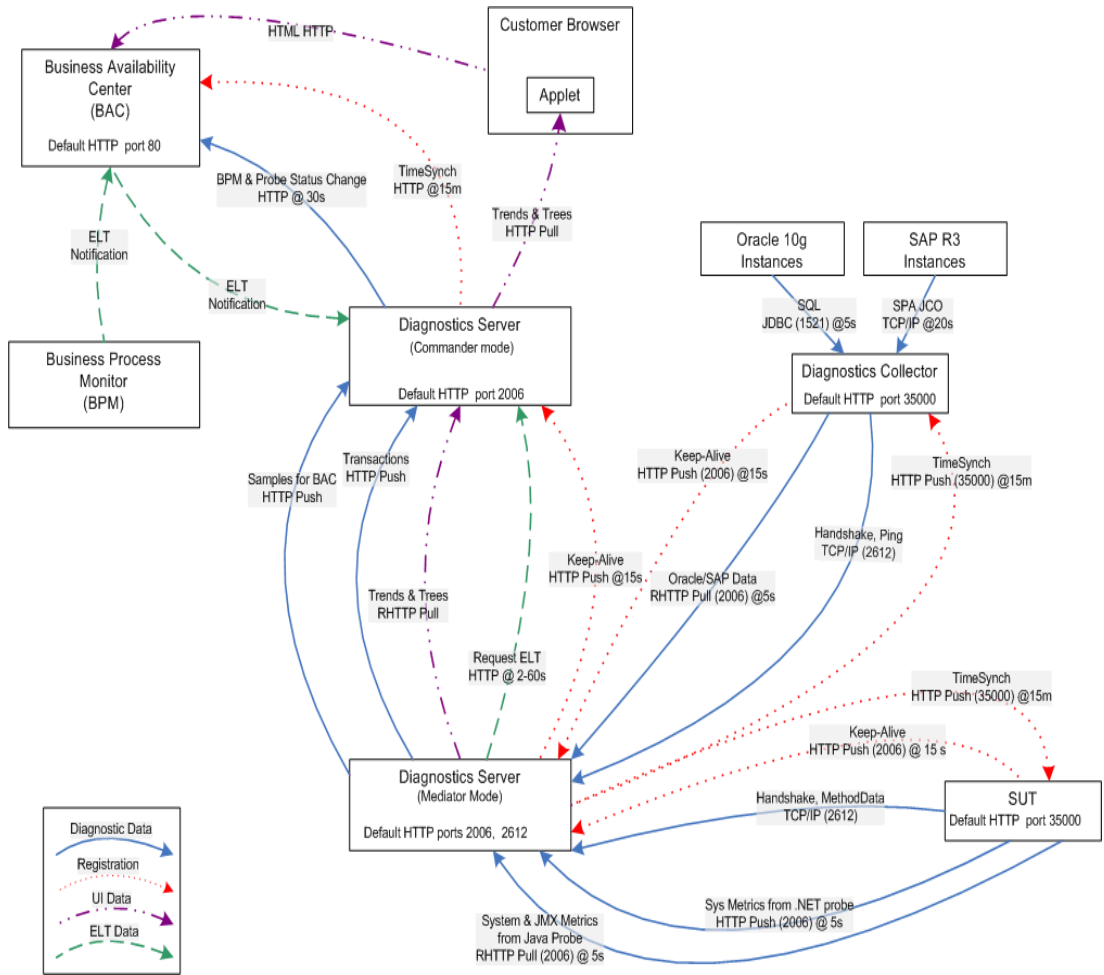
Diagnostics Component Configuration and Communication Diagrams

This appendix provides you with a diagram to assist you as you install and configure the Diagnostics components.

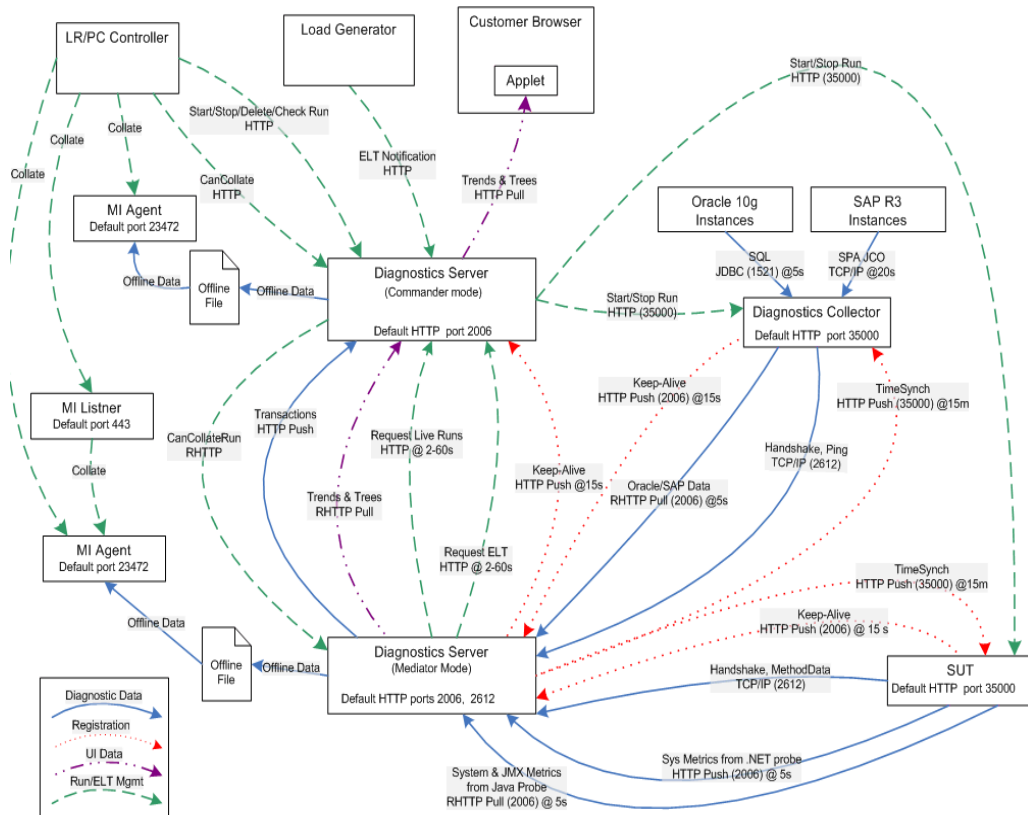
This appendix describes:	On page:
Communications with Mercury Business Availability Center	546
Communications with LoadRunner and Performance Center	547

Note: The configuration information described in this appendix is intended for experienced users with in-depth knowledge of Mercury Diagnostics. The diagrams are intended to provide a high-level view, not provide an in-depth knowledge of the working of the components.

