

A person in a dark suit is walking, carrying a white briefcase. A red cable is attached to the briefcase and trails behind it on the floor. The background is a plain, light-colored wall.

# MERCURY DIAGNOSTICS FOR J2EE & .NET™ 3.6

Supporting LoadRunner and Performance Center  
Version 8.1

Installation and User's Guide

**MERCURY™**



# **Mercury Diagnostics 3.6 for J2EE & .NET**

Supporting LoadRunner and Performance Center 8.1  
Installation and User's Guide



## Mercury Diagnostics 3.6 for J2EE & .NET Supporting LoadRunner and Performance Center 8.1 Installation and User's Guide

This manual, 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.

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 Interactive Corporation  
379 North Whisman Road  
Mountain View, CA 94043  
Tel: (650) 603-5200  
Toll Free: (800) TEST-911  
Customer Support: (877) TEST-HLP  
Fax: (650) 603-5300

© 2004 - 2005 Mercury Interactive Corporation, All rights reserved

If you have any comments or suggestions regarding this document, please send them via e-mail to [documentation@mercury.com](mailto:documentation@mercury.com).

---

# Table of Contents

## **PART I: INTRODUCTION**

<b>Chapter 1: Mercury Diagnostics for J2EE &amp; .NET Product Overview ...</b>	<b>3</b>
Introducing Mercury Diagnostics for J2EE & .NET .....	4
Diagnostics for J2EE & .NET Components.....	5
Diagnostics Data Flow .....	7
Diagnostics Performance Metric Processing .....	8
Diagnostics Layers .....	9
Diagnostics Performance Metric Reporting .....	11

## **PART II: INSTALLATION AND CONFIGURATION OF DIAGNOSTICS FOR J2EE & .NET COMPONENTS**

<b>Chapter 2: Preparing to Install Mercury Diagnostics for J2EE &amp; .NET .....</b>	<b>15</b>
Recommended Deployment Configuration .....	16
Host System Requirements for the Diagnostics Components .....	17
Upgrading from Earlier Versions of Diagnostics.....	19
Other Considerations Before Installing the Diagnostics Components.....	19
Recommended Order of Installation.....	21
Planning the Installation .....	22
<b>Chapter 3: Installing the Mercury Diagnostics Commander.....</b>	<b>31</b>
Installing the Commander on a Windows Machine .....	32
Installing the Commander on a UNIX Machine .....	38
Starting and Stopping the Commander .....	42
Verifying the Commander Installation.....	45
Determining the Version of the Commander that is Installed .....	46
Uninstalling the Commander .....	46

<b>Chapter 4: Installing LoadRunner 8.1 and the LoadRunner Diagnostics Add-in.....</b>	<b>49</b>
Understanding the LoadRunner Diagnostics Add-in.....	50
Installing Mercury LoadRunner 8.1 .....	50
Installing the Mercury LoadRunner Diagnostics Add-in for J2EE & .NET .....	51
Configuring LoadRunner for Diagnostics for J2EE & .NET .....	55
<b>Chapter 5: Installing the Mercury Diagnostics Mediator .....</b>	<b>57</b>
Installing the Mediator on a Windows Machine .....	58
Installing the Mediator on a UNIX Machine .....	67
Starting and Stopping Mediators .....	73
Verifying the Mediator Installation .....	75
Troubleshooting Mediator Issues .....	76
Configuring the Mediator .....	76
Determining the Version of the Installed Mediator .....	76
Uninstalling the Mediator.....	76
Upgrading to a Newer Version of the Mediator.....	77
<b>Chapter 6: Installing the Mercury Diagnostics Probe for .NET .....</b>	<b>79</b>
About the Mercury Diagnostics Probe for .NET .....	79
Installing the .NET Probe .....	80
Verifying the .NET Probe Installation.....	92
Configuring the .NET Probe.....	93

### **PART III: INSTALLATION AND CONFIGURATION OF DIAGNOSTICS PROBE FOR J2EE**

<b>Chapter 7: Introducing J2EE Probe Installation and Application Server Configuration .....</b>	<b>97</b>
About Installing and Configuring the J2EE Probe .....	97
About Configuring the Application Server .....	98
<b>Chapter 8: Installing the Mercury Diagnostics Probe for J2EE.....</b>	<b>99</b>
Installing the J2EE Probe on a Windows Machine .....	100
Installing the J2EE Probe on a UNIX Machine .....	118
Installing the J2EE Probe on a z/OS Mainframe .....	130
Installing the J2EE Probe Using the Generic Installer .....	133
Verifying the J2EE Probe Installation.....	134
Using the J2EE Probe with Deep Diagnostics .....	136
Overriding the Default Probe Host Machine Name.....	136
Determining the Version of the J2EE Probe That is Installed.....	136
Upgrading to a Newer Version of the J2EE Probe .....	137
Uninstalling the J2EE Probe .....	140

<b>Chapter 9: Running the JRE Instrumenter .....</b>	<b>141</b>
About the JRE Instrumenter .....	141
Running the JRE Instrumenter.....	142
<b>Chapter 10: Configuring the Application Server and Probe Using the Configuration Utility .....</b>	<b>147</b>
About the Configuration Utility .....	147
Starting the Mercury Configuration Utility .....	148
Defining and Configuring Application Server Instances.....	149
Configuring Application Server Instance Startup Scripts .....	153
Removing the Probe Configuration from the Application Server Instance Startup Script.....	155
Deleting an Application Server Definition.....	157
Reconfiguring a Probe .....	158
<b>Chapter 11: Configuring the Application Servers Manually .....</b>	<b>161</b>
About Configuring the Application Server .....	161
Configuring WebSphere Application Servers.....	163
Configuring WebLogic Application Servers .....	177
Configuring the Oracle9i Application Server .....	185
Configuring the JBoss Application Server .....	189
Configuring the SAP NetWeaver Application Server .....	192
Configuring a Generic Application Server .....	194
Configuring the Probes for Multiple Application Server Instances ..	196

## **PART IV: USING MERCURY DIAGNOSTICS FOR J2EE & .NET**

<b>Chapter 12: Setting Up Diagnostics for J2EE &amp; .NET on LoadRunner 8.1 .....</b>	<b>207</b>
About Setting Up Diagnostics for J2EE & .NET.....	208
Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1 .....	209
Configuring LoadRunner Scenarios to use Diagnostics for J2EE & .NET .....	211
<b>Chapter 13: Setting Up Diagnostics for J2EE &amp; .NET on Performance Center 8.1 .....</b>	<b>217</b>
About Setting Up Diagnostics for J2EE & .NET.....	218
Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1 .....	218
Configuring Performance Center Load Tests to use Diagnostics for J2EE & .NET.....	220

<b>Chapter 14: Introducing Diagnostics for J2EE &amp; .NET Screens .....</b>	<b>225</b>
Viewing Diagnostics Data from the Controller .....	225
Introducing the Diagnostics Screens.....	227
Drilling Down Into the Diagnostics Metrics.....	228
Using the Diagnostics Navigation and Display Controls .....	230
<b>Chapter 15: Using the Diagnostics for J2EE &amp; .NET Screens .....</b>	<b>239</b>
Analyzing Performance Using the Diagnostics Overview Screen .....	240
Analyzing Performance with the Transaction Trends Screen.....	244
Analyzing Performance with the Server Request Screens .....	249
Analyzing Performance with the Layer Screens.....	255
Analyzing Performance with the Virtual Machines Screen .....	261
Analyzing Performance with the Aggregate Profile Screen.....	267
<b>Chapter 16: J2EE &amp; .NET Diagnostics Analysis Graphs .....</b>	<b>273</b>
About J2EE & .NET Diagnostics Graphs.....	273
Viewing J2EE & .NET Diagnostics in the Summary Report .....	274
Viewing J2EE & .NET Diagnostics Data .....	276
J2EE & .NET Diagnostics Graphs.....	292
J2EE & .NET Server Diagnostics Graphs.....	302
<b>Chapter 17: Web Service Support .....</b>	<b>313</b>
About Web Service Support.....	313
Writing VuGen Scripts for Web Service Support .....	314
<b>Chapter 18: Troubleshooting Diagnostics for J2EE &amp; .NET .....</b>	<b>315</b>
Component Installation Interrupted on a Solaris Machine .....	316
Version Mismatch Between Diagnostics Commander and LoadRunner Add-In.....	317
J2EE Probe Fails to Operate Properly.....	318

**PART V: APPENDIXES**

<b>Appendix A: Using the System Health Monitor .....</b>	<b>321</b>
Introducing the System Health Monitor .....	322
Accessing the System Health Monitor .....	322
Using the System Health Monitor .....	326
Drilling Down Into The System Health Monitor Map .....	329
Customizing the System Health Monitor Display .....	335
Creating and Using System Health Monitor Snapshots .....	337
Troubleshooting Using the System Health Monitor.....	339
<b>Appendix B: Advanced Diagnostics Commander Configuration .....</b>	<b>341</b>
Adjusting the Heap Size for the Commander's VM.....	341
Configuring the Commander for Multi-Homed Environments.....	343



<b>Appendix C: Advanced Diagnostics Mediator Configuration .....</b>	<b>347</b>
Configuring the Mediator for Large Installations.....	348
Tuning the Mediator Garbage Collection .....	353
Tuning the Mediator for a Smaller Installation .....	355
Overriding the Default Mediator Host Machine Name .....	356
Configuring the Mediator for Multi-Homed Environments .....	357
Reducing Mediator Memory Usage.....	360
LoadRunner / Performance Center Diagnostics	
Mediator Assignments.....	361
Using the Mediator Configuration Web Pages .....	362
Configuring the Mediator for the LoadRunner Offline file.....	366
<b>Appendix D: Advanced J2EE Probe and Application Server</b>	
<b>Configuration .....</b>	<b>369</b>
Configuring the J2EE Probe for Various Mercury Products .....	370
Setting the J2EE Probe Product Mode Properties .....	371
Configuring the J2EE Probe and Application Server for	
Deep Diagnostics .....	375
Unconfiguring the Probe for a Product.....	376
Configuring the Application Server for Allocation Capture.....	377
Specifying Layers to Instrument .....	378
Controlling Automatic Method Trimming.....	380
Controlling J2EE Probe Throttling.....	382
Configuring a Probe With a Proxy Server .....	383
Specifying Probe Properties as Java System Properties.....	384
<b>Appendix E: Advanced .NET Probe Configuration .....</b>	<b>385</b>
Automatically Discovering ASP.NET Applications.....	385
Customizing the Instrumentation for ASP.NET Applications .....	386
Restarting IIS.....	389
Elements Used in the Probe_config.xml File .....	390
Disabling Logging.....	397
Overriding the Default Probe Host Machine Name.....	397
<b>Appendix F: Configuring Diagnostics Components to</b>	
<b>Work with a Firewall.....</b>	<b>399</b>
Overview of Configuring Diagnostics for a Firewall.....	400
Collating Mediator Offline Files over a Firewall .....	401
Installing and Configuring the Mercury MI Listener .....	402
Configuring the Mediator to Work with a Firewall .....	402
Configuring LoadRunner and Performance Center to Work	
with Diagnostics Firewalls.....	409

<b>Appendix G: Configuring Diagnostics Components for HTTPS.....</b>	<b>413</b>
Configuring the Commander to Receive HTTPS .....	414
Configuring a Mediator to Communicate via HTTPS with the Commander .....	420
Configuring a J2EE Probe to Communicate via HTTPS with the Commander .....	422
Configuring a .NET Probe to Communicate via HTTPS with the Commander .....	424
Configuring the Mediator to Receive HTTPS.....	426
Configuring the Commander to Communicate via HTTPS with the Mediator.....	429
<b>Appendix H: Configuring Diagnostics Components for HTTP/HTTPS Proxy.....</b>	<b>431</b>
Configuring the Mediator to Communicate Through an HTTP/HTTPS Proxy .....	432
Configuring the J2EE Probe to Communicate Through an HTTP/HTTPS Proxy .....	433
Configuring a .NET Probe to Communicate Through an HTTP/HTTPS Proxy .....	434
Proxy from Commander To Mediator/Probes in Load Test Environment .....	434
Proxy Server Configuration for HTTPS .....	436
<b>Appendix I: Diagnostics Component Configuration and Communication Diagrams .....</b>	<b>437</b>
Diagnostics Component Configuration and Communication .....	438
<b>Appendix J: Diagnostics Upgrade Paths for Mercury Products.....</b>	<b>439</b>
Overview of Product Upgrade Paths .....	440
General Upgrade Recommendations .....	440
Product Upgrade Paths .....	441
<b>Index.....</b>	<b>443</b>

---

# Welcome to This Guide

Welcome to the *Mercury Diagnostics 3.6 for J2EE & .NET Installation and User's Guide*. This guide describes how to install and configure the Diagnostics for J2EE & .NET components and how to use Diagnostics for J2EE & .NET with Mercury LoadRunner 8.1 and Mercury Performance Center 8.1.

Diagnostics for J2EE & .NET provides complete visibility into the transaction performance of J2EE & .NET applications. Diagnostics for J2EE & .NET can be fully integrated into LoadRunner and Performance Center to monitor and analyze your J2EE & .NET applications as part of your performance testing process.

---

**Note:** For information about LoadRunner 8.1 or Performance Center 8.1, refer to the relevant product documentation.

---

## How This Guide Is Organized

This guide contains the following parts:

### **Part I Introduction**

Provides a high level overview of the **Mercury Diagnostics for J2EE & .NET** features, components, architecture, and outputs.

### **Part II Installation and Configuration of Diagnostics for J2EE & .NET Components**

Describes how to install and configure the Diagnostics Components - the Commander, Mediator and Probe for .NET. Describes how to install the LoadRunner Diagnostics add-in.

### **Part III Installation and Configuration of Diagnostics Probe for J2EE**

Describes the processes for installing and configuring the J2EE Probe and for configuring the application server on which your J2EE applications run.

### **Part IV Using Mercury Diagnostics for J2EE & .NET**

Describes the process of configuring LoadRunner and Performance Center so that Mercury Diagnostics for J2EE/.NET can be enabled for use in a load test. Describes the screens, graphs, tables and charts that are used to present the diagnostic performance metrics for the application that is being analyzed.

### **Part V Appendixes**

Describes more detailed and advanced information about selected topics.

# Mercury Diagnostics for J2EE & .NET Online Documentation

## Online Documentation for LoadRunner

You can access the **Readme**, which provides last-minute news and information about Diagnostics for J2EE & .NET, from the **Start** menu.

You can also access an online PDF version of the Mercury Diagnostics for J2EE & .NET guide. To access the PDF guide, select **Start > Programs > Mercury LoadRunner > J2EE & .NET**.

Online Help is available from specific LoadRunner windows by clicking in the window and pressing **F1** or clicking the **Help** button.

## Online Documentation for Performance Center

Online Help is available from specific Performance Center windows by clicking the **Help** button in the relevant window.

## Additional Online Resources

**Customer Support Online** 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>. Alternatively, click the **Help** button and choose **Customer Support Online**.

**Mercury Interactive on the Web** 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. Click the **Help** button and choose **Mercury Interactive on the Web**. 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 **Mercury Diagnostics for J2EE & .NET**.  
Note that if **Mercury Diagnostics for J2EE & .NET** 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:

<b>UI Elements</b>	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 <b>Save</b> button.”
<i>Arguments</i>	This style indicates method or function arguments and book titles. For example, “Refer to the <i>Mercury User’s Guide</i> .”
< <b>Replace Value</b> >	Angle brackets enclose a part of a file path or URL address that should be replaced with an actual value. For example, < <b>MyProduct installation folder</b> >\bin.
Example	This style is used for examples and text that is to be typed literally. For example, “Type Hello in the edit box.”
<b>Function_Name</b>	This style indicates method or function names. For example, “The <b>wait_window</b> statement has the following parameters:”
CTRL	This style indicates keyboard keys.
[ ]	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

---

## **Mercury Diagnostics for J2EE & .NET Product Overview**

This chapter introduces you to Mercury Diagnostics for J2EE & .NET by giving you a high level overview of its features, components, architecture, and outputs.

The following topics are included in this chapter:

- Introducing Mercury Diagnostics for J2EE & .NET
- Diagnostics for J2EE & .NET Components
- Diagnostics Data Flow
- Diagnostics Performance Metric Processing
- Diagnostics Layers
- Diagnostics Performance Metric Reporting

## Introducing Mercury Diagnostics for J2EE & .NET

Diagnostics for J2EE & .NET is an application performance optimization solution that is designed to help you improve the performance of your applications on the J2EE and .NET platforms throughout the application lifecycle. Diagnostics for J2EE & .NET has been integrated with Mercury's Application Diagnostics and Application Monitoring solutions to provide you with the insight and information that you need build, develop, test, and monitor applications that perform efficiently and effectively.

Diagnostics for J2EE provides you with the capability to monitor the performance of your applications that run on most of the J2EE compliant application servers. You install a Mercury Diagnostics Probe for J2EE on the application server instances where the application that you want to monitor runs. The J2EE Probe collects performance metrics on the servlets, JSPs, EJBs, JNDI, JDBC, JMS, and Struts method calls that are performed by your application, as well as on the custom classes that you specify.

Diagnostics for .NET provides you with the capability to monitor the performance of your applications that run on the Microsoft .NET Framework. The Mercury Diagnostics Probe for .NET uses runtime instrumentation to capture method latency information from specified applications. By default, the .NET Probe captures methods from the ASP and ADO tiers and MSMQ. Custom business logic methods can be captured by creating a custom instrumentation specification file for your application.