Communications with Mercury Business Availability Center



Communications with LoadRunner and Performance Center



F

Upgrading Diagnostics and Other Mercury Products

This appendix provides you with the recommended paths for upgrading your current deployments of Diagnostics and of other Mercury products to use the components included in Mercury Diagnostics 6.5.

This appendix describes:	On page:
Overview of Product Upgrade Paths	549
General Upgrade Recommendations	550
Diagnostics Compatibility With Earlier Diagnostics Versions	551
Upgrading Diagnostics Components to Diagnostics 6.5	552
Diagnostics Compatibility with Other Mercury Products	561

Overview of Product Upgrade Paths

The following instructions help you understand how to upgrade Mercury products that you are currently using so that you can take advantage of the new features of Mercury Diagnostics 6.5.

You should consult with Mercury Customer Support if you have any questions about the information provided here.

General Upgrade Recommendations

The following recommendations are generally applicable when upgrading from earlier versions of Diagnostics or from versions of products that did not work with Diagnostics.

- ▶ Upgrade decisions should be made on a case-by-case basis—each product deployment is unique, and the recommended upgrade path for one deployment may not apply to another deployment that appears to be similar.
- ▶ Before you install the latest version of a component on the same host that was used for the earlier version of the component, make sure that the host meets the system requirements for the new component.
- ▶ If you have decided to upgrade a probe, you should also upgrade the Diagnostics Server to make sure that you can take full advantage of the latest features of the probe.
- ▶ You should contact Mercury Customer Support when you need to upgrade from an earlier version of Mercury Business Availability Center, or Performance Center.

Diagnostics Compatibility With Earlier Diagnostics Versions

The 6.5 version of the Diagnostics Server is capable of working with some earlier probe versions and the 6.5 version of the probe is capable of working with earlier versions of the Diagnostics Server.

The following table shows the compatibility of Diagnostics 6.5 components with the components of earlier product releases. Because of changes to the architecture of the product, component versions not listed in this table are not compatible with the Diagnostics 6.5 components.

D6.5 Diagnostics Component	Older Diagnostics Component	Compatible Diagnostics Version
Diagnostics Server	J2EE Probe	4.1, 4.2, 6.5
Diagnostics Server	.NET Probe	3.6, 4.0, 4.0 sp1, 4.1, 4.2, 6.5
Diagnostics Server	Collector	4.1, 4.2, 6.5
J2EE Probe	Diagnostics Server	4.2, 6.5
.NET Probe	Diagnostics Server	3.6, 4.0, 4.0 sp1, 4.1, 4.2, 6.5
Collector	Diagnostics Server	4.1, 4.2, 6.5

Upgrading Diagnostics Components to Diagnostics 6.5

The following instructions guide you in the process of upgrading an existing Diagnostics component to Diagnostics 6.5.

Upgrading a 4.1 or 4.2 Diagnostics Server

Note:

- ▶ If you update a Diagnostics Server you must upgrade all of the Diagnostics Servers in your deployment. All Diagnostics Servers in your deployment must be running the same Diagnostics version.
 - ▶ If you are an MMS customer, contact MMS Support for upgrade instructions.
-

To upgrade a 4.1 or 4.2 Diagnostics Server:

- 1** Shut down the Diagnostics Server that is to be upgraded.
- 2** Verify that the Diagnostics Server has been shut down by attempting to connect to http://<diagnostics_server_host>:2006/. (If you configured your Diagnostics server to use a port other than the default, you should use that port number 2006.) You cannot connect to the Diagnostics Server when it has been shut down.
- 3** Back up the Diagnostics data that is stored in the installation directories of the old Diagnostics Server. The directories to be backed up are:
 - ▶ `<diagnostics_server_install_dir>\archive`
 - ▶ `<diagnostics_server_install_dir>\storage`
- 4** Install the new Diagnostics Server into the same installation directory that you used for the previous version of the component.

- 5 During the component installation you will be asked if it is okay to overwrite the existing JVM. You must reply **Yes** so that the installer will overwrite the existing JVM

Note: When the installer has finished, the length of time that it takes for the Diagnostics Server to start varies depending on the amount of persisted data stored because the Diagnostics Server must first re-index this data before it can start. The more persisted data, the longer the Diagnostics Server will take to start the first time after the upgrade.

- 6 The installer creates a new `<diagnostics_server_install_dir>\etc` directory and stores a backup copy of the old directory as `<diagnostics_server_install_dir>\etc.old`.
- 7 When the installer has finished, compare the `\etc` directory and the `\etc.old` directory so that you can determine the differences between the two. (It may be helpful to use a diff/merge tool for this purpose.)

Apply any differences that were caused by the customizations that you made to the `\etc.old` directory to the `\etc` directory so that they will not be lost. You should look for the following changes:

Property File	Configuration Properties To Be Copied to the New Diagnostics Server
<code>alerting.properties</code>	SNMP and SMTP servers, mail addresses
<code>security.properties</code>	If the system is set up for SSL mode, all parameters should be updated and certificates manually copied to the new <code>/etc</code> folder
<code>server.properties -</code>	Timeout/Trimming settings
<code>thresholds.configuration</code>	Update any changes

- 8 If the system is integrated with LoadRunner or Performance Center, copy `run_id.xml` from the old `\etc` directory to the new `\etc` directory to ensure that the Run ID is properly incremented for future runs.

- 9 If you are upgrading the Diagnostics Server in Commander mode, copy the **DiagnosticsServer.lic** file from the `\etc.old` directory to the new `\etc` directory.
- 10 Restart the Diagnostics Server. The Diagnostics Server is started automatically when the installer finishes. You must stop the Diagnostics Server and then start it again in order for the customizations that you made to the properties in the `\etc` directory to take effect.

Note: The Diagnostics Servers can be started in any order whether they are in Diagnostics Server in Commander mode or a Diagnostics Server in Mediator mode. However, Diagnostics data will not be recorded until all of the Diagnostics Servers in Mediator modes are up and running.

- 11 You can verify that the upgraded Diagnostics Server is running by checking the version in the System Health graph. Browse to the URL `http://<commanding_server>:<port>/registrar/health`. Double-click the Diagnostics Server icon. The version under Configuration should be 6.5.x.x if the upgrade was successful and the Diagnostics Server was restarted.

Upgrading a 4.1 or 4.2 J2EE Probe

To upgrade a 4.1 or 4.2 J2EE Probe:

- 1 Install the Mercury Diagnostics Probe for J2EE (version 6.5) into a different directory than the current probe's installation directory. This lets you avoid disruptions to your production environment by leaving monitored applications running during the upgrade process.

Be sure to use the same probe name, probe group name and mediator server as the old probe installation in order to ensure that the persisted data for your application will match up with the metrics captured by the new probe.

Note: The new probe installation will not begin monitoring your applications until you have updated the startup scripts to start the new probe and restarted the applications as described in these instructions.

- 2 The installer creates a <probe_install_dir>\etc directory in the new installation directory.

Note: If you install the new probe into the old installation directory the installer creates a back-up of the previous versions <probe_install_dir>\etc directory and names it <probe_install_dir>\etc.old.

- 3 When the installer has finished, compare the new \etc directory and the old \etc directory so that you can determine the differences between the two. (It may be helpful to use a diff/merge tool for this purpose.)

Apply any differences that were caused by the customizations that you made to the old \etc directory to the new \etc directory so that they will not be lost. You should look for the following changes:

Property File	Configuration Properties To Be Copied to the New Diagnostics Server
auto_detect.points	See the following step for upgrade instructions.
capture.properties	Depth and latency trimming
inst.properties	
dispatcher.properties	<ul style="list-style-type: none"> ▶ minimum.sql.latency - this is a new property that should be set correlated with the Diagnostics Mediator property server.properties/sql.latency.trim ▶ sql.parsing.mode
dynamic.properties	cpu.timestamp.collection.method
metrics.config	Verify that any metric that you uncommented in the previous version is also uncommented in the new version so that you can continue to use the metrics that you are used to.
security.properties	If the system is set up for SSL mode, set all properties and copy the certificates from the old property file to the property file.

- 4 Copy custom points that you have created and points that you have modified from the **auto_detect.points** file in the old **\etc** directory to the new **\etc** directory. Be sure to check the points for RMI, LWMD, **args_by_class** when looking for points you may have modified.

Note:

- ▶ You must be extra careful during this process because the structure and layout of the capture points file has changed significantly. For more information about the capture points file and instrumentation, see Chapter 14, “Instrumenting an Application”.
- ▶ If you had points defined that used code snippets in the previous version of the points file, you must be aware that changes have been made to improve the security of the code snippets. The code snippet is no longer included in the point; but is instead included in a separate file. For more information see “Defining Points With Code Snippets” on page 247.

You may continue to include the code snippets using the less secure method by adding the property **require.code.security.file=false** to the property file **etc/code/custom_code.properties**. This allows the snippet attribute to be placed directly in the **auto_detect.points** file. This is less secured than the newer method because anyone that can access the probe's /files URL can modify the points file.

- 5 Update the applications start-up script to point to upgraded probe installation. This includes the **-javaagent** or **-Xbootclasspath**.
- 6 At an approved time, shut down the applications that were being monitored by the old probe.
- 7 Restart the applications using the revised startup script to allow the new version of the probe to begin monitoring your applications.
- 8 Clear your browser's cache and restart the browser before you attempt to access the J2EE Diagnostics Profiler user interface. Failure to do this may result in a size mismatch error message.

- 9 You can verify that the upgraded J2EE Probe is running, by checking the version in System Health. Browse to the System Health http://<commanding_server>:<port>/registrar/health, and double-click the probe icon. The version listed under Configuration should be 6.5.x.x, if the upgrade was successful and the J2EE Probe was started.

Upgrading a 4.1 or 4.2 .NET Probe

To upgrade a 4.1 or 4.2 .NET Probe:

- 1 Shut down all applications that are being monitored by the current .NET Probe.
- 2 Stop IIS.
- 3 Back-up the <probe_install_dir>\etc directory for the current probe installation.
- 4 Uninstall the .NET Probe using **Start > Settings > Control Panel > Add or Remove Programs**.
- 5 Install the new Mercury Diagnostics Probe for .NET.

The installer creates a <probe_install_dir>\etc directory in the new installation directory.

Note: If you install the new probe into the old installation directory the installer creates a back-up of the previous versions <probe_install_dir>\etc directory and names it <probe_install_dir>\etc.old.

- 6 When the installer has finished, compare the new `\etc` directory and the old `\etc` directory so that you can determine the differences between the two. (It may be helpful to use a diff/merge tool for this purpose.)