The J2EE and .NET Probes send the metrics to the Diagnostics Mediator where the information for events is filtered and aggregated. The Mediator sends the aggregated performance information to LoadRunner and the Commander where it is displayed using graphs and reports. The information presented enables you to analyze the performance of your application.

## Diagnostics for J2EE & .NET Components

You must install and configure the components of Diagnostics for J2EE & .NET so that you can gather, process, and review the performance metrics for your application. This section introduces you to the Diagnostics Components. For information on installing the Diagnostics Components, see Part II, “Installation and Configuration of Diagnostics for J2EE & .NET Components.”

Diagnostics for J2EE & .NET consists of the following components:

- ▶ Diagnostics Probes for J2EE and .NET
- ▶ Diagnostics Mediator
- ▶ Diagnostics Commander

### Diagnostics Probes for J2EE and .NET

The Mercury Diagnostics Probes are installed on the machine that hosts your application. The Probes are responsible for capturing events from your application and sending the performance metrics to a Mediator. The Probe captures events such as method invocations, collection sites, the beginning and end of business transactions and server transactions.

The Mercury Diagnostics Probe for J2EE is a lifecycle probe that works with many of Mercury’s Diagnostics products such as LoadRunner, Business Availability Center, Performance Center, and Deep Diagnostics.

For a diagram showing the data flow, see “Diagnostics Data Flow” on page 7.

You must configure the Probe and the application environment to enable the Probe to monitor your application. For information on how to configure the .NET Probe and the application environment, see Chapter 6, “Installing the Mercury Diagnostics Probe for .NET”. For information on how to configure the J2EE Probe and the application environment, see “Introducing J2EE Probe Installation and Application Server Configuration” on page 97

## **Diagnostics Mediator**

The Diagnostics Mediator is responsible for receiving raw data from the Probes, pre-processing and aggregating the raw data, and passing the aggregated data to the Diagnostics Commander.

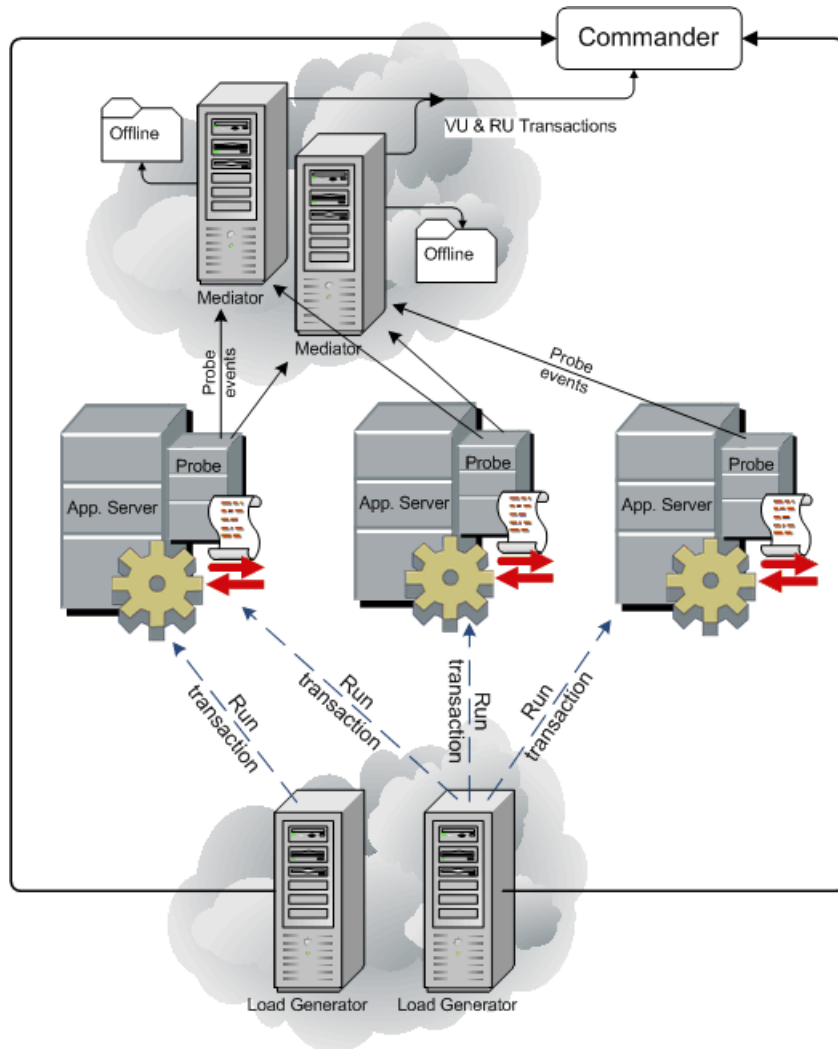
The Mediator does the preprocessing and aggregation of the raw data from the Probes so that the Probes will not have to do this work. Since the Mediator does the this piece of the processing for the Probes, they have a negligible effect on the performance of the applications that they monitor.

## **Diagnostics Commander**

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

## Diagnostics Data Flow

The following diagram illustrates how data flows between the components of Diagnostics for J2EE & .NET when used in an Application Diagnostics mode with your Mercury load testing application.



### **Description of Data Flow**

- 1** The Load Generators run Vuser transactions on one or more of the application servers.
- 2** The Controller (not shown) sends the Commander a list of the Probes that you have selected to take part in the load test.
- 3** The Probe, installed on an application server, captures the events from the application and transmits the event information to the Mediator.
- 4** The load generator sends an End of Logical Transaction via the Commander to the Mediator.
- 5** The Mediator transmits the aggregated event data to the Commander. In addition, the Mediators store the data locally for offline analysis.
- 6** The data is transmitted to the Commander, which stores it locally.
- 7** The Commander prepares the metrics to be presented to the user in graphs and charts that indicate the performance of the monitored applications.

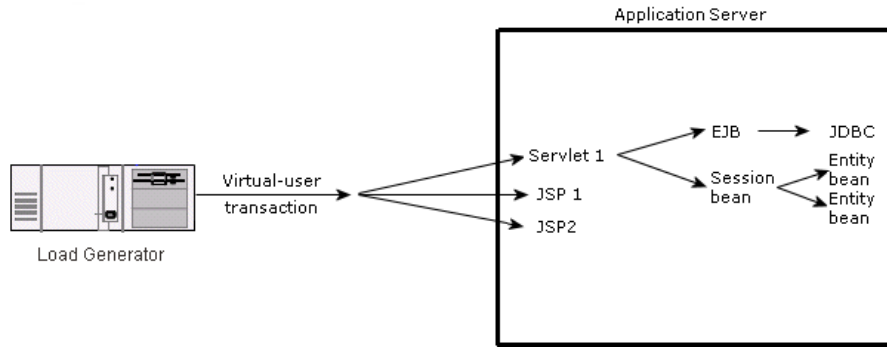
## **Diagnostics Performance Metric Processing**

Diagnostics for J2EE & .NET gathers and aggregates the performance metrics from your application based upon the following rules:

- ▶ Performance information from multiple probes on multiple servers can be gathered, aggregated, and reported.
- ▶ The J2EE Probe monitors each thread of servlets, JSP pages and other layers on the application server that have been directly invoked in the context of a Vuser transaction using the VuGen Web protocol.
- ▶ If a directly-invoked thread spawns additional threads, the additional threads can also be monitored if the **Monitored Server Requests** option is selected during LoadRunner / Performance Center Diagnostics configuration (selected by default).
- ▶ The transaction time that is displayed by Diagnostics for J2EE & .NET for transactions only includes the time that the transaction spends in the J2EE/.NET server, and therefore, does not include the transaction's think time.



- The Probe breaks down the transactions into separate server requests. The performance metrics are calculated for the complete transaction as well as for each server request.



## Diagnostics Layers

Diagnostics groups the performance metrics for classes and methods into layers based upon the resources that are used to perform the processing. The layers make it easier for you to isolate and identify the areas of the system that may be contributing to performance issues.

The following section lists the layers that have been defined for the platforms that Diagnostics monitors. The methods and classes that are associated with each Layer are defined in the property files for the Probe. For more information see “Specifying Layers to Instrument” on page 378.

For information on the viewing the performance metrics by layer see “Analyzing Performance with the Layer Screens” on page 255.

### J2EE Layers

The default J2EE layers are:

- Servlets
- JSPs
- Struts

- Entity Beans
- Session Beans
- JNDI
- JDBC
- JMS

### **.NET Layers**

The default .NET layers are:

- Web layer WEB.ASP (Active Server Pages)
- Database layer DB.ADO (ActiveX Data Objects)
- Messaging layer MSG.MSMQ (Microsoft Message Queuing)

### **Portal Layers**

Diagnostics groups the performance metrics for the method calls associated with processing for portals into layers.

### **BEA WebLogic Portal 8.1**

- Entitlement
- User Profile
- Personalization
- Content

### **SAP Portal Layers**

- Portal Runtime
- Portal Authorization
- Portal Authentication
- Portal JNDI
- Portal Components Runtime
- Portal Components Content Generation

- Portal Components Services
- Portal Components Service
- R3
- Dyn Page
- JSP Dyn Page

## Diagnostics Performance Metric Reporting

While your Mercury performance testing application is executing Vuser transactions against your application, Diagnostics for J2EE & .NET is gathering, analyzing and displaying the performance metrics. Diagnostics displays the performance metrics in real-time as the load test is executing.

The performance metrics that Diagnostics displays include latency and throughput for transactions, server requests, layers, and virtual machines. Diagnostics also keeps track of exceptions, and timeouts. Diagnostics displays the performance metrics in graphs and tables which allow the user to drill down into specific transactions or server requests to discover the method calls that contributed to the poorly performing processes.

---

**Note:** The LoadRunner Analysis Web Breakdown and Database Breakdown graphs can provide you with additional performance metrics to enhance your understanding of your applications performance characteristics. For more information on these graphs, refer to the *Mercury LoadRunner Analysis User's Guide*.

---

Diagnostics for J2EE & .NET uses the following screens to display your application's performance metrics:

- ▶ Transaction Trends - See "Analyzing Performance with the Transaction Trends Screen" on page 244.
- ▶ Layer Breakdown - See "Analyzing Performance with the Layer Screens" on page 255.
- ▶ Server Request Trends and Server Request Breakdown - See "Analyzing Performance with the Server Request Screens" on page 249.
- ▶ Call Profile - See "Analyzing Performance with the Aggregate Profile Screen" on page 267.
- ▶ Virtual Machine Trends - See "Analyzing Performance with the Virtual Machines Screen" on page 261.
- ▶ Diagnostics Overview - See "Analyzing Performance Using the Diagnostics Overview Screen" on page 240.

# Part II

---

## Installation and Configuration of Diagnostics for J2EE & .NET Components



# 2

---

## Preparing to Install Mercury Diagnostics for J2EE & .NET

This chapter gives you the information and instructions that will help you to plan and prepare for the installation and configuration of the Diagnostics for J2EE & .NET components.

The chapter includes the following sections:

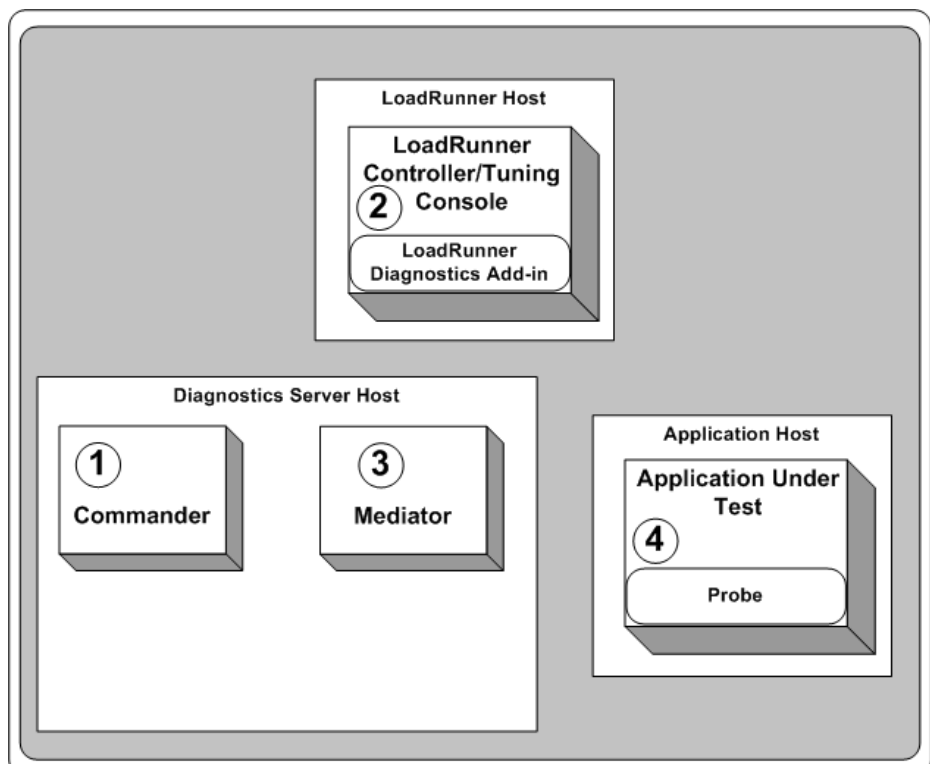
- Recommended Deployment Configuration
- Host System Requirements for the Diagnostics Components
- Upgrading from Earlier Versions of Diagnostics
- Other Considerations Before Installing the Diagnostics Components
- Recommended Order of Installation
- Planning the Installation

## Recommended Deployment Configuration

This section provides you with a recommended deployment configuration for LoadRunner 8.1 or Performance Center 8.1 and Diagnostics for J2EE & .NET. More detailed instructions about the installation and configuration of the Diagnostics components are provided in the chapters of this document that follow.

### Deployment with LoadRunner

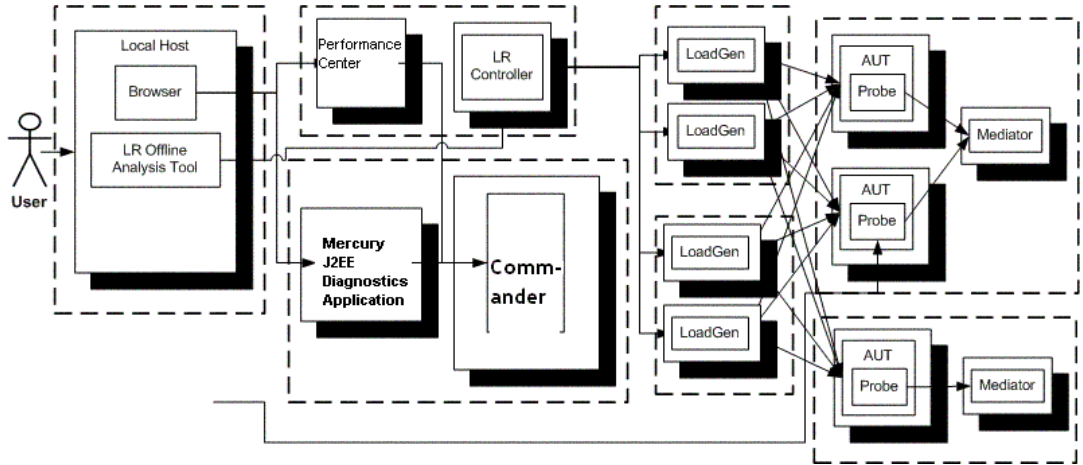
The following diagram shows a typical deployment configuration for 10 to 15 VMs, with a moderate load using a total of three host machines. For deployments with larger loads or more VMs you may want to consider hosting the Mediator on its own machine. The deployment that will work best for you is dependent upon the number of VMs that you are monitoring and the size of the load that you are running. This diagram has been provided as a reference for you as you review the order of installation.





## Deployment with Performance Center

The following diagram shows a suggested architecture for deploying Mercury Diagnostics for J2EE with Performance Center:



## Host System Requirements for the Diagnostics Components

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

The following section describes the recommended system configurations for hosting the Mercury Diagnostics for J2EE & .NET components. Please refer to the deployment diagram in the previous section to understand the host machines that are described in this section.

### LoadRunner 8.1

Refer to the *Mercury LoadRunner Installation Guide* for information about the system configuration for this machine.

### Performance Center 8.1

Refer to the *Mercury Performance Center System Configuration and Installation Guide* for information about the system configuration for the Performance Center machines.

#### Commander Host Machine

The system requirements presented for the Commander are for a large implementation.

<b>Computer/Processor:</b>	Windows: Dual Processor Pentium 2.4GHz Solaris: 800MHz
<b>Operating System:</b>	Windows 2000/2003 Server SP4 Solaris
<b>Memory:</b>	2 GB RAM
<b>Free Hard Disk Space:</b>	10 GB of disk space after installation

#### Mediator Host Machine

The system requirements described in this section are applicable for deployments where the Mediator is the only diagnostics component that has been installed on the Mediator host machine. The Mediator should be installed on its own host machine when there are a large number of VMs being monitored.

<b>Computer/Processor:</b>	Windows: Dual Processor Pentium 2.4GHz Solaris: 800MHz
<b>Operating System:</b>	Windows 2000/2003 Server SP4 Solaris
<b>Memory:</b>	1 GB RAM
<b>Free Hard Disk Space:</b>	10 GB of disk space after installation

### **Application Host Machine**

The Application host machine is the machine where you have installed the your application and the Diagnostics Probe(s). The only requirement for the J2EE Probe is that there be 60 MB of memory in addition to the memory required by the system under test. For the .NET Probe there should be 10 MB of memory in addition to the memory required by the system under test.