Apply any differences that were caused by the customizations that you made to the old `\etc` directory to the new `\etc` directory so that they will not be lost. You should look for the following changes:

Property File	Configuration Properties To Be Copied to the New Diagnostics Server
<code>metrics.config</code>	Check for any metrics added previous to the upgrade.
<code>probe_config.xml</code>	Add <code>monitorheap="true"</code> if you will be using the new memory diagnostics features.
<code>.points</code>	Copy all points files.

- 7 Restart IIS.
- 8 Restart the applications that were being monitored by the old probe.
- 9 Clear your browser's cache and restart the browser before you attempt to access the .NET Diagnostics Profiler user interface. Failure to do this may result in a size mismatch error message.
- 10 You can verify that the upgraded .NET Probe is running by checking the version in System Health. Browse to http://<commanding_server>:<port>/registrar/health, and double-click the probe icon. The version listed under Configuration should be 6.5.x.x, if the upgrade was successful and the .NET Probe was started.

Upgrading a 4.1 or 4.2 Collector

To upgrade a 4.1 or 4.2 .NET Collector:

- 1 Shut down the Diagnostics Collector that you want to upgrade.
- 2 Install the new Collector into the installation directory for the old version of the Collector.

- ▶ Be sure to use the same probe name, probe group name and mediator server as the old collector installation in order to ensure that the persisted data for your application will match up with the metrics captured by the new collector.
 - ▶ During the component installation you will be asked if it is okay to overwrite the existing JVM. You must reply **Yes** so that the installer will overwrite the existing JVM
- 3** The installer creates a `<probe_install_dir>\etc` directory in the installation directory. It creates a back-up of the previous version's `<probe_install_dir>\etc` directory and names it `<collector_install_dir>\etc.old`.
 - 4** When the installer has finished, compare the new `\etc` directory and the old `\etc` directory to determine the differences between the two. (It may be helpful to use a diff/merge tool for this purpose.)

Apply any differences that were caused by the customizations that you made to the old `\etc` directory to the new `\etc` directory so that they will not be lost.

Property File	Configuration Properties To Be Copied to the New Diagnostics Server
<code>oracle-config.xml</code>	If the collector is monitoring an Oracle environment
<code>r3config.xml</code>	If the collector is monitoring an R3 environment
<code>security.properties</code>	If the system is set up for SSL mode, set all properties and copy the certificates from the old property file to the property file.

- 5** Stop the Diagnostics Collector.
- 6** Start the Diagnostics Collector.
- 7** You can verify the upgraded Collector by checking the version in the System Health graph. Browse to http://<commanding_server>:<port>/registrar/health, and double-click the Collector icon. The version listed under Configuration should be 6.5.x.x, if the upgrade was successful and the Collector was started.

Upgrading the LoadRunner 8.1 FP4 Addin for 4.1 or 4.2

To upgrade the LoadRunner 8.1 FP4 Addin for 4.1 or 4.2:

- 1** Shut down the LoadRunner Controller and Agent Process.
- 2** Install the LoadRunner Addin.
- 3** Restart the LoadRunner machine.
- 4** Start the LoadRunner Controller.
- 5** For existing scenarios, select the probes for the run as follows:
 - In the LoadRunner Controller, select **Diagnostics > Configuration**.
 - In the Diagnostics Distribution dialog box, select **Configure**.

You may see the existing probes with a red status. Copies of these probes also appear with a blue status.
 - Clear any probes that appear in red, and select any of the blue probes to be included in your load test.

Upgrading the Performance Center 8.1 FP4 Patch for 4.1 or 4.2

To upgrade the Performance Center 8.1 FP4 Patch for 4.1 or 4.2:

- Verify that PC 8.1 FP3 or FP4 is installed. One of these feature packs must be installed in order to work with Diagnostics.

Diagnostics Compatibility with Other Mercury Products

The following table shows the compatibility of Diagnostics 6.5 components with versions of other Mercury products. Because of architectural changes to the product, versions not listed in this table are not compatible with the Diagnostics 6.5 components.

Mercury Product	Version Compatible with Diagnostics 6.5
LoadRunner	8.1 FP4
Performance Center	8.1 FP4
Business Availability Center	6.1 and higher

G

Troubleshooting Mercury Diagnostics

This appendix provides troubleshooting tips for problems that may occur when using Mercury Diagnostics.

This appendix describes:	On page:
Component Installation Interrupted on a Solaris Machine	564
J2EE Probe Fails to Operate Properly	564
J2EE Probe Throttling Seems Excessive	564
Error During WAS Startup with Mercury Diagnostics Profiler for J2EE	565
Missing Server-Side Transactions	566

Component Installation Interrupted on a Solaris Machine

If a component installer on a Solaris machine is interrupted before it has finished installing the component, there is no option for automatically uninstalling or reinstalling the component. You must manually clean up the partial installation of the component before you can start the installation again.

To manually clean up after an interrupted installation:

- 1** Clean the installation directory.
- 2** Delete `~/vpd.properties` and `~/vpd.patches`.
- 3** Delete the Solaris directories: `/var/sadm/pkg/IS*` and `/var/sadm/pkg/MERQ`.

J2EE Probe Fails to Operate Properly

If the J2EE Probe does not operate properly, check whether the `ClassLoader.class` file located in the folder `<probe_instal_dir>\classes\boot\java\lang\` was created during the installation process.

If the file was not created, run the JRE Instrumenter to create it. See Chapter 8, “Running the JRE Instrumenter.”

J2EE Probe Throttling Seems Excessive

By reviewing the probe logs, you can determine when throttling is being started and stopped. If throttling is being engaged more than you would like you should check to make sure that the probe thread configuration is synchronized with the number of threads that the application server has been configured to allow. If the application server has more threads than the probe can handle, throttling is engaged sooner and may not be disengaged.

For information on tuning the probe to reduce throttling see “Controlling Probe Throttling” on page 404.

Error During WAS Startup with Mercury Diagnostics Profiler for J2EE

Symptoms:

Class Loader errors occur when starting WAS with the Mercury Diagnostics Profiler for J2EE.

Reason:

Additional classes need to be excluded from the instrumentation.

Solution:

- 1 Open the property file, <probe_install_dir>\etc\inst.properties
- 2 Update the **classes.to.exclude** property to exclude **!com\.ibm\..*** by appending the class to the end of the existing values.

```
classes.to.exclude=!aik\.security\.*;!c8e\.*;!org\.jboss\.net\.protocol\.file\.Handler,!org\.jboss\.net\.protocol\.file\.URLConnection,!.*ByCGLIB.*;!com\.ibm\..*
```

Missing Server-Side Transactions

Symptoms:

The server requests for each Probe are displayed in Diagnostics but the BPM transactions that are associated with the server requests are not displayed.

There are two symptoms to look for in the server.log file:

“not dropping at least one transaction that timed out” – this indicates that a transaction has not received any data for a period of time (10m by default) and has not received the ELT. This warning is issued infrequently, and only when the transaction times out...after this warning you should see the transaction data in the UI. For more information on ELT see “Reducing Diagnostics Server Memory Usage” on page 323.

“Late data received for time period that was already persisted. Adjusting data by...” – this indicates that the server received an ELT unreasonably late, but before the transaction timed out. The data will be reported, but not at the same time that BAC or MMS reported it.

Reason:

If you do not see either of the log messages listed above and there is no transaction data the most likely cause is the BPM is not running the scripts.

Solution:

- 1 Verify that BPM is running in Business Availability Center or MMS and that the BPM is running.
- 2 Verify the state of the profile in the BPM Console.

H

Using UNIX Commands

When running an installation on a UNIX platform, you can usually follow the instructions that appear on the screen. The on-screen instructions can be confusing at times.

If something is unclear, use the following guidelines:

- ▶ To select an option from a list of options, type the number corresponding to the option and press `ENTER`. Then type `0` and press `ENTER` again to confirm your choice.
- ▶ When selecting multiple options, for each selection type the corresponding number and press `ENTER`. Only when you have finished selecting all the options that you want to select, type `0` and press `ENTER` again to confirm your choices.
- ▶ If you have selected an option and want to clear it, retype the corresponding number, or type the number of another option, and press `ENTER`. Then type `0` and press `ENTER` again to confirm your choice.
- ▶ When entering information at a prompt:
 - ▶ To accept a default value that is displayed at the prompt, press `ENTER`.
 - ▶ Type the information, and press `ENTER` to continue.
- ▶ To continue with the next step of an installation, type `1` to select **Next**, and press `ENTER`.
- ▶ To go back to previous prompts to make changes, type `2` to select **Previous**, and press `ENTER`.
- ▶ To cancel an installation, type `3` to select **Cancel**, and press `ENTER`.
- ▶ To redisplay a prompt, type `4` to select **Redisplay**, and press `ENTER`.

Using Regular Expressions

When you specify the instrumentation definitions for each probe instance in the capture points file, you can use regular expressions for most of the arguments in a point.

A regular expression is a string that specifies a complex search phrase. By using special characters, such as a period (`.`), asterisk (`*`), caret (`^`), and brackets (`[]`), you can define the conditions of a search.

Note: Regular expressions in Diagnostics must be prefaced with an exclamation point.

By default, Diagnostics treats all characters in a regular expression literally, except for the period (`.`), hyphen (`-`), asterisk (`*`), caret (`^`), brackets (`[]`), parentheses (`()`), dollar sign (`$`), vertical line (`|`), plus sign (`+`), question mark (`?`), and backslash (`\`). When one of these special characters is preceded by a backslash (`\`), Diagnostics treats it as a literal character.

This appendix describes:	On page:
Common Regular Expression Operators	570
Combining Regular Expression Operators	576

Common Regular Expression Operators

This section describes some of the more common operators that can be used to create regular expressions.

Note: For a complete list and explanation of supported regular expression characters, refer to the Regular Expressions section in the Microsoft VBScript documentation.

Operator	Used for
(\)	Rendering Special Characters Literal Creating Special Characters out of Literal Characters
(.)	Matching Any Single Character
([xy])	Matching Any Single Character in a List
([^xy])	Matching Any Single Character Not in a List
([x-y])	Matching Any Single Character within a Range
(*)	Matching Zero or More Specific Characters
(+)	Matching One or More Specific Characters
(?)	Matching Zero or One Specific Character
(())	Grouping Regular Expressions
()	Matching One of Several Regular Expressions
(^)	Matching the Beginning of a Line
(\$)	Matching the End of a Line
(\w)	Matching Any AlphaNumeric Character Including the Underscore
(\W)	Matching Any Non-AlphaNumeric Character

Using the Backslash Character

A backslash (\) can serve two purposes. It can be used in conjunction with a special character to indicate that the next character be treated as a literal character. For example, \. would be treated as period (.) instead of a wildcard (see “Matching Any Single Character” on page 572).

Alternatively, if the backslash (\) is used in conjunction with some characters that would otherwise be treated as literal characters, such as the letters n, t, w, or d, the combination indicates a special character. For example, \n stands for the newline character.

For example:

- ▶ w matches the character w
- ▶ \w is a special character that matches any word character including underscore
- ▶ \\ matches the literal character \
- ▶ \(matches the literal character (

For example, if you were looking for a file called:

filename.ext

the period would be mistaken as an indication of a regular expression. To indicate that the period is not part of a regular expression, you would enter it as follows:

filename\.ext

Note: If a backslash character is used before a character that has no special meaning, the backslash is ignored. For example, \z matches z.

Matching Any Single Character

A period (.) instructs Diagnostics to search for any single character (except for \n). For example:

welcome.

matches **welcomes**, **welcomed**, or **welcome** followed by a space or any other single character. A series of periods indicates the same number of unspecified characters.