## **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 where other Mercury products must be upgraded so that the features of Diagnostics can be accessed, you should review the instructions in Appendix J, “Diagnostics Upgrade Paths for Mercury Products.” These instructions will help you to understand how to upgrade your current Mercury products and the Diagnostics components.

## **Other Considerations Before Installing the Diagnostics Components**

---

**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 Add-in. Performance Center does not require any add-in.
- ▶ The settings for the time and the time zones for 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 Commander**

- ▶ Do not install a Probe on the same machine as the Commander.

---

**Note:** This recommendation applies only to a system that is experiencing high load.

---

- ▶ The machine that hosts the Commander must be located in the same LAN segment as the hosts for the Controller/Tuning Console.

### **Mercury Diagnostics Mediator**

- ▶ The Mercury Diagnostics Mediator, and each of the Probes that are expected to be able to communicate with it, must have the same Logical LAN ID property. For more information on setting the Logical LAN ID for the Mediator, see “Installing the Mercury Diagnostics Mediator” on page 57, .
- ▶ The Mediator can be installed on the machine that is the host to the Commander for an average size deployment. If you are monitoring a larger number of VMs you should consider moving the Mediator to a machine that is not the host to any other diagnostics components.
- ▶ If the Mediator and Commander are installed on the same machine and you want to run them concurrently, you will need to copy the launch file from the Commander to the Mediator and run them both from the Mediator. For more information, see the note on page 74

### **Mercury Diagnostics Probe for J2EE**

- ▶ The J2EE Probe, and the Mediator that it is to communicate with, must have the same Logical LAN ID property. For more information on setting the Logical LAN ID for probes, see the appropriate section of the Probes Installation, on page 99.
- ▶ The J2EE Probe is installed on the same machine as the Java application under test.

### **Mercury Diagnostics Probe for .NET**

- ▶ The .NET Probe, and the Mediator with which it is to communicate, must have the same Logical LAN ID property. For more information on setting the

Logical LAN ID for probes, see the appropriate section of the Probe Installation, in Chapter 6, “Installing the Mercury Diagnostics Probe for .NET”.

- ▶ The .NET Probe is installed on the same machine as the .NET application under test.

## Recommended Order of Installation

Careful planning and preparation for installing the components of Diagnostics for J2EE & .NET can enable you to complete the installation and configuration steps quickly and help you to avoid complications and errors.

Before beginning the preparations for the installation, 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 17.

### **2 Install the Mercury Diagnostics for J2EE & .NET components.**

- ▶ Install the Commander - see Chapter 3, “Installing the Mercury Diagnostics Commander.”
- ▶ Install the LoadRunner Add-in - see Chapter 4, “Installing LoadRunner 8.1 and the LoadRunner Diagnostics Add-in.” Performance Center does not require an add-in.
- ▶ Install the Mediator - see Chapter 5, “Installing the Mercury Diagnostics Mediator.”

- ▶ Install the Probe.
  - For a J2EE environment, see Chapter 8, “Installing the Mercury Diagnostics Probe for J2EE.”
  - For a .NET environment, see Chapter 6, “Installing the Mercury Diagnostics Probe for .NET.”

**3 Configure the application server to work with the probes.**

See Chapter 7, “Introducing J2EE Probe Installation and Application Server Configuration.”

**4 Specify the layers that you want to view.**

See “Specifying Layers to Instrument” on page 378.

**5 Configure LoadRunner / Performance Center to use Mercury Diagnostics for J2EE & .NET.**

See Chapter 12, “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1” and Chapter 13, “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1.”

Once you have set up Diagnostics for J2EE & .NET and created and initiated your load test, you can view the performance of your J2EE/.NET applications as the load test executes. For details, see Part IV, “Using Mercury Diagnostics for J2EE & .NET.”

## Planning the 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 host machines within your network topography.

The following tables have been provided to help you plan the installation of the Diagnostics components.

## LoadRunner and Performance Center Configuration for Diagnostics for J2EE & .NET

Before you configure LoadRunner, you need to install the Diagnostics Add-in on the LoadRunner host machine. Performance Center does not require any add-in.

To configure LoadRunner and Performance Center to work with the Diagnostics components, you must be able to provide the following information.

Information Required	Where to find it	Value
<b>Commander Host name or IP address</b>	System Health Monitor	
<b>Commander Port number</b>	System Health Monitor Default value is <b>2006</b>	

For detailed configuration instructions for LoadRunner, see Chapter 12, “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1.” For detailed configuration instructions for Performance Center see Chapter 13, “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1.”

## Mediator

Information Required	Where to find it	Value
Commander Host name or IP address	System Health Monitor	
Commander Host Port number	System Health Monitor Default value is <b>2006</b>	
Name of Mercury product(s) that will use the Mediator.	Choose according to product license. <ul style="list-style-type: none"> <li>• <b>Performance Center 8.1 / LoadRunner 8.1</b></li> <li>• <b>Mercury Business Availability Center 5.0</b></li> </ul>	
Logical LAN ID	This is user defined at the time that the Mediator or Probes are installed  The Mediator must have the <b>same LAN ID as the Probes</b> that are expected to be able to communicate with it.	
Will the Mediator be used in a Mercury Managed Services environment?		



**J2EE Probe**

Information Required	Where to find it	Value
Name of Mercury product(s) that will use the Probe.	Choose according to product license. <ul style="list-style-type: none"> <li>• Performance Center 8.1 / LoadRunner 8.1</li> <li>• Mercury Business Availability Center 5.0</li> <li>• Deep Diagnostics</li> </ul>	
Probe name	A <i>unique</i> string; Created by user. <b>Important:</b> The name of the probe should indicate the probe type. For example: J2EEProbe1	
Logical LAN ID	This is user defined at the time that the Mediator or Probes are installed  The Probe must have the <b>same LAN ID as the Mediators</b> that are expected to be able to communicate with it.	
Commander Host name or IP address	System Health Monitor	
Commander Host Port number	System Health Monitor Default value is <b>2006</b>	
Mediator Host name or IP address	System Health Monitor	

Information Required	Where to find it	Value
<b>Mediator Host Port number</b>	System Health Monitor Default value is <b>2612</b>	
<b>Application Server that the J2EE Probe will be monitoring</b>	The host system administrator. Choose one of the following: <ul style="list-style-type: none"> <li>• <b>WebLogic6x</b></li> <li>• <b>WebLogic7x</b></li> <li>• <b>WebLogic8x</b></li> <li>• <b>WebSphere4x</b></li> <li>• <b>WebSphere5x</b></li> <li>• <b>Oracle9ias</b></li> <li>• <b>Oracle10g</b></li> <li>• <b>JBoss3x</b></li> <li>• <b>SAP Web AS 6.40</b></li> </ul>	
<b>Application Server configuration properties</b>	The host system administrator.  The details will vary according to the application sever that you are using.	
<b>Deep Diagnostics install directory</b> (Only if you will be using Deep Diagnostics.)	The location where the Deep Diagnostics Server was installed.	
<b>Deep Diagnostics Application Definition Name</b> (Only if you will be using Deep Diagnostics.)	See the Deep Diagnostics User Documentation for the location of this information.	

Information Required	Where to find it	Value
<p><b>Location of the JRE executable</b></p>	<p>The host system administrator.</p> <p>This will depend upon the type of application server you are configuring.</p> <p>See “Running the JRE Instrumenter” on page 141.</p>	

**.NET Probe**

Information Required	Where to find it	Value
Probe ID	Created dynamically by the Probe at runtime. <b>Default value:</b> <AppDomainName>.NET	
LAN ID	This is user defined at the time that the Mediator or Probes are installed.  The Probe must have the <b>same LAN ID as the Mediators</b> that are expected to be able to communicate with it.	
Web Port Min	System Administrator.  The lowest port number in a range of ports that the commander can use to communicate with its probes.  <b>Default value: 35000</b>	
Web Port Max	System Administrator  The highest port number in a range of ports that the commander can use to communicate with its probes.  <b>Default value: 35100</b>	
Diagnostics Commander URL	URL that the probe will use to register with the Commander.	
Commander Host name or IP address	System Health Monitor	

Information Required	Where to find it	Value
<b>Commander Host Port number</b>	System Health Monitor Default value is <b>2006</b> .	
<b>Mediator Host name or IP address</b>	System Health Monitor	
<b>Mediator Host Port number</b>	System Health Monitor Default value is <b>2612</b> .	



# 3

---

## Installing the Mercury Diagnostics Commander

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

The Commander is the first Diagnostics component installed because it contains the processes that facilitate communication between the load testing application and the other Diagnostics components. The Commander also contains the System Health Monitor that allows you to verify the status and configuration of the Diagnostics components as you proceed with the installation process.

This chapter contains the following sections:

- ▶ Installing the Commander on a Windows Machine
- ▶ Installing the Commander on a UNIX Machine
- ▶ Starting and Stopping the Commander
- ▶ Verifying the Commander Installation
- ▶ Determining the Version of the Commander that is Installed
- ▶ Uninstalling the Commander

## Installing the Commander on a Windows Machine

To install the Commander on a Windows machine:

- 1 Insert the Installation CD into a CD-ROM drive.

For LoadRunner, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Windows Installation - Disc*.

For Performance Center, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Windows Installation - Disc*.

- 2 The installer should start automatically.

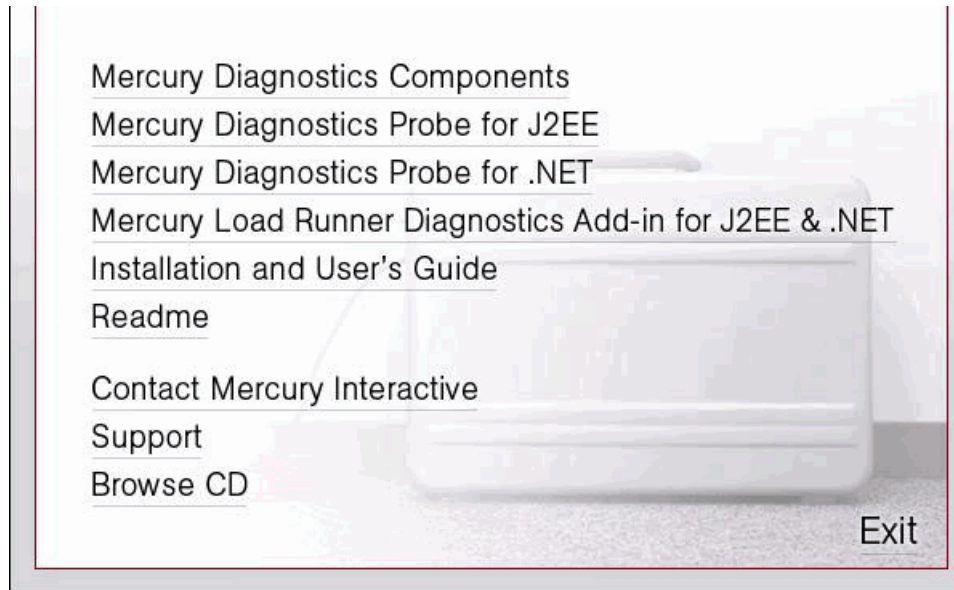
If the installer does not start, run the installer from the Windows Start menu. Click **Start > Run** and then type the location of your CD-ROM drive followed by the name of the installer program, **setup.exe**.

For example, if your CD-ROM drive letter is M, type:

```
m:\setup.exe
```



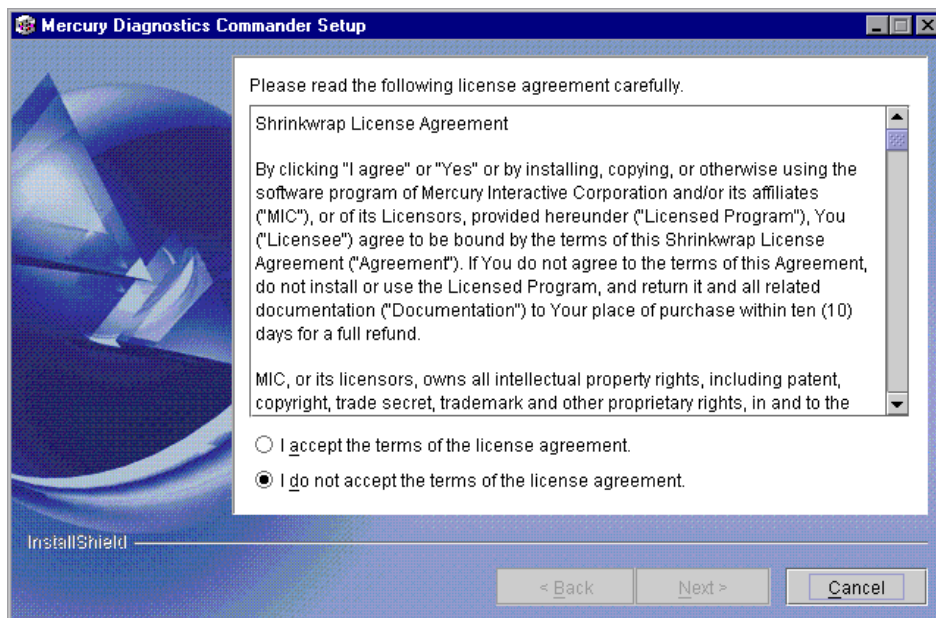
- 3 The installer displays the Mercury Diagnostics for J2EE & .NET main installation menu.



Navigate to the installer for the Commander by clicking the **Mercury Diagnostics Components** link from the menu.

Select the installer for the Commander to initiate the installation process.

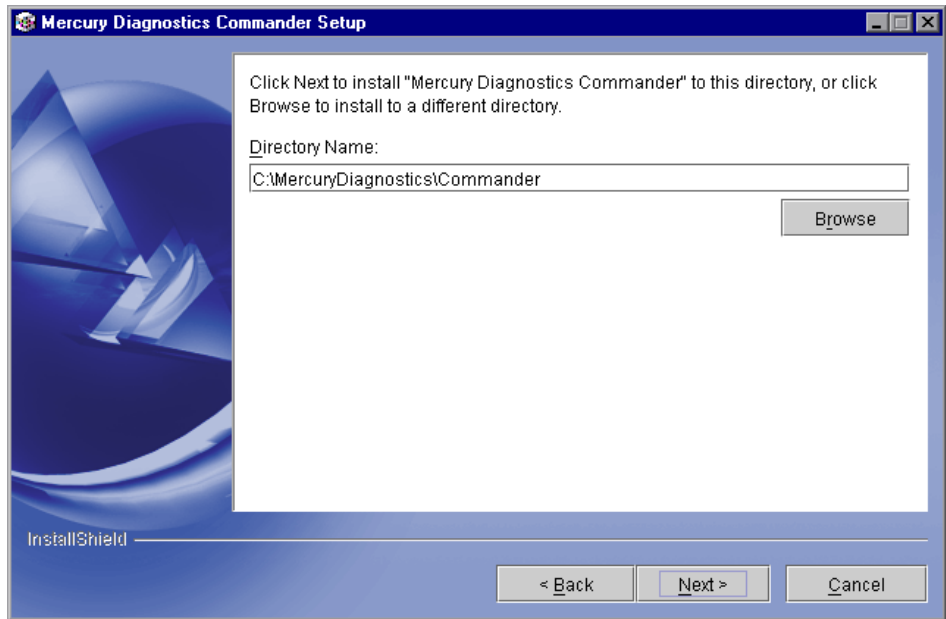
4 The installer displays the software license agreement.



Read the agreement. To accept, select “I accept the terms of the license agreement”.

Click **Next** to proceed with the installation.

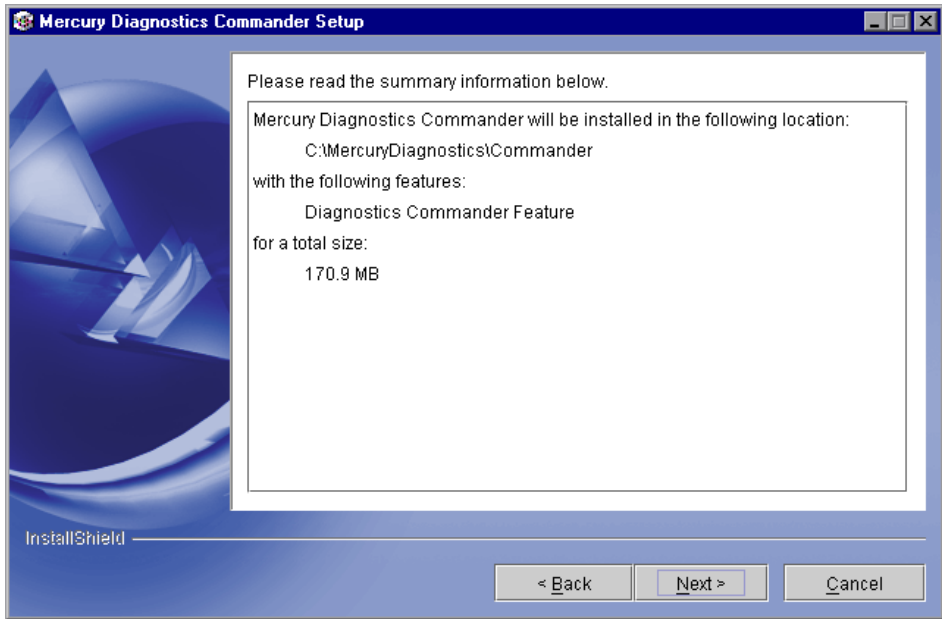
- 5 The installer displays the dialog that allows you to specify the path to the directory where the Commander is to be installed



Ensure that the location where you want the Commander to be installed has been specified in the **Directory Name** text box either by typing in the path to the desired installation directory or by clicking **Browse** to navigate to the desired location.

Click **Next** when you are ready to proceed with the installation.

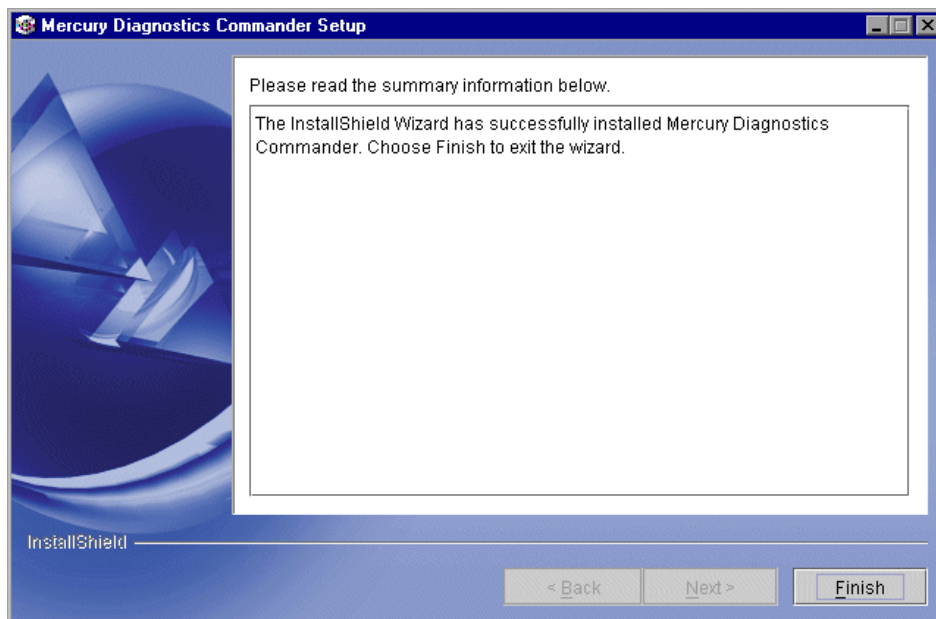
- 6 The installer displays the Pre-Installation Summary Information dialog so that you can review the installation options that you have selected and make any changes before the component is actually installed.



Click **Next** to install the Commander.

When the installation is complete, the Commander is started and an icon is placed on the task bar.

- 7 The installer displays the Post Installation Status dialog to let you know how the installation processing went.



Click **Finish** to close the installer.

## Installing the Commander on a UNIX Machine

Diagnostics Commander installers have been provided for the UNIX platform. The following instructions give you the information necessary to install the Diagnostics Commander in the supported UNIX environments using either a graphics based installation or a console mode installation.

The installer screens that you will see in a graphics based installation are the same as those documented for the Windows installer in “Installing the Commander on a Windows Machine” on page 32.

---

**Note:** For LoadRunner, The UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

For Performance Center, The UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

The installer is located in the **MercuryDiagnostics/Commander** folder.

---

**To install the Commander on a UNIX machine:**

---

**Note:** The following instructions and screen shots are for a Commander installation on a Solaris machine.

---

- 1** Insert the Mercury Diagnostics for J2EE & .NET Disc and locate the installer folder **MercuryDiagnostics/Commander**.
- 2** Copy the installer to the machine where the Commander is to be installed.
- 3** Change the mode of the installer file to make it executable.

**4** Execute the installer.

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

```
./install.sh -console
```

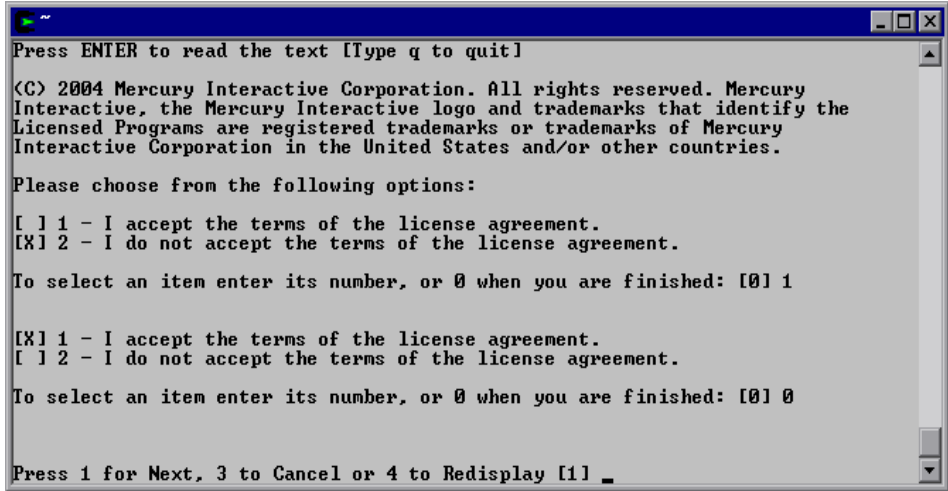
The installer will start and display the license agreement as shown in the following step.

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

```
./install.sh
```

The installer will display the same screens that are displayed for the Windows installer as shown in “Installing the Commander on a Windows Machine” on page 32.

5 The installer begins by displaying the software license agreement.



Read the agreement.

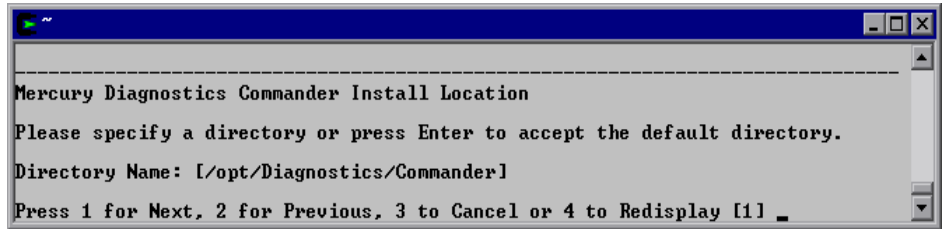
As you read, you may press Enter to move to the next page of text or type **q** to jump to the end of the license agreement.

Enter **1** to accept the terms of the license agreement and then enter **0** to accept your selection.

Enter **1** to continue.



- 6 The installer prompts you to specify the path to the directory where the Commander is to be installed.



---

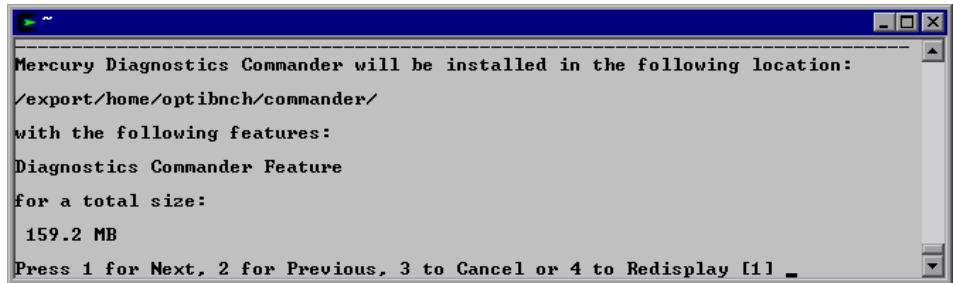
**Note:** The user id that is being used to run the installer must have permission to write to this directory.

---

Specify the directory where you want the Commander to be installed or press Enter to accept the default directory.

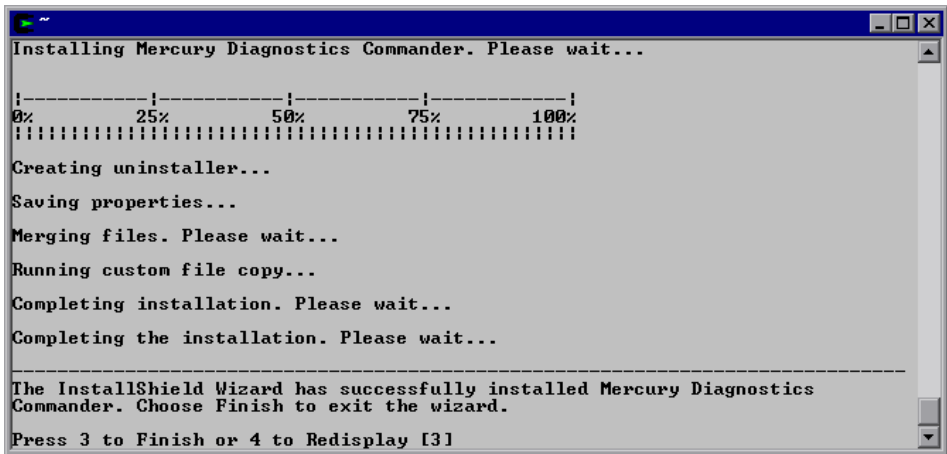
Enter **1** to continue.

- 7 The installer displays the Pre-Installation Summary Information so that you can review the installation options that you have selected and make any changes before the component is actually installed.



Enter **1** to start the installation of the Commander. If you would like to make changes to the options that you selected enter **2** to return the previous prompts.

- 8 The installer begins installing the Commander and displays a progress bar and status messages so that you can monitor the processing.



- 9 After installation is complete, type 3 to exit the installer.

You must start the Commander after the installer has finished. See “Starting and Stopping the Commander” on page 42 for the instructions.

---

**Note:** The user id under which the Commander will run must have read/write/execute permissions so that the Commander can write the log files and create the temporary directories and storage files that are used at run time.

---

## Starting and Stopping the Commander

### Instructions for a Windows Machine

To start the Commander on a Windows machine:

Choose **Start > Programs > Mercury Diagnostics Commander > Start Mercury Diagnostics Commander**

**To stop the Commander on a Windows machine:**

Choose **Start > Programs > Mercury Diagnostics Commander > Stop Mercury Diagnostics Commander**

**Instructions for a Solaris Machine****To start the Commander on a Solaris machine:**

- 1 Ensure that the M\_LROOT environment variable is defined as the root directory of the Commander.**

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

```
$> setenv M_LROOT /opt/MercuryDiagnostics/Commander
```

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 Use m\_daemon\_setup with the -install option as in the following example:**

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

---

**Note:** If you are running the Commander and the Mediator on the same machine, you may have problems running them concurrently. To enable the Commander and the Mediator to run concurrently on the same machine:

- Copy the launch file from the Commander to the Mediator by entering the following:

```
cp <commander_install_dir>\launch_service\dat\nanny\commander.nanny  
<mediator_install_dir>\launch_service\dat\nanny\.
```

- ▶ Define the **M\_LROOT** environment variable as the root directory of the *Mediator* and not of the Commander. For example, in *tcsh*, you could enter: `$> setenv M_LROOT /opt/MercuryDiagnostics/Mediator`
  - ▶ Start the Mediator. When you start the Mediator, the Commander will also start running.
- 

**To stop the Commander on a Solaris machine:**

Use **m\_daemon\_setup** with the `-remove` option as in the following example:

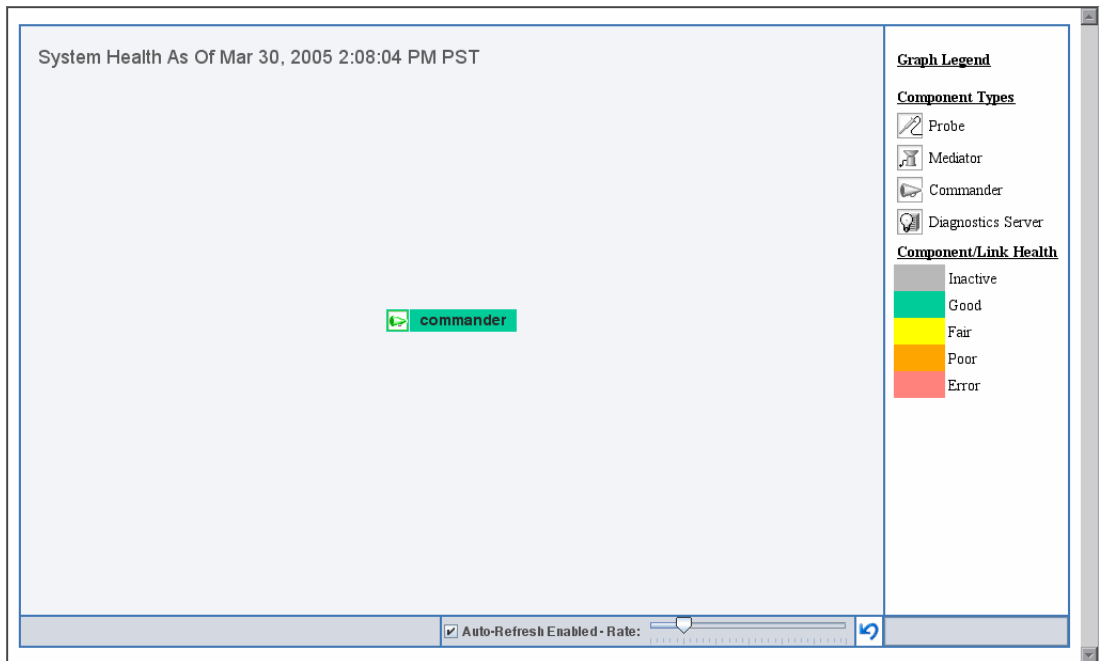
```
$> ./m_daemon_setup -remove
```

## Verifying the Commander Installation

To verify that the Commander has been installed correctly and that it has started properly, use the System Health Monitor. For instructions, see Appendix A, “Using the System Health Monitor.”

If you have been following the recommended installation sequence, after you have installed the Commander you will be able to use the System Health Monitor to verify that the Commander was installed and started.

The Commander should be the only component displayed on the System Health Monitor as shown in the following screen image:



The System Health Monitor is part of the Commander component. If you are unable to access the System Health Monitor after the Commander has been installed, then either you are entering an incorrect URL or the Commander did not start. See “Starting and Stopping the Commander” on page 42 for instructions on starting the Commander.

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.

## Determining the Version of the Commander that is Installed

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

### To determine the version of the Commander:

Locate the version file `<commander_install_dir>\version.txt`. The file contains the 4 digit version number, as well as the build number.

---

**Note:** The version of the Diagnostics Commander and the version of the LoadRunner Diagnostics Add-In must be exactly the same. See “Version Mismatch Between Diagnostics Commander and LoadRunner Add-In” on page 317 for more information.

---

## Uninstalling the Commander

### To uninstall the Diagnostics Commander from a Windows machine:

- ▶ On a Windows machine execute **uninstall.exe** which is located in the `<commander_install_dir>\_uninst` directory.
- ▶ You can also uninstall the Commander from the Start menu by selecting **Start > Programs > Mercury Diagnostics Commander > Uninstall Mercury Diagnostics Commander**.

**To uninstall the Diagnostics Commander from a Solaris machine:**

- 1** Locate the **uninstall** executable in the <commander\_install\_dir>/\_uninst directory.
- 2** Execute **uninstall** with the **-console** option as in the following example:

```
$>./uninstall -console
```





# 4

---

## Installing LoadRunner 8.1 and the LoadRunner Diagnostics Add-in

To enable the Diagnostics for J2EE & .NET functionality within LoadRunner, you must install the Mercury LoadRunner Diagnostics Add-in for J2EE & .NET. You can only install the Diagnostics Add-in once you have already installed the Diagnostics Commander and Mercury LoadRunner 8.1.

---

**Note:** For Performance Center it is not necessary to install any add-in to enable the Diagnostics for J2EE & .NET functionality.

---

This chapter contains the following sections:

- Understanding the LoadRunner Diagnostics Add-in
- Installing Mercury LoadRunner 8.1
- Installing the Mercury LoadRunner Diagnostics Add-in for J2EE & .NET
- Configuring LoadRunner for Diagnostics for J2EE & .NET

## Understanding the LoadRunner Diagnostics Add-in

The LoadRunner Diagnostics Add-in enables you to access Diagnostics functionality from within LoadRunner. Once the LoadRunner Diagnostics Add-in 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 Commander and the version of the LoadRunner Diagnostics Add-In must be exactly the same. See “Version Mismatch Between Diagnostics Commander and LoadRunner Add-In” on page 293 for more information.

---

## Installing Mercury LoadRunner 8.1

If you have not yet installed LoadRunner, you must install it before you can install the Mercury LoadRunner Diagnostics Add-in for J2EE & .NET. For instructions on installing LoadRunner, refer to the *Mercury LoadRunner Installation Guide*.

## Installing the Mercury LoadRunner Diagnostics Add-in for J2EE & .NET

The Diagnostics Add-in must be installed on the host machine for the LoadRunner Controller.

---

**Note:** If LoadRunner Analysis is not installed on the same machine as the LoadRunner Controller or Tuning Console, you must also install the Diagnostics Add-in on the host machine for LoadRunner Analysis.

---

### To install the LoadRunner Diagnostics Add-in:

- 1 Insert the Installation CD into a CD-ROM drive.