To match any single character including \n, enter:

(.|\\n)

For more information on the () regular expression characters, see “Grouping Regular Expressions” on page 574. For more information on the | regular expression character, see “Matching One of Several Regular Expressions” on page 575.

Matching Any Single Character in a List

Square brackets instruct Diagnostics to search for any single character within a list of characters. For example, to search for the date 1967, 1968, or 1969, enter:

196[789]

Matching Any Single Character Not in a List

When a caret (^) is the first character inside square brackets, it instructs Diagnostics to match any character in the list except for the ones specified in the string. For example:

```
[^ab]
```

matches any character except **a** or **b**.

Note: The caret has this special meaning only when it is the first character displayed within the brackets.

Matching Any Single Character within a Range

To match a single character within a range, you can use square brackets ([]) with the hyphen (-) character. For example, to match any year in the 1960s, enter:

```
196[0-9]
```

A hyphen does not signify a range if it is displayed as the first or last character within brackets, or after a caret (^).

For example, [-a-z] matches a hyphen or any lowercase letter.

Note: Within brackets, the characters ".", "*", "[", and "\" are literal. For example, [.*] matches . or *. If the right bracket is the first character in the range, it is also literal.

Matching Zero or More Specific Characters

An asterisk (*) instructs Diagnostics to match zero or more occurrences of the preceding character. For example:

```
ca*r
```

matches **car**, **caaaaaar**, and **cr**.

Matching One or More Specific Characters

A plus sign (+) instructs Diagnostics to match one or more occurrences of the preceding character. For example:

```
ca+r
```

matches **car** and **caaaaaar**, but not **cr**.

Matching Zero or One Specific Character

A question mark (?) instructs Diagnostics to match zero or one occurrences of the preceding character. For example:

```
ca?r
```

matches **car** and **cr**, but nothing else.

Grouping Regular Expressions

Parentheses (()) instruct Diagnostics to treat the contained sequence as a unit, just as in mathematics and programming languages.

Using groups is especially useful for delimiting the argument(s) to an alternation operator (|) or a repetition operator (*, +, ?, {}).

Matching One of Several Regular Expressions

A vertical line (|) instructs Diagnostics to match one of a choice of expressions. For example:

`foo|bar`

causes Diagnostics to match either **foo** or **bar**.

`fo(o|b)ar`

causes Diagnostics to match either **fooar** or **fobar**.

Matching the Beginning of a Line

A caret (^) instructs Diagnostics to match the expression only at the start of a line, or after a newline character.

For example:

`book`

matches **book** within the lines **book**, **my book**, and **book list**, while

`^book`

matches **book** only in the lines **book** and **book list**.

Matching the End of a Line

A dollar sign (\$) instructs Diagnostics to match the expression only at the end of a line, or before a newline character. For example:

`book`

matches **book** within the lines **my book**, and **book list**, while a string that is followed by (\$), matches only lines ending in that string. For example:

`book$`

matches **book** only in the line **my book**.

Matching Any AlphaNumeric Character Including the Underscore

`\w` instructs Diagnostics to match any alphanumeric character and the underscore (A-Z, a-z, 0-9, `_`).

For example:

`\w*` causes Diagnostics to match zero or more occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (`_`). It matches **Ab**, **r9Cj**, or **12_uYLgeu_435**.

For example:

`\w{3}` causes Diagnostics to match 3 occurrences of the alphanumeric characters—A-Z, a-z, 0-9, and the underscore (`_`). It matches **Ab4**, **r9_**, or **z_M**.

Matching Any Non-AlphaNumeric Character

`\W` instructs Diagnostics to match any character other than alphanumeric characters and underscores.

For example:

`\W` matches **&**, *****, **^**, **%**, **\$**, and **#**.

Combining Regular Expression Operators

You can combine regular expression operators in a single expression to achieve the exact search criteria you need.

For example, you can combine the `.` and `*` characters to find zero or more occurrences of any character (except `\n`).

For example,

`start.*`

matches **start**, **started**, **starting**, **starter**, and so forth.

You can use a combination of brackets and an asterisk to limit the search to a combination of non-numeric characters. For example:

`[a-zA-Z]*`

To match any number between 0 and 1200, you need to match numbers with 1 digit, 2 digits, 3 digits, or 4 digits between 1000-1200.

The regular expression below matches any number between 0 and 1200.

`(([0-9]?[0-9]?[0-9]|1[01][0-9][0-9]|1200)`



Multi-Lingual User Interface Support

In Diagnostics 6.5 and later, the Diagnostics user interface (UI) can be viewed in multiple languages in your Web browser. This applies in a Windows environment, when Diagnostics is integrated with Mercury Business Availability Center or running in standalone mode (no integration).

If Diagnostics is integrated with LoadRunner or Performance Center, the display language of the UI is determined by the client locale setting (defined in the Regional Settings of your operating system)

This appendix explains how to view the Diagnostics user interface in a specific language. The Diagnostics UI can be viewed in the following languages in your Web browser:

Language	Language preference in Web browser
English	English
Simplified Chinese	Chinese (China) [zh-cn], Chinese (Singapore) [zh-sg]
Korean	Korean [ko]
Japanese	Japanese [ja]

You use the language preference option in your browser to select how you view Diagnostics. The language preference chosen affects only the user's local machine and not the Diagnostics Server or any other user accessing the same Diagnostics Server.

To view the Diagnostics UI in a specific language:

- 1** Install the appropriate language's fonts on your local machine if they have not yet been installed. If you choose a language in your Web browser whose fonts have not been installed, the Diagnostics user interface uses the default language of your local machine.

Assume, for example, that the default language on your local machine is English and the Web browser is configured to use Japanese. If Japanese fonts are not installed on the local machine, the Diagnostics user interface is displayed in English.

- 2** If you are using Internet Explorer, configure the Web browser on your local machine to select the language in which you want to view the Diagnostics user interface. For details, see <http://support.microsoft.com/kb/306872/en-us>.