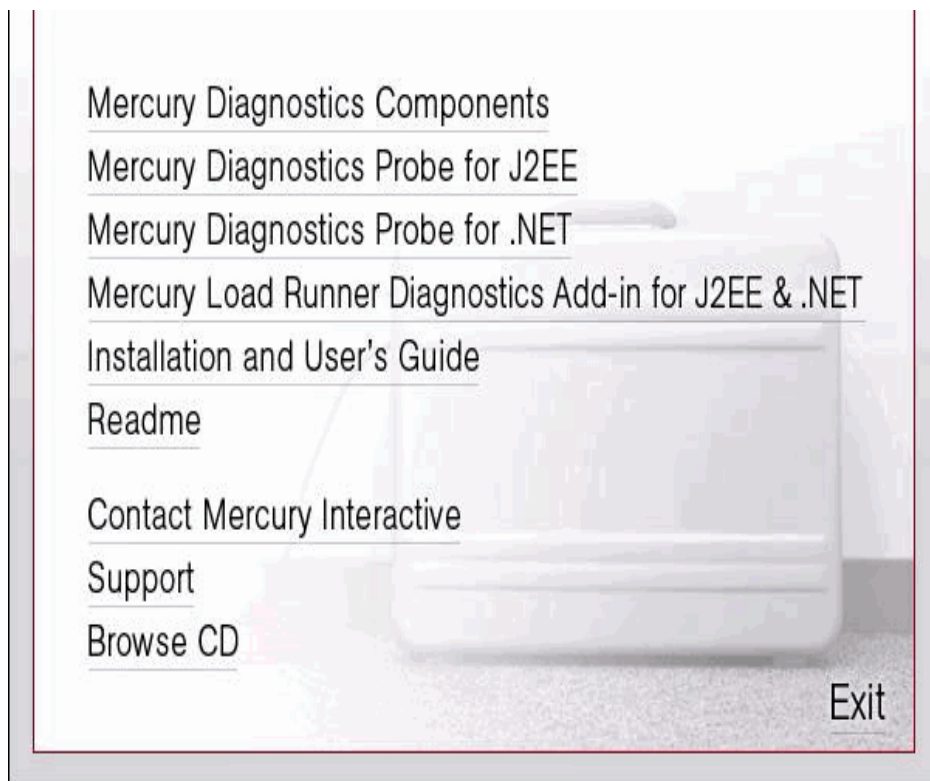
The installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Windows Installation - Disc*.

- 2 If the installer does not automatically start, select **Start > Run** and then enter the mapping of your CD-ROM drive, followed by **setup.exe**.

For example, if your CD-ROM drive is mapped to the “m” drive, enter:

```
m:\setup.exe
```

- 3 The Mercury Diagnostics for J2EE & .NET Setup program begins, and displays the main installation screen.



Click **Mercury LoadRunner Diagnostics Add-in for J2EE & .NET** to start the InstallShield Wizard.

- 4 The software license agreement is displayed. Read the agreement and click **Yes** to accept it. The Registration Information dialog box opens.

**LoadRunner Controller 8.1 J2EE\NET Diagnostics AddIn Setup**

**Registration Information** **MERCURY™**

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.

Name:

Company:

Maintenance number:

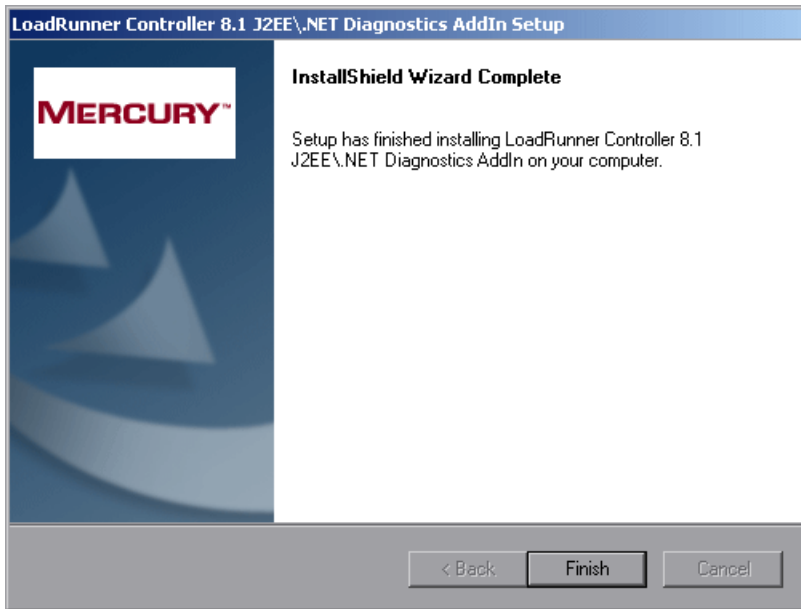
InstallShield

< Back   Next >   Cancel

- 5 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, displaying the Copying files progress bar.

- 6 When the installation process is complete, the installation wizard displays a confirmation message.



Click **Finish** to close the dialog box and to complete the installation.

---

**Note:** If you are installing the LoadRunner Diagnostics Add-in 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 Diagnostics for J2EE & .NET**

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 Diagnostics for J2EE & .NET on LoadRunner 8.1” on page 207 for detailed instructions for configuring LoadRunner.





# 5

---

## Installing the Mercury Diagnostics Mediator

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

The chapter contains the following sections:

- Installing the Mediator on a Windows Machine
- Installing the Mediator on a UNIX Machine
- Starting and Stopping Mediators
- Verifying the Mediator Installation
- Troubleshooting Mediator Issues
- Configuring the Mediator
- Determining the Version of the Installed Mediator
- Uninstalling the Mediator
- Upgrading to a Newer Version of the Mediator

## Installing the Mediator on a Windows Machine

To install the Mercury Diagnostics Mediator on a Windows machine:

- 1 Insert the Installation CD into a CD-ROM drive.

For LoadRunner, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Windows Installation - Disc*.

For Performance Center, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Windows Installation - Disc*.

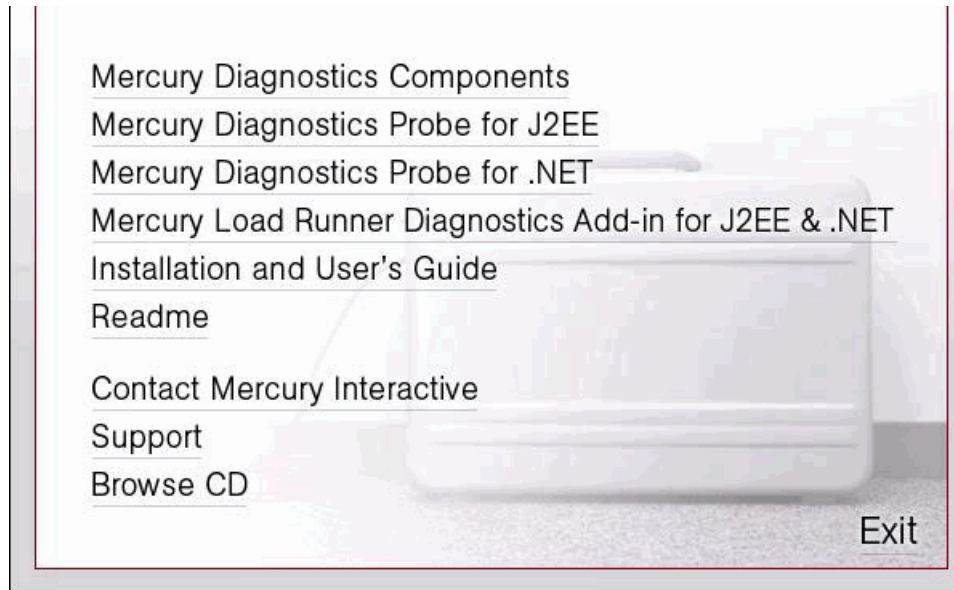
- 2 The installer should start automatically.

If the installer does not start, run the installer from the Windows Start menu. Click **Start > Run** and then type the location of your CD-ROM drive followed by the name of the installer program, **setup.exe**.

For example, if your CD-ROM drive letter is M, type:

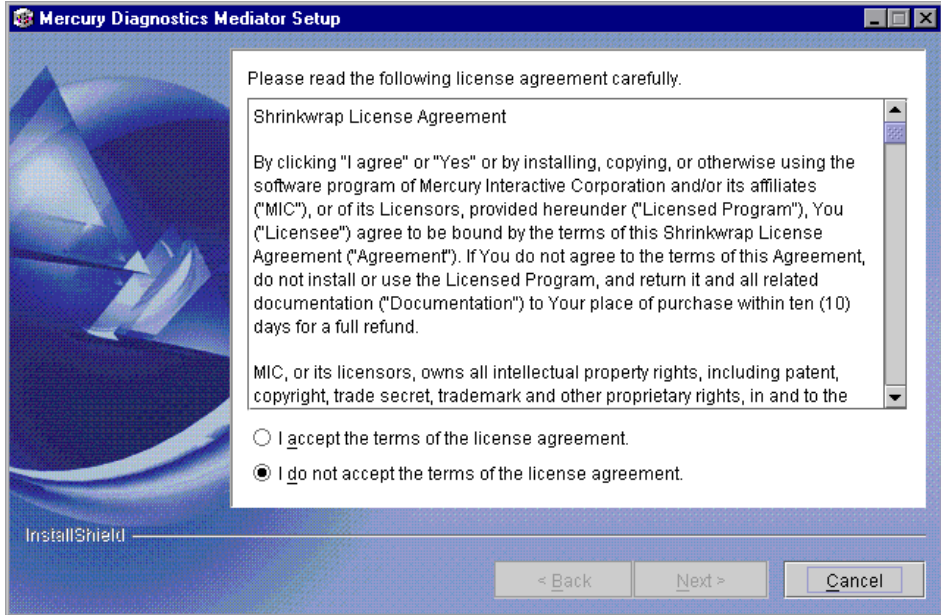
```
m:\setup.exe
```

- 3 The installer displays the Mercury Diagnostics for J2EE & .NET main installation menu.



Click **Mercury Diagnostics Components** and select the installer for the Mediator to initiate the installation process.

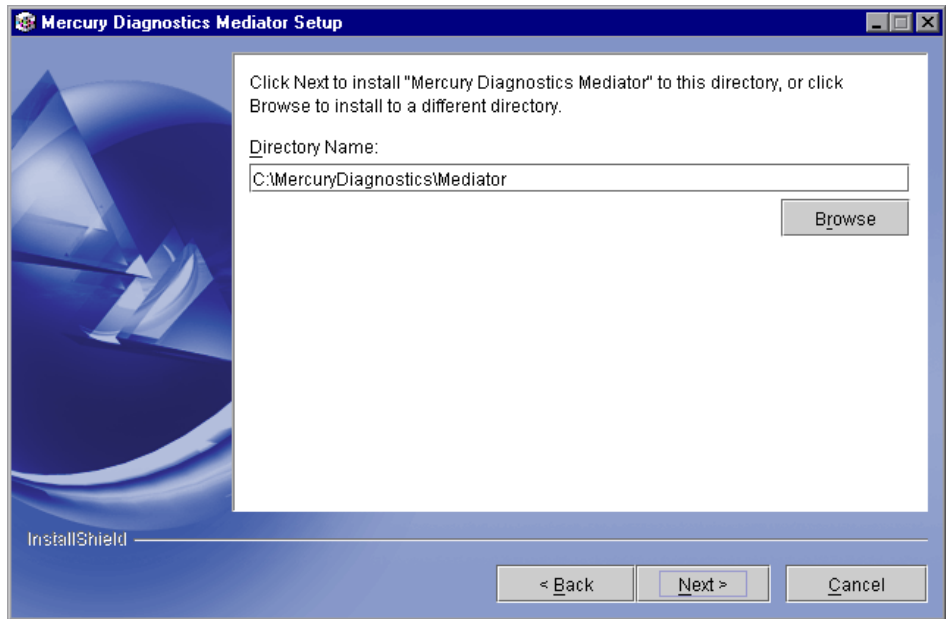
4 The installer displays the software license agreement.



Read the agreement and select “I accept the terms of the license agreement” to accept the agreement.

Click **Next** to proceed with the installation.

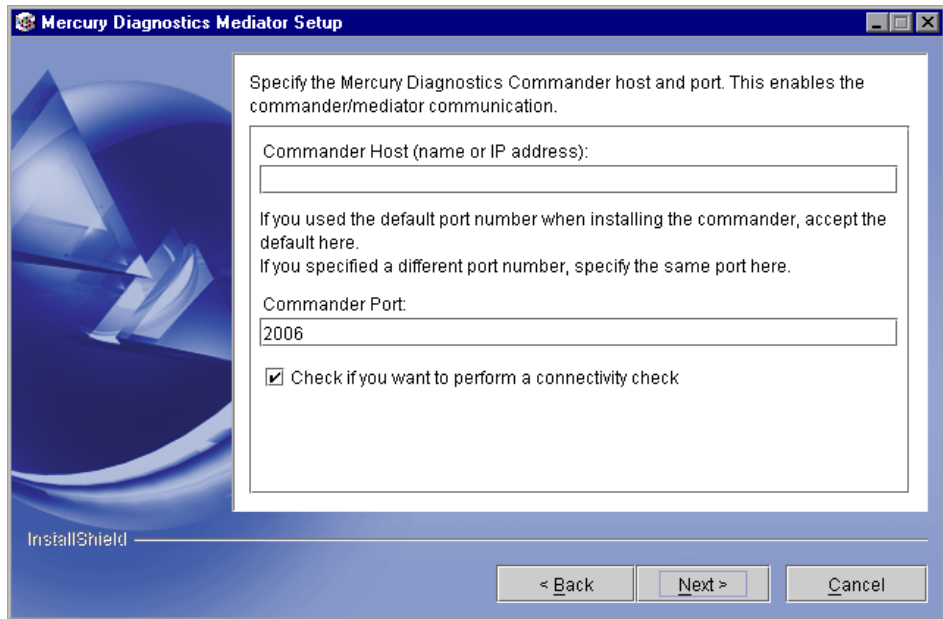
- 5 The installer displays the dialog where you specify the path to the directory where you want the Mediator to be installed.



Ensure that the location where you want the Mediator to be installed has been specified in the Directory Name text box either by typing in the path to the desired installation directory or by clicking **Browse** to navigate to the desired location.

Click **Next** when you are ready to proceed with the installation.

- 6 The next window asks you for information about the Commander that enables communication between the Mediator and the Commander.



- ▶ Enter the host name or IP address of the machine on which the Commander is installed.

---

**Note:** 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.

For information about ensuring that the correct Mediator host name is used when there is a firewall or NAT in place or where your host machine is multi-homed see “Uninstalling the Mediator” on page 76.

---

- ▶ Enter the Commander’s port number.

The default port number is 2006. If you specified a different port number when installing the Commander, specify the same port number here.

Indicate whether you want to perform a connectivity check to make sure that the Commander's host name can be resolved. The connectivity check will let you know right away if you have made an error in the information that you provided about the Commander, or if there is a communication problem between the Commander's host machine and the Mediator's host machine.

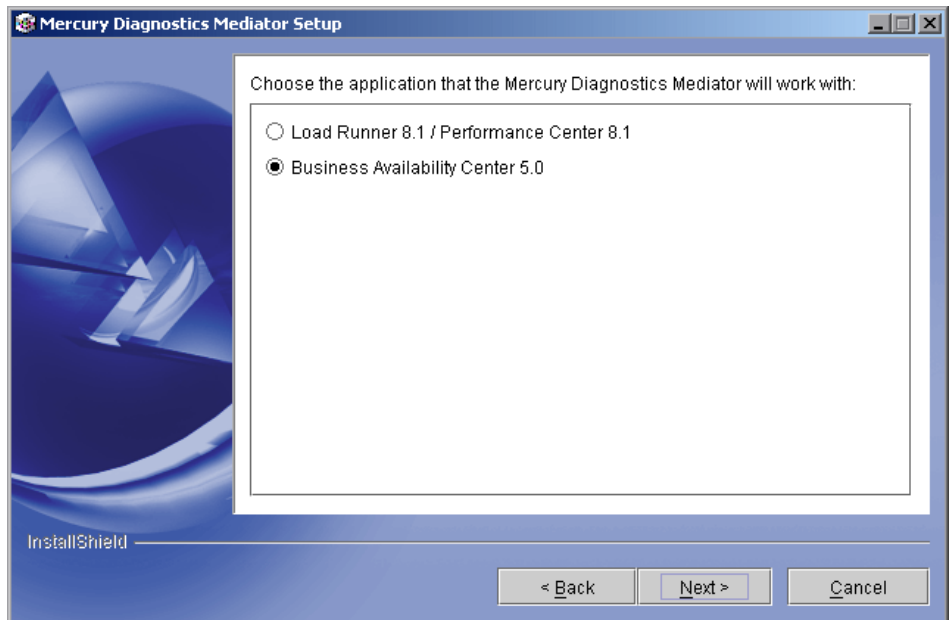
Click **Next** when you are ready to proceed with the installation.

---

**Note:** If the Commander's host name cannot be resolved the installer will display an error message. If the host name is resolved, the Logical LAN Identification window opens.

---

The installer displays the dialog that allows you to select the applications that the Mediator is being installed to work with.



Select **Performance Center 8.1 / LoadRunner 8.1**.