Continue with step 4.

- 3** If you are using FireFox, configure the Web browser on your local machine as follows:
 - a** Select **Tools > Options > Advanced**. Click **Edit Languages**. The Language dialog box opens.
 - b** Highlight the language in which you want to view Diagnostics.
If the language you want is not listed in the dialog box, expand the **Select language to add** list, select the language, and click **Add**.
 - c** Click **Move Up** to move the selected language to the first row.
 - d** Click **OK** to save the settings. Click **OK** to close the Language dialog box.
- 4** Close your existing browser and open Diagnostics in a new browser. The Diagnostics user interface is displayed in the selected language.

Index

A

- advanced configuration
 - Diagnostics Server 486
- advanced Mediator assignment 327
- AM product mode, Probe settings 391, 392
- application server
 - configuring *See* application server configuration
- application server configuration 167
 - generic 201
 - JBoss 196
 - multiple instances 393
 - Oracle 9i 193
 - SAP NetWeaver 199
 - WebLogic 6.1 184
 - WebLogic 7 187
 - WebLogic 8 188
 - WebLogic 8.1 188
 - WebSphere 169
 - WebSphere 4.0 170
 - WebSphere 5.x ,6.x 174
- application server, advanced configuration 387
- application servers, supported 6
- Automatic Method Trimming
 - controlling 402
 - depth 403
 - latency 402

B

- backslash (\) 571
- Business Availability Center, *See* Mercury Business Availability Center

C

- capture points files 238, 273, 289
 - mandatory point entries 242
 - optional point entries 244
- ClassLoader class, recreating 564
- communication diagrams 545
- component configuration 545
- configuring application servers
 - WebLogic 184
- configuring application servers, *See* application server configuration

D

- data management, *See* Diagnostics data management
- default Mediator assignment 327
- deployment configuration 18
- deployment, recommended configuration 18
- depth trimming 403
- Diagnostics components 4
 - host requirements 7
 - synchronizing time between 310
- Diagnostics data flow 18
- Diagnostics data management
 - backing up data 536, 539
 - custom screen data 528
 - data sizes & data retention 533
 - performance considerations 535
 - performance history data 530, 531
- Diagnostics Distribution dialog box 507
- Diagnostics layers
 - .NET layers 292
 - J2EE layers 290
 - Portal layers 293

Diagnostics Probe for .NET, *See* Probe, .NET
Diagnostics Probe for J2EE, *See* J2EE Probe
Diagnostics Server 12

- adjusting heap size 314
- administration 481
- changing default port 318
- configuration pages 328, 483
- configuring 49
- configuring for large installation 314
- configuring for multi-homed environments 319
- configuring LoadRunner offline file 328
- configuring time synchronization 312
- configuring, advanced 309, 481
- HTTPS communication 419
- installing (Unix) 35
- installing (Windows) 24, 53
- jetty.xml file, modifying 320
- jetty.xml file, sample 322
- LoadRunner / Performance Center
 - assignments 327
- modifying properties 487
- overriding default host name 317
- reducing memory usage 323
- setting event host name 319
- starting and stopping 46
- synchronizing time between
 - Diagnostics components 310
- System Health Monitor 514, 515
- system requirements 17
- troubleshooting 52
- uninstalling 50
- verifying installation 48
- viewing advanced configuration options 486

Diagnostics setup menu 24, 89
disable Probes 450
documentation updates xviii

E

event host

- setting name for 319

F

filter

- System Health Monitor 523

H

heap size 314
host requirements, Diagnostics components 7
HTTP proxy communication 431

- .NET Probe 433
- Diagnostics Server in Mediator Mode 432
- J2EE Probe 433

HTTPS communication 418

- enabling for Diagnostics Server in Commanding mode 419
- enabling for Mercury Business Availability Center server 428

I

installation

- Diagnostics Server (Unix) 35
- Diagnostics Server (Windows) 24, 53
- interruption during 564
- J2EE Probe 103
- planning 11
- Probe for .NET 88
- Probe for .NET, verifying 100
- recommended order 18
- requirements, *See* installation requirements

installation requirements

- .NET Diagnostics Profiler 10
- .NET Probe 10, 11
- Diagnostics Server 7
- J2EE Diagnostics Profiler 9
- J2EE Probe 8

J

J2EE Diagnostics Profiler

- disabling 397
- Probe settings, PRO product mode 390, 392

- WAS startup error 565, 566
- J2EE Probe
 - administration page 409
 - configuring and installing, about 104, 144
 - controlling log messages 399
 - excessive throttling 564
 - failed operation 564
 - installing 103
 - installing as a Profiler 109
 - installing on z/OS 137
 - installing using generic installer 139
 - installing with a Mercury Diagnostics Server 113
 - silent installation 141
 - system requirements 8
 - throttling, controlling 404
 - uninstalling 154
 - upgrading 151
- J2EE Probe advanced configuration 387
 - adding a Mercury product 389
 - AM product mode–Application Management Probe settings 391, 392
 - Automatic Method Trimming 402
 - log messages 399
 - PRO product mode–Diagnostics Profiler for J2EE Probe settings 390, 392
 - proxy server 407
 - removing a Mercury product 392
 - reverse HTTP for Probe in MMS 407
 - specifying properties as java system properties 407
 - throttling 404
- J2EE/.NET Diagnostics Configuration dialog box 448, 450, 507, 508
- java executable (Windows)
 - changing 137
- java system properties in Probe 407
- JBoss, application server configuration 196
- JDK/JRE executable 109, 117
- jetty.xml file
 - modifying 320
 - sample 322
- JMX metrics
 - accessing 471
 - collecting 474
 - understanding patterns 476
- JRE Instrumenter
 - about 155
 - running 157
- L**
- latency trimming 402
- light-weight memory Diagnostics (LWMD) 352
- Linux custom metrics 466
- LoadRunner
 - AddIn, installing 223
 - Diagnostics Server details, specifying 228
 - Diagnostics, configuring scenarios to use 230
 - Diagnostics, setting up 227
- LoadRunner Offline File
 - advanced Diagnostics Server configuration 328
 - estimating size of 328
 - reducing size of 329
- log messages 399
- LWMD *See* light-weight memory Diagnostics (LWMD)
- M**
- Mediator assignment
 - advanced 327
 - default 327
- memory usage, reducing 323
- Mercury Business Availability Center
 - assigning permissions for Diagnostics users 219
 - changing the Diagnostics server details 213
 - Diagnostics configuration 213
 - Diagnostics monitoring, enabling in Business Process Monitor (BPM) related views 217