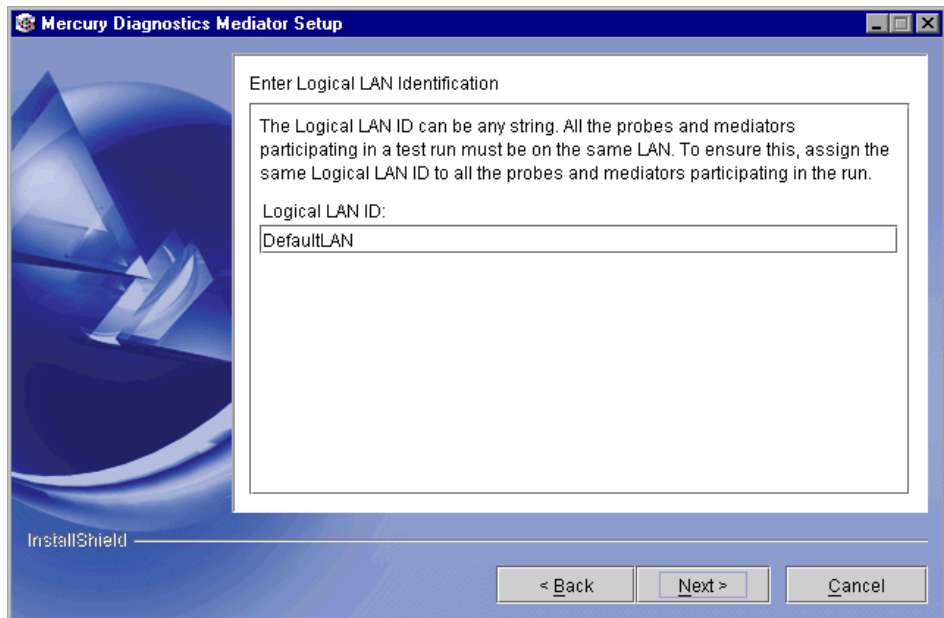
---

**Note:** This installation guide is written assuming that you are installing the Mediator for use with LoadRunner 8.1 or Performance Center 8.1. If you are installing the Mediator for use with the Mercury Business Availability Center 5.0, please see the installation guide for that product for further instructions on installing the Mediator.

---

Click **Next** to proceed with the installation.

**7** The installer displays the Logical LAN ID dialog.



The network traffic between the Probe and the Mediator is high-volume. For this reason, the Mediator and the Probes that communicate with it must be located on the same LAN. The Logical LAN ID is not a physical LAN ID. The value that you enter for the Mediator and each of the probes that you expect to be able to work with the Mediator must be exactly the same.

Enter the ID of the LAN on which the Probe and Mediator are running in to the **Logical LAN ID** box or accept the default.



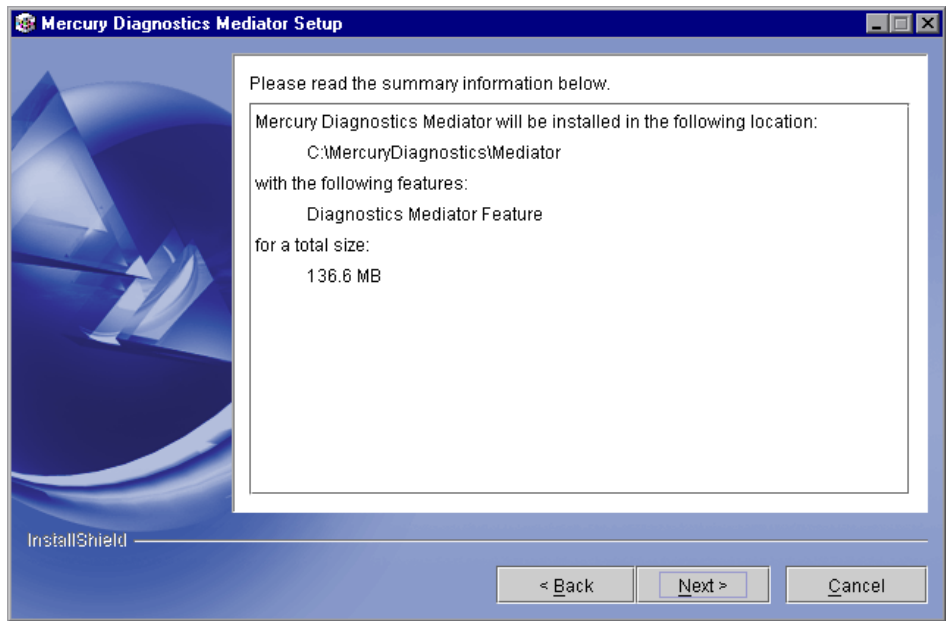
---

**Note:** The Logical LAN ID is case-sensitive.

---

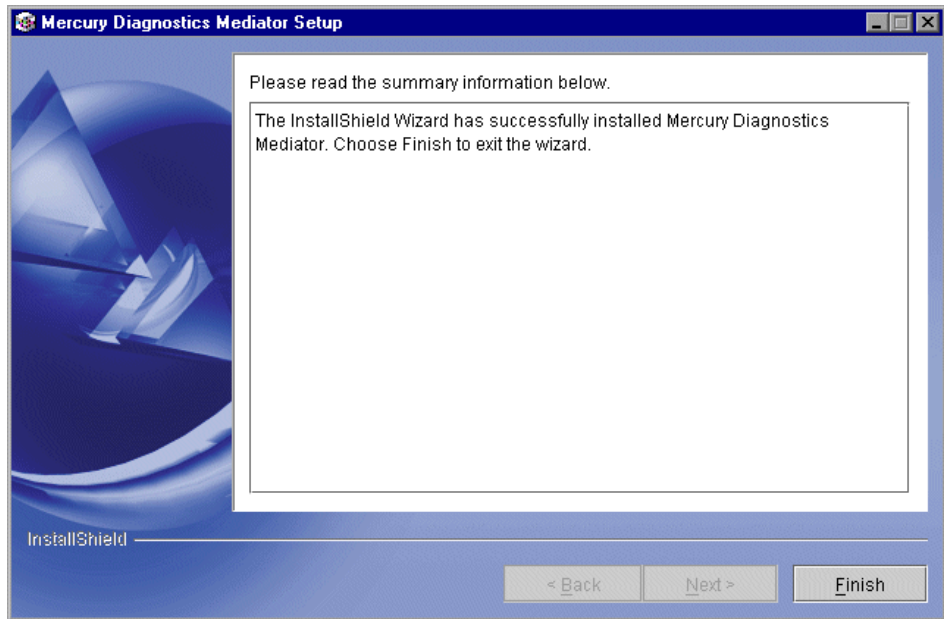
Click **Next** when you are ready to proceed with the installation.

- 8 The installer then displays a dialog with the Pre Installation summary information for your review. 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**.

- 9 The installer displays an installation status message to let you know that the installation was successfully completed.



Click **Finish** to end the installation.

When the installation is complete, the Mediator starts automatically and an icon appears on the task bar.

- 10 Verify that the Mediator is working properly using the System Health Monitor. See “Verifying the Mediator Installation” on page 75.
- 11 Start the Mediator. For details, see “Starting and Stopping Mediators” on page 73.

## Installing the Mediator on a UNIX Machine

The instructions in this section provide you with the information necessary to install the Mediator in most UNIX environments using either a graphics based installation or a console mode installation.

The installer screens that you will see in a graphics based installation are the same as those documented for the Windows installer in “Installing the Mediator on a Windows Machine” on page 58.

---

**Note:** For LoadRunner, the UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

For Performance Center, the UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

The installer is located in the **MercuryDiagnostics/Mediator** directory.

---

### To install a Mediator on a UNIX machine in console mode:

---

**Note:** The following instructions and screen shots are for a Mediator installation on a Solaris machine. These same instructions should apply for the other certified UNIX platforms.

---

- 1** Insert the Mercury Diagnostics for J2EE & .NET CD and locate the **MercuryDiagnostics/Mediator/** directory where the Mediator installer is located.
- 2** Copy the installer to the machine where the Mediator is to be installed.
- 3** Change the mode of the installer file to make it executable.
- 4** Execute the installer.

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

```
./install.sh -console
```

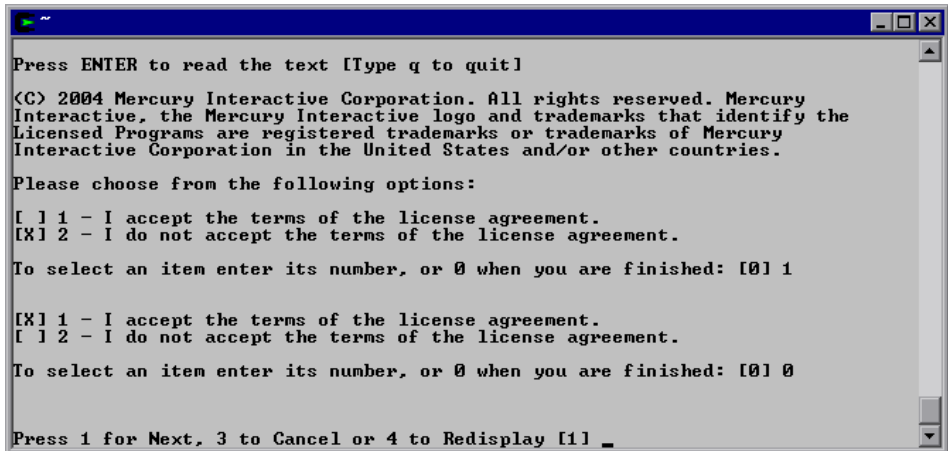
The installer will start and display the license agreement as shown in the following step.

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

```
./install.sh
```

The installer will display the same screens that are displayed for the Windows installer as shown in “Installing the Mediator on a Windows Machine” on page 58.

**5** The installer begins by displaying the software license agreement.



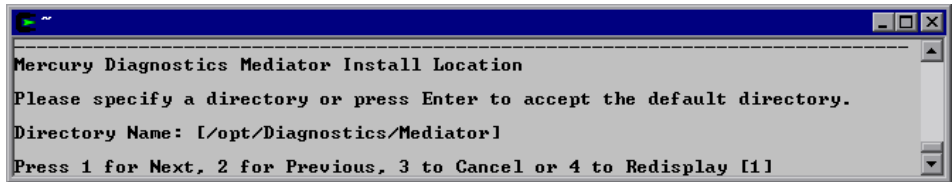
Read the agreement.

As you read you may press Enter to move to the next page of text or type **q** to jump to the end of the license agreement.

Enter **1** to accept the terms of the license agreement and then enter **0** to accept your selection.

Enter **1** to continue.

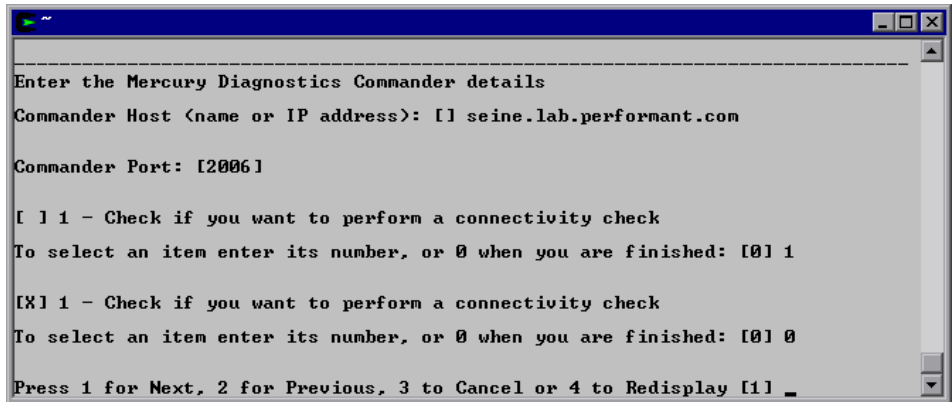
- 6** The installer prompts you to specify the path to the directory where the Mediator is to be installed.



Specify the directory where you want the Mediator to be installed or press Enter to accept the default directory.

Enter **1** to continue.

- 7** The installer asks you for information about the Commander that enables the communication between the Mediator and the Commander.



- Enter the host name or IP address of the machine on which the Commander is installed.

---

**Note:** 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.

For information about ensuring that the correct Mediator host name is used when there is a firewall or NAT in place or where your host machine is multi-homed see “Uninstalling the Mediator” on page 76.

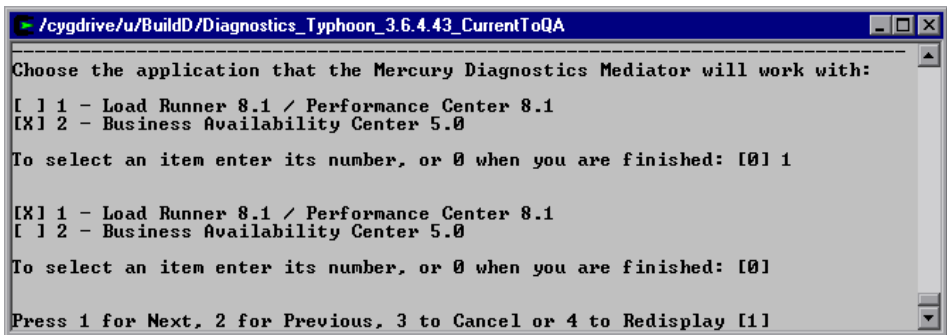
---

- ▶ Enter the Commander’s port number.

The default port number is 2006. If you specified a different port number when installing the Commander, specify the same port number here.

- ▶ Enter **1** to indicate that you want to perform a connectivity check to make sure that the Commander’s host name can be resolved. The connectivity check will let you know right away if you have made an error in the information that you provided about the Commander, or if there is a communication problem between the Commander’s host machine and the Mediator’s host machine.
- ▶ Enter **0** to accept the selection that you made and then enter **1** to continue.

- 8** The installer then prompts you to indicate the application with which the Mediator is being installed to work.



Enter the **1** to select LoadRunner 8.1 / Performance Center 8.1 and then enter **0** to confirm your choice.

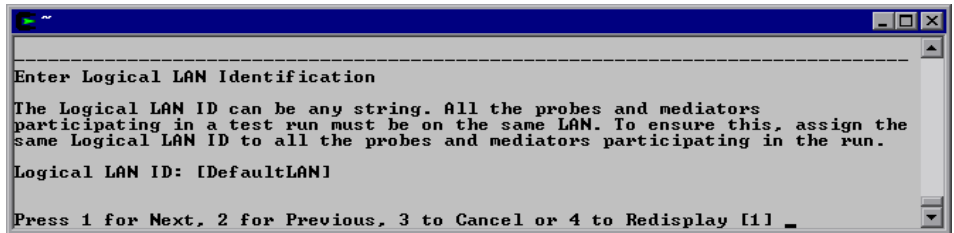
---

**Note:** This installation guide is written assuming that you are installing the Mediator for LoadRunner 8.1 or Performance Center 8.1. If you are installing this Mediator for use with the Mercury Business Availability Center 5.0, please see the installation guide for that product for further instructions on installing the Mediator.

---

Enter **1** to continue.

- 9 The installer then prompts you for the Logical LAN ID.



The network traffic between the Mediator and the Probes is high volume. For this reason the Mediator and the Probes that communicate with it must be located on the same LAN. The Logical LAN ID is not a physical LAN ID. The value that you enter for the Mediator and each of the Probes that you expect to be able to work with the Mediator must be exactly the same.

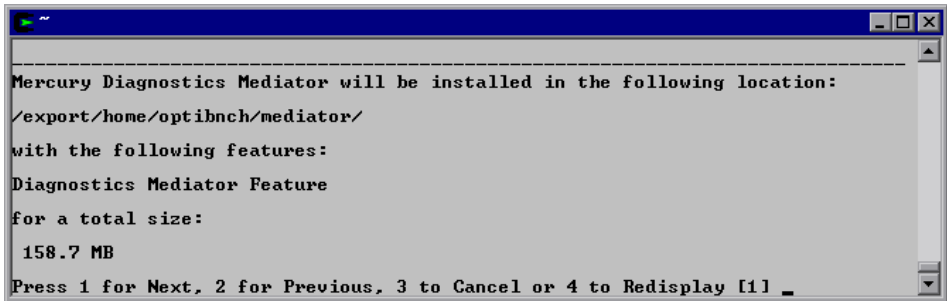
---

**Note:** The Logical LAN ID is case-sensitive.

---

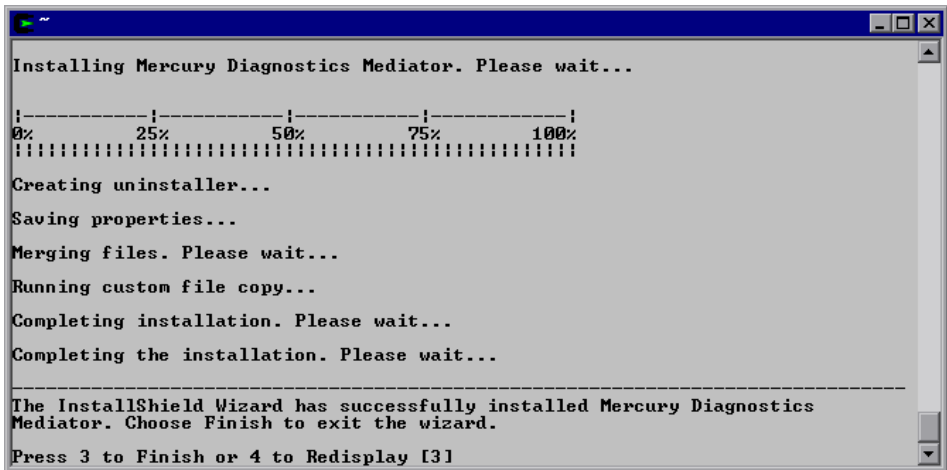
Enter the Logical LAN ID or accept the default. Enter **1** to continue.

- 10 The installer displays the Pre Installation Summary Information so that you can review the installation options that you have selected and make any changes before the component is actually installed.



Enter **1** to start the installation of the Mediator. If you would like to make changes to the options that you selected enter **2** to return the previous prompts.