Index

- Diagnostics server details, specifying 209
- Diagnostics view 215
 - setting up to use Diagnostics 208
- Mercury Business Availability Center server, HTTPS communication 428
- Mercury Customer Support Web site xviii
- Mercury Diagnostics Probe for .NET, *See* Probe, .NET
- Mercury Diagnostics Probe for J2EE, *See* J2EE Probe
- Mercury Home Page xviii
- Mercury Managed Services (MMS)
 - configuring reverse HTTP for Probe 407
 - enabling reverse HTTP for Probe 408
- Mercury product, adding to Probe 389
- metrics collector
 - modifying default port 461
 - system metrics 461
- metrics, Diagnostics
 - System Health Monitor 517
- MMS, *See* Mercury Managed Services (MMS)

N

- NET (.NET) Probe, *See* Probe, .NET
- NET Diagnostics Profiler
 - enabling 346

O

- Oracle 9i, application server configuration 193
- order of installation, recommended 18

P

- Performance Center
 - Diagnostics server details, specifying 232
 - load tests, configuring to use Diagnostics 234
 - setting up to use Diagnostics 231
- permissions, *See* user permissions
- preinstallation considerations 17
- PRO product mode, Probe settings 390, 392

- Probe
 - disable 450
 - System Health Monitor 515
- Probe instrumentation
 - .NET layers 292
 - capture points files 238, 273, 289
 - J2EE layers 290
 - portal layers 293
 - understanding Diagnostics layers 272, 290
- Probe, .NET 87
 - configuring 102
 - elements and attributes 357, 358
 - enabling 334
 - installation, verifying 100
 - installing 88
 - installing as a Profiler 92
 - installing with a Mercury Diagnostics Server 94
 - system requirements 10, 11
 - uninstalling 102
 - version, determining 102
- Probe, .NET advanced configuration
 - ASP.NET applications 336
 - classes/methods 341
 - customizing instrumentation for ASP.NET applications 341
 - depth trimming 351
 - disabling logging 354
 - elements and attributes 358
 - latency trimming and throttling 346
 - light-weight memory Diagnostics (LWMD) 352
 - overriding default Probe host name 354
- Probe, J2EE, *See* J2EE Probe
- product security 490
- product upgrade paths 549
 - general recommendations 550
- proxy server
 - configuring for Probe 407

R

- refresh rate
 - System Health Monitor 524

- regular expressions
 - backslash (\) 571
- requirements
 - Diagnostics Server 17
- reverse HTTP
 - configuring for Probe in MMS 407
 - disabling for Probe in MMS 408
 - enabling for Probe in MMS 408
- S**
- SAP NetWeaver, application server
 - configuration 199
- security permissions 491
- Server, Diagnostics *See* Diagnostics Server
- setup menu 24, 89
- snapshots
 - System Health Monitor 525
- Solaris custom metrics 465
- solution templates 295
 - understanding data 296
 - viewing information 296
- System Health Monitor 505
 - accessing 506
 - accessing from browser 506
 - Commander properties 515
 - component configuration
 - information 518
 - component icons 511
 - component information, viewing
 - detailed 516
 - component log information 519
 - component map 510
 - component metric information 517
 - component status information 519
 - customizing display 521
 - Diagnostics Server in Commander
 - mode properties 514
 - Diagnostics Server in Mediator mode
 - properties 515
 - displaying legend 521
 - displaying load 522
 - filter by customer 523
 - icons 510
 - legend 510
 - miscellaneous information 519
 - Probe properties 515
 - refreshing data manually 524
 - refreshing display 524
 - snapshots, exporting 525
 - snapshots, importing 526
 - system log information 520
 - tooltips, component status and host
 - configuration 514
 - viewing log history 520
 - viewing troubleshooting tips 520
- system metrics 453, 459
 - capture 453
 - customizing 454
 - customizing in Windows 462
 - customizing on Linux host 466
 - customizing on Solaris host 465
 - default 460
 - metrics collector 454, 461
 - metrics collector entries 454
 - modifying captured metrics 457
 - stopping capture 457
- system requirements
 - Diagnostics Probe for .NET host 10
 - Diagnostics Probe for J2EE host 8
 - Diagnostics Server host 7
- T**
- throttling 404
 - J2EE Probe 404
 - tuning 406
 - turning off 405
- time synchronization 310
- troubleshooting Diagnostics 563
- troubleshooting Diagnostics Servers 52
- U**
- uninstalling Diagnostics Server 50
- UNIX commands 567
- updates, documentation xviii
- upgrade paths 549
 - general recommendations 550
- upgrading earlier versions of Diagnostics 20

Index

- user permissions 490
 - accessing Diagnostics, default users 492
 - assigning for Diagnostics users in Mercury Business Availability Center 219
 - managing user details 493, 496

W

- WebLogic 6.1, application server configuration 184
- WebLogic 7, application server configuration 187
- WebLogic 8, application server configuration 188
- WebLogic 8.1, application server configuration 188
- WebLogic, application server configuration 184
- WebSphere 4.0, application server configuration 170
- WebSphere 5.x and 6.x, application server configuration 174
- WebSphere, application server configuration 169
- Windows custom metrics 462

Z

- z/OS
 - installing the Probe 137