- 11 The installer begins installing the Commander and displays a progress bar and status messages so that you can monitor the processing.



- 12 After installation is complete, type **3** to exit the procedure.
- 13 Start the Mediator. For details, see "Starting and Stopping Mediators" on page 73.



- 14 Verify that the Mediator is working properly using the System Health Monitor. See “Verifying the Mediator Installation” on page 75.

## Starting and Stopping Mediators

### Instructions for a Windows Machine

To start a Mediator on a Windows machine:

Choose **Start > Programs > Mercury Diagnostics Mediator > Start Mercury Diagnostics Mediator**.

To stop a Mediator on a Windows machine:

Choose **Start > Programs > Mercury Diagnostics Mediator > Stop Mercury Diagnostics Mediator**.

### Instructions for Solaris

To start a Mediator on a Solaris machine:

- 1 Ensure that the **M\_LROOT** environment variable is defined as the root directory of the Mediator.

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

```
$> setenv M_LROOT /opt/MercuryDiagnostics/Mediator299
```

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 Locate the **m\_daemon\_setup** executable in the **<Mediator\_Install\_Dir>/bin** directory.
- 3 Use **m\_daemon\_setup** with the “install” option as in the following example:

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

---

**Note:** If the Mediator is running on the same machine as the Commander, the Mediator may fail to start. To enable the Commander and the Mediator to run concurrently on the same machine:

- Stop the Commander.
- Copy the launch file from the Commander to the Mediator by entering the following:  
cp <commander\_install\_dir>\launch\_service\dat\nanny\commander.nanny <mediator\_install\_dir>\launch\_service\dat\nanny\.
- Ensure that the **M\_LROOT** environment variable is defined as the root directory of the *Mediator* and not the Commander.
- Start the Mediator. When you start the Mediator, the Commander will also start running.

---

**To stop a Mediator on a Solaris machine:**

- 1 Locate the **m\_daemon\_setup** executable in the **<Mediator\_Install\_Dir>/bin** directory.
- 2 Use **m\_daemon\_setup** with the **-remove** option as in the following example:

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

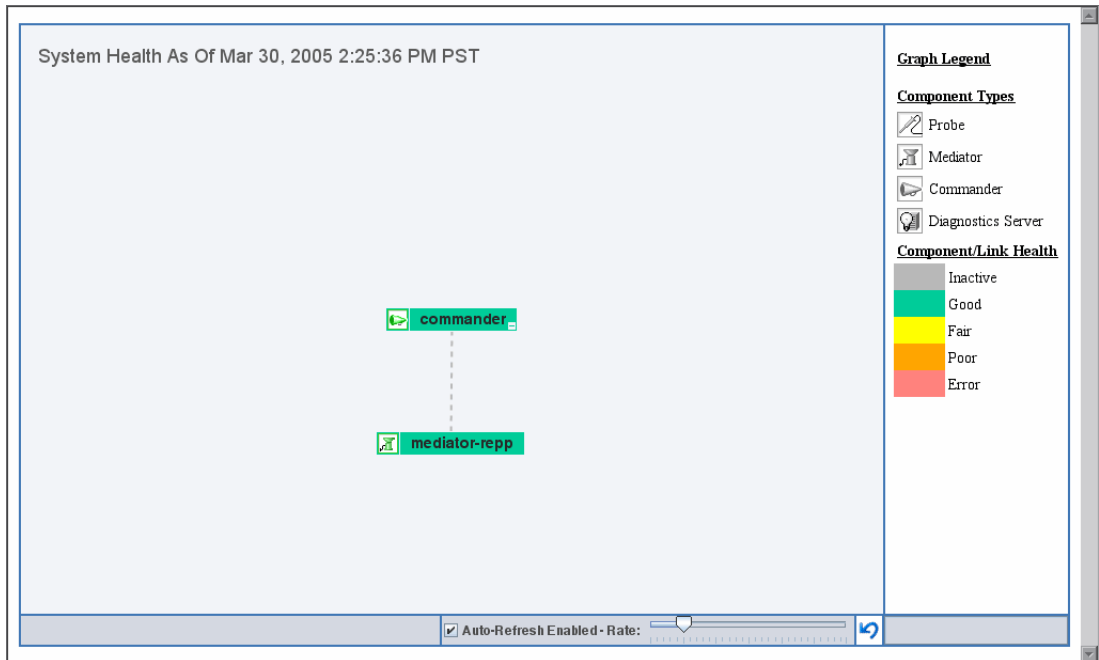
## Verifying the Mediator Installation

To verify that the Mediator has been installed correctly and that it has been able to establish connectivity with the other Diagnostics components, use the System Health Monitor. For instructions, see Appendix A, “Using the System Health Monitor.”

If you have been following the recommended installation sequence, after you have installed the Mediator you will be able to verify the following:

- The Mediator was successfully installed and it has successfully established connectivity to the Commander.

The new Mediator should be shown as a child of the Commander on the System Health Monitor as shown in the following screen image:



You should leave the System Health Monitor displayed in your browser. You will be using it to verify the progress of the component installation and to identify and troubleshoot any problems that you encounter as you proceed through the rest of the component installations.

## Troubleshooting Mediator Issues

To troubleshoot Mediator problems, use the System Health Monitor as explained in the preceding section, “Verifying the Mediator Installation.” For additional information see Appendix A, “Using the System Health Monitor.”

## Configuring the Mediator

The Mediator is installed with a default configuration that should enable it to work properly in most situations. You may encounter situations where changing the configuration of the Mediator could enable better Mediator performance or allow it to work in unusual situations.

For information on configuring the Mediator, see Appendix C, “Advanced Diagnostics Mediator Configuration.”

## Determining the Version of the Installed Mediator

When you are requesting support it is a good idea to know the version of the Diagnostics component that you have a question about.

**To determine the Mediator version:**

Locate the version file `<mediator_install_dir>\version.txt`. The file contains the four digit version number, as well as the build number.

## Uninstalling the Mediator

**To uninstall the Diagnostics Mediator from a Windows machine:**

- On a Windows machine execute `uninstall.exe` which is located in the `<mediator_install_dir>\_uninst` directory.
- You can also uninstall the Mediator from the Start menu by selecting:  
**Start > Programs > Mercury Diagnostics Mediator >  
Uninstall Mercury Diagnostics Mediator**

**To uninstall the Diagnostics Mediator from a Solaris machine:**

- 1 Locate the **uninstall\*** executable in the <mediator\_install\_dir>\_uninst directory.
- 2 Execute **uninstall\*** with the **-console** option as in the following example:

```
$>./uninstall -console
```

## Upgrading to a Newer Version of the Mediator

The following instructions will guide you in upgrading a Mediator to a newer version of the Mercury Diagnostics Mediator. Be sure to read the entire set of instructions and the sections immediately following before you begin the installation process. This will ensure that you understand the process and your options for upgrading the Mediators in your configuration.

---

**Note:** The Mediator installer does not upgrade existing installations of the Mediator. The instructions that follow are manual steps that you must take to make sure that the upgrade works as expected.

---

---

**Note:** The current version of the Mediator has been designed to work with the current versions of the Diagnostics components that are part of the LoadRunner / Performance Center 8.1 installation and will not work with earlier versions of these products. The previous versions of the Mediator will not work with the current versions of the above mentioned Mercury products.

---

**To upgrade the Mediator to a newer version:**

- 1** Shutdown the Probes that are associated with the existing Mediator.
- 2** Shutdown all the previous version of the Mediator. Be sure that you are logged into the host machine as the same user id that was used to start the Mediators.
- 3** Uninstall the previous version of the Mediator, making sure to do so as the same user id that installed them. See “Uninstalling the Mediator” on page 76.

Be sure to reboot your machine if the uninstaller instructs you to do so.

- 4** Uninstall the previous versions of the Probe from those machines on which you want to install the new Probe so that the applications performance data will be processed by the new Mediator. See “Upgrading to a Newer Version of the J2EE Probe” on page 137.
- 5** Install the new version of the Mediators and Probes as instructed in Chapter 2, “Preparing to Install Mercury Diagnostics for J2EE & .NET.”

**Phased Upgrade**

If you do not want to shut down all of the Probes at once you can use the same instructions provided above leaving the Probes that you do not want to shut down running. The Probes that are still running will report errors when the Mediator is shut down, but the application server should continue running normally.

**Running Two Versions of the Mediator Side-by-side**

You can run older and newer versions of the Mediator side-by-side on the same host machine while you work through your upgrade plan. To do this you must install the newer version of the Mediator in a different installation directory than the earlier version and you must make sure that the new Mediator is not configured to use the same port as the old mediator.

# 6

---

## Installing the Mercury Diagnostics Probe for .NET

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

The following topics are included in this chapter:

- About the Mercury Diagnostics Probe for .NET
- Installing the .NET Probe
- Verifying the .NET Probe Installation
- Configuring the .NET Probe

### About the Mercury Diagnostics Probe for .NET

The Mercury Diagnostics Probe for .NET (.NET Probe) is responsible for capturing events from a .NET application and sending the event metrics to the Mediator.

The .NET Probe is installed on the System Under Test (SUT). The .NET Probe uses runtime instrumentation to capture method latency information from specified applications. By default, the .NET Probe captures methods from the ASP and ADO tiers and MSMQ for ASP.NET applications. Custom business logic methods can be captured by creating a custom instrumentation specification file, known as a points file, for your application. The .NET Probe provides a low-overhead capture solution that works with Mercury's Application Diagnostics (AD) and Application Monitoring (AM) products.

.NET Probe initialization occurs only when an instrumented method is encountered for the first time. Initialization refers to the registration of the Probe with the Commander . An instrumented method is any method defined in any of the points files, for which the Probe was configured. 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.

For more information about configuring the .NET Probe see “Advanced .NET Probe Configuration” on page 385.

## Installing the .NET Probe

To install the .NET probe:

- 1 Insert the Installation CD into a CD-ROM drive.

For LoadRunner, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Windows Installation - Disc*.

For Performance Center, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Windows Installation - Disc*.

- 2 To execute the .NET Probe installer, click **Start > Run** and then type the location of your CD-ROM drive followed by the path to the setup program, **DotNetProbe\Setup\setup.msi**.

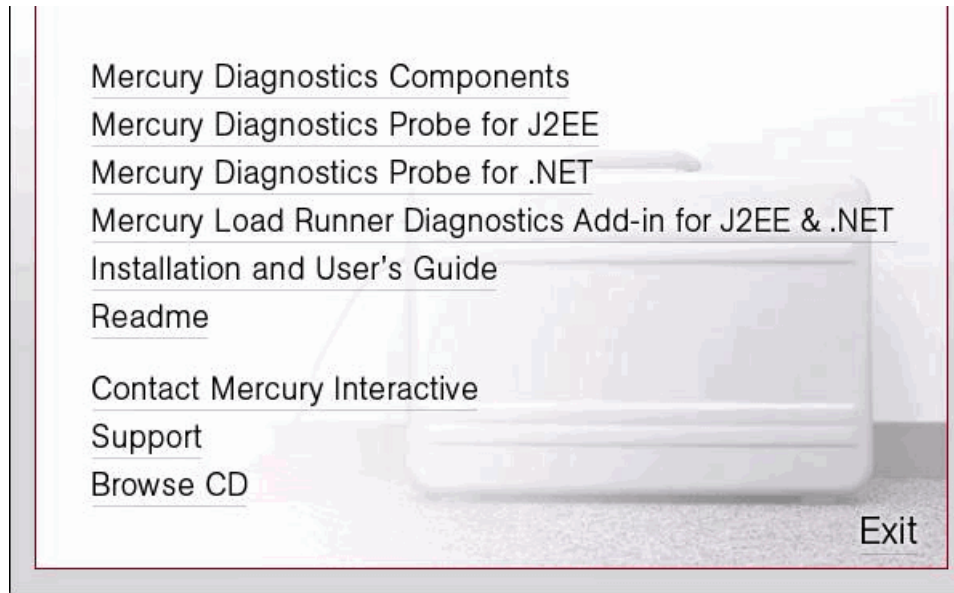
For example, if your CD-ROM drive letter is M, type:

```
m:\DotNetProbe\Setup\setup.msi
```

Alternatively, you can double click the **setup.msi** filename to execute the installer.

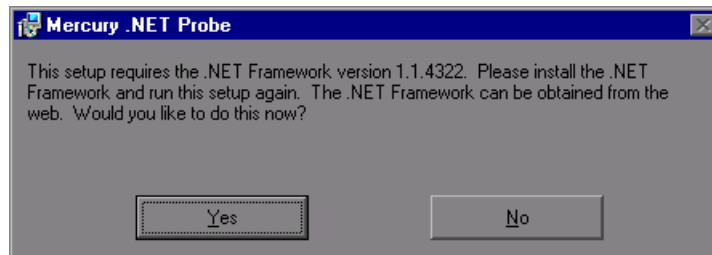


- 3 The installer displays the Mercury Diagnostics for J2EE & .NET main installation menu.



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

- 4 If the .NET Framework has not been installed on the host machine, the installer displays the following dialog box:

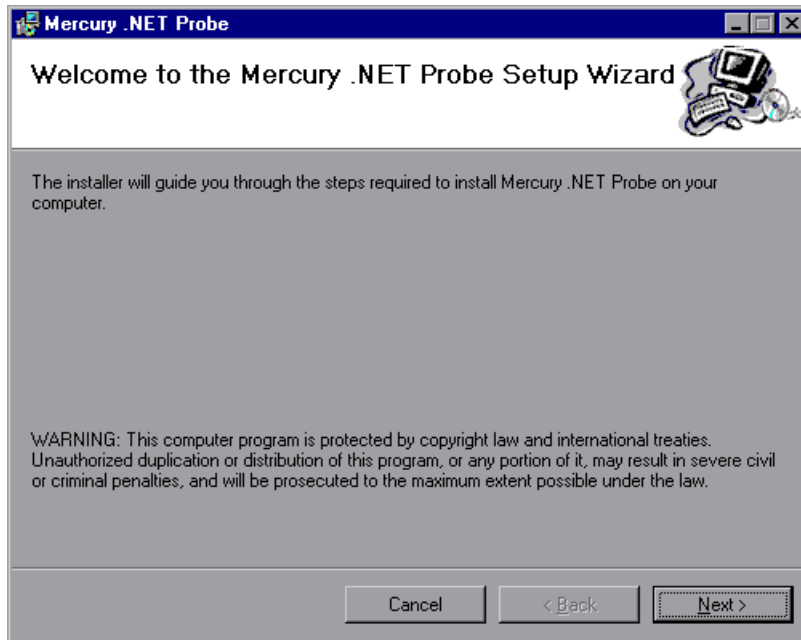


Click **Yes** to obtain the .NET Framework from the Web. Clicking **No** ends the installation.

- 5 If you click **Yes**, the installer opens Microsoft’s download page, where you may select the .NET Framework download.

After the .NET Framework installation restarts your computer, run the .NET Probe installation again.

- 6 The installer displays the Welcome dialog box to begin the installation of the .NET Probe.

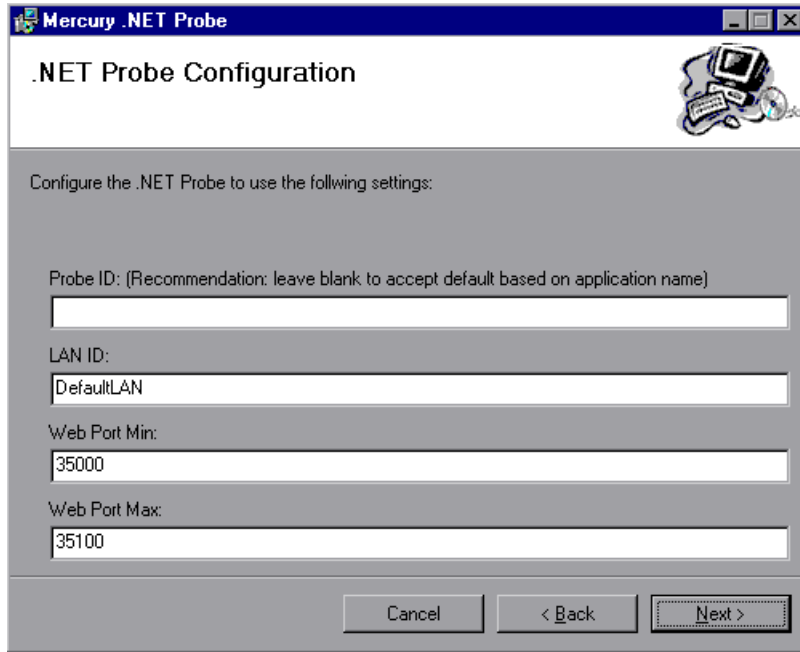


Click **Next** to proceed with the installation of the .NET Probe.

**7** The installer displays the License Agreement dialog.

Read the agreement. Select **I Agree**, to accept the agreement and click **Next** to proceed with the installation.

- 8 The installer displays the Mercury .NET Probe Configuration dialog box.



In the .NET Probe Configuration dialog box, enter the relevant configuration information as follows:

- **Probe ID:** The name that is used to identify the Probe within Diagnostics for J2EE & .NET. The .NET Probe will auto-generate a Probe ID based upon the application's appdomain name if you leave this field blank.

---

**Note:** It is recommended that you leave the Probe ID field blank and allow the Probe to auto-generate the Probe ID. Please read the following information carefully if you decide to enter your own Probe ID

---

### Considerations When Entering Your Own Probe ID:

- Only the following characters are valid in the Probe ID: letters, digits, dash, underscore and period.
- Create a Probe ID that will enable you to recognize the application that the Probe is monitoring and whether the Probe is a J2EE or .NET probe.

For example, the Probe ID for the first .NET Probe installed that will be monitoring an application named PetWorld could be:

PetWorld\_Dotnet\_Probe1

- When you specify a Probe ID, all of the Probes on the host machine will be forced to use the same Probe ID. If you want to override the default names, use the following substitution macros to enhance the ID at runtime.

\$(MACHINENAME): Machine's host name

\$(APPDOMAIN): Application's appdomain name

\$(PID): Application's process ID

The default Probe ID auto-generated by the Probe when the Probe ID 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”.

- ▶ **LAN ID:** The Logical LAN ID is not a physical LAN ID. The LAN ID value that you enter for the Mediator and each of the probes that you expect to be able to work with the Mediator must be exactly the same.

In the **LAN ID** box, enter the ID of the LAN on which the Probe and Mediator are running or accept the default.

---

**Note:** The Logical LAN ID is case-sensitive.

---

- ▶ **Web Port Min:** The lowest port number in a range of ports that the commander can use to communicate with its Probes.

- ▶ **Web Port Max:** The highest port number in a range of ports that the commander can use to communicate with its Probes.

---

**Note:** The default range is 100 ports, 35000 - 35100.

---

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 to listen for incoming requests from the Commander and Mediator.

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 another application have previously claimed them.

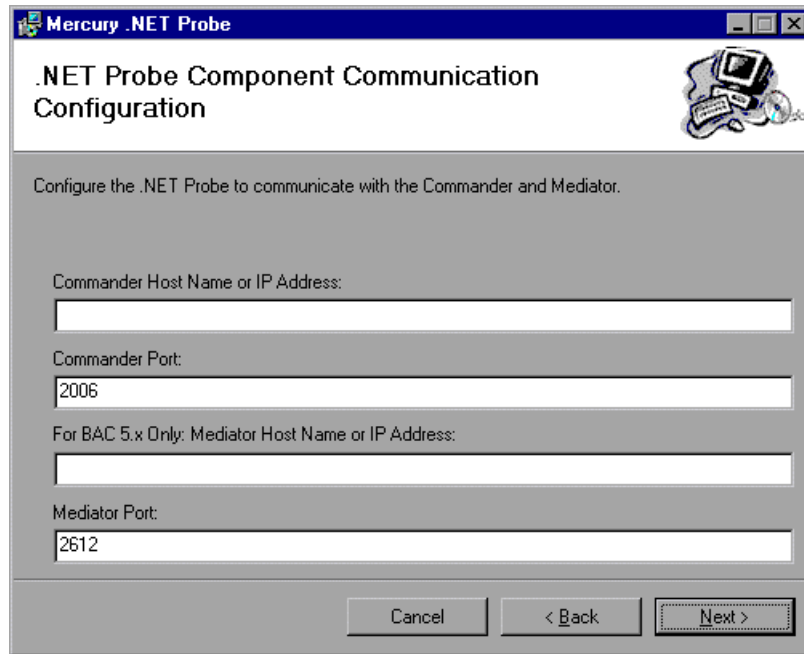
The port range needs to be large enough to, at a minimum, accommodate the maximum number of Probes that will be concurrently running on the machine.

**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 probes to account for ASP.NET's appdomain recycling
- Each port within the range does not have to be available as long as there are enough available ports within the range to satisfy the previous bullets requirements.
- 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** when you are ready to proceed with the installation.

- 9 The installer displays the .NET Probe Component Communication Configuration dialog.



The screenshot shows a Windows-style dialog box titled "Mercury .NET Probe" with a subtitle ".NET Probe Component Communication Configuration". The dialog contains the following fields and controls:

- Instruction: "Configure the .NET Probe to communicate with the Commander and Mediator."
- Field: "Commander Host Name or IP Address:" (empty text box)
- Field: "Commander Port:" (text box containing "2006")
- Field: "For BAC 5.x Only: Mediator Host Name or IP Address:" (empty text box)
- Field: "Mediator Port:" (text box containing "2612")
- Buttons: "Cancel", "< Back", and "Next >" (the "Next >" button is highlighted with a dashed border).

Enter the communication parameters that enable the .NET Probe to communicate with the Commander and the Mediator.

- **Commander Host Name or IP Address:**
- **Commander Port:** Enter the port number where the Commander is listening for communications from the Probe. The default port number is 2006. If you assigned a different port when the Commander was installed, make sure to enter the same port number here.

**Note:** When you enter the host name and port for a Mediator, the installer configures the .NET Probe so that it will work in the Application Monitoring mode with BAC 5.x. When no Mediator information is specified, the installer configures the .NET Probe so that it will work in the Application Diagnostics mode with Performance Center 8.1 / LoadRunner 8.1.

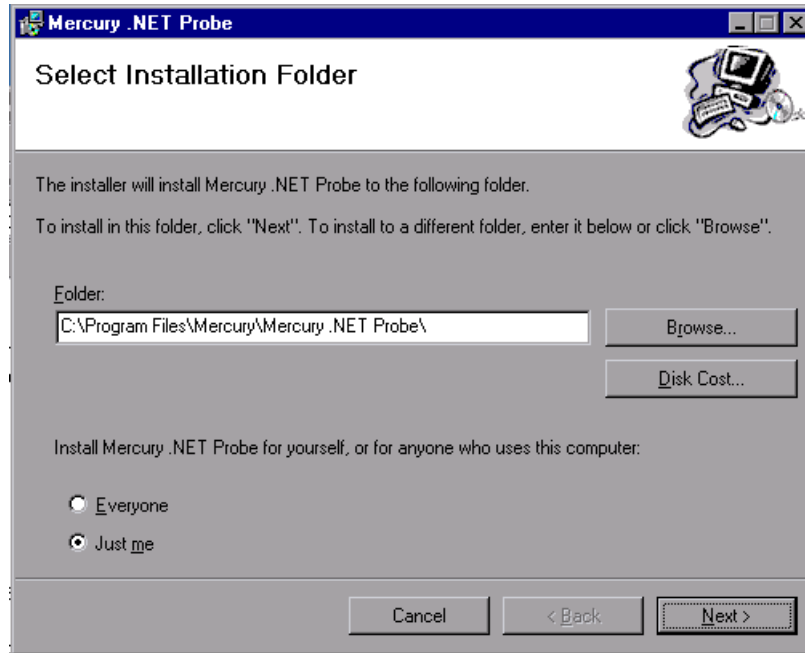
Only enter the Mediator host name and port if you want the Probe to be configured for Application Monitoring mode for BAC 5.x.

---

- **For BAC 5.x only: Mediator Host Name or IP Address:**
- **Mediator Port:** Enter the port number where the Mediator is listening for communications from the Probe. The default port number is 2612. If you assigned a different port when the Mediator was installed, make sure to enter the same port number here.

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

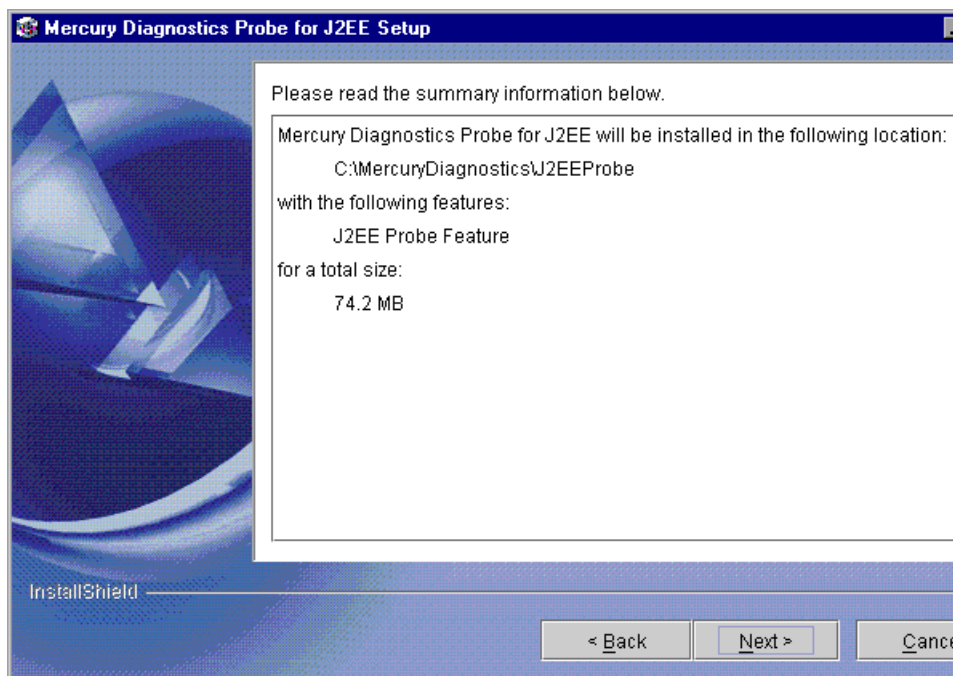


**10** The installer displays the Select Installation Folder dialog.

- ▶ Ensure that the location where you want the .NET Probe to be installed has been specified in the **Folder** text box either by typing in the path to the desired installation directory or by clicking **Browse** to navigate to the desired location.
- ▶ The **Disk Cost** button allows you to check the amount of available disk space on the drives on the host machine. Use this functionality to make sure that there is enough room for the Probe installation.
- ▶ Specify which users of the host machine will be able to access the .NET Probe.
  - Choosing **Everyone** causes the shortcuts for the Probe to appear on the Start Menu that everyone sees.
  - Choosing **Just Me** causes the shortcuts to be installed on the Start Menu for the particular user who is logged on when the Probe is installed.

Click **Next** when you are ready to proceed with the installation.

**11** The installer displays the Confirm Installation dialog.

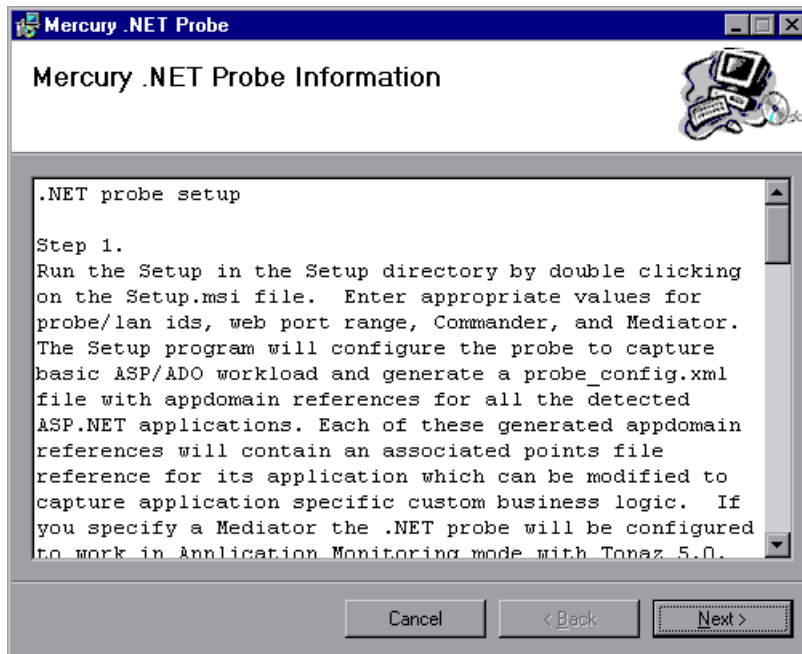


At this point, the installer has gathered all of the information that it needs to install the .NET Probe.

Click **Next** to start the .NET Probe installation.

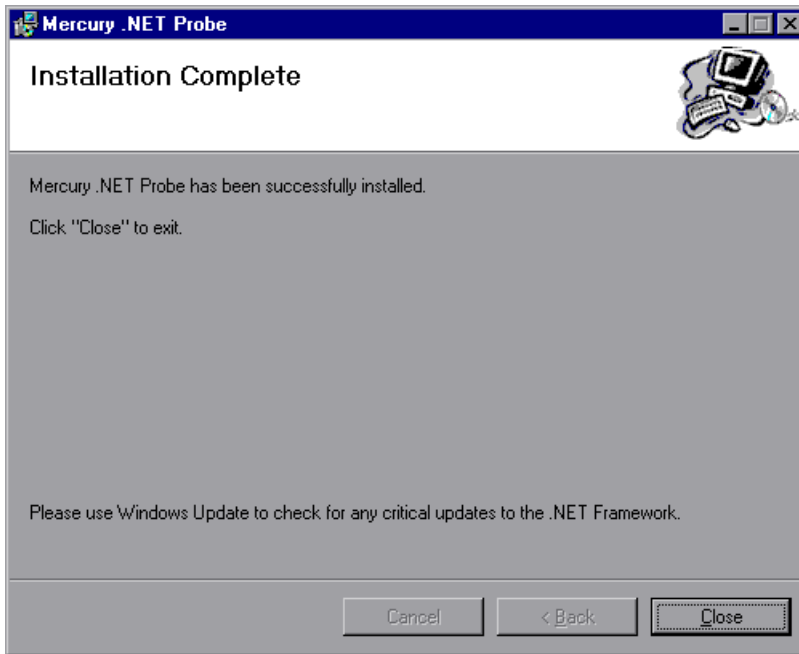
- 12 When the .NET Probe installation is complete, the installer displays a dialog that provides instructions for configuring the Probe so that it can be used to monitor your application. The installer also stores this same information in the readme.rtf file in the <probe\_install\_dir>.

The steps for configuring the .NET Probe are discussed in greater detail in “Advanced .NET Probe Configuration” on page 385



Click **Next** when you are ready to proceed.

- 13 The installer displays the final dialog in the process to confirm that the installation has been successful.



Click **Close** to exit the installer.

- 14 After you exit the installer, you must restart either IIS or the Web publishing service before you can use Mercury Diagnostics Probe for .NET with ASP.NET applications.
- 15 Verify that the .NET Probe was installed correctly as described in "Verifying the .NET Probe Installation," below.

## Verifying the .NET Probe Installation

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 A, "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 Mediator was successfully installed and is communicating with the Commander.
- ▶ The .NET Probe was successfully installed is communicating with the Commander.

---

**Note:** The .NET Probe will not be displayed in System Health when first installed because the Probe does not register with the Commander until it is started. The Probe is started and is registered with the Commander 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.

---

## Configuring the .NET Probe

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

For details on configuring the .NET Probe see Appendix E, “Advanced .NET Probe Configuration.”



# Part III

---

## Installation and Configuration of Diagnostics Probe for J2EE





# 7

---

## Introducing J2EE Probe Installation and Application Server Configuration

This chapter introduces the processes for installing and configuring the J2EE Probe and for configuring the application server on which your J2EE applications run. The installation and configuration processes are presented in detail in separate chapters.

### About Installing and Configuring the J2EE Probe

Before your application can be monitored by the J2EE Probe, you must install the Probe and configure it so that it will work in your environment.

#### **Install the J2EE Probe**

The Probe is installed on the machine that hosts the application that you wish to monitor. For instructions for installing the Probe on the supported platforms see “Installing the Mercury Diagnostics Probe for J2EE” on page 99.

#### **Configure the Probe**

The default configuration of the probe has been set to effectively monitor your application with very little impact on your application. Certain situations may require changes to these default settings. See, “Advanced J2EE Probe and Application Server Configuration” on page 369 for instructions for configuring the application server and the J2EE Probe for these advanced situations.

## About Configuring the Application Server

Before your application can be monitored by the J2EE Probe, you must configure your application server and the way that it is instrumented.

### Instrument the JRE

When you install a Probe, you must run the JRE Instrumenter to prepare your application for the instrumentation that allows the Probe to monitor the application's processing. The JRE Instrumenter runs automatically during the Probe installation unless you elect to skip it at that time. If you elect to not run the JRE Instrumenter during the installation of the Probe, you may run it manually as described in "Running the JRE Instrumenter" on page 142.

### Modify the Application Startup Script

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. There are two ways to configure the application servers:

- ▶ Using the automated tool, Mercury Configuration Utility. For more information see "Configuring the Application Server and Probe Using the Configuration Utility," on page 147.
- ▶ By manually updating the application server startup scripts. For more information see "Configuring the Application Servers Manually," on page 161.

---

**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 Instances" on page 196.

---

# 8

---

## 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 for using a generic installer are also provided for installing the J2EE Probe on other platforms.

This chapter contains the following sections:

- ▶ Installing the J2EE Probe on a Windows Machine
- ▶ Installing the J2EE Probe on a UNIX Machine
- ▶ Installing the J2EE Probe on a z/OS Mainframe
- ▶ Installing the J2EE Probe Using the Generic Installer
- ▶ Verifying the J2EE Probe Installation
- ▶ Using the J2EE Probe with Deep Diagnostics
- ▶ Overriding the Default Probe Host Machine Name
- ▶ Determining the Version of the J2EE Probe That is Installed
- ▶ Upgrading to a Newer Version of the J2EE Probe
- ▶ Uninstalling the J2EE Probe

---

**Note:** You must configure the Probe and the application server before you can use the Probe to monitor your application. See the instructions in Chapter 7, “Introducing J2EE Probe Installation and Application Server Configuration.”

---

## Installing the J2EE Probe on a Windows Machine

To install a J2EE Probe on a Windows machine:

- 1 Insert the installation disc into a CD-ROM drive.

---

**Note:** For LoadRunner, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Windows Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

For Performance Center, the installer for Windows can be found on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Windows Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

---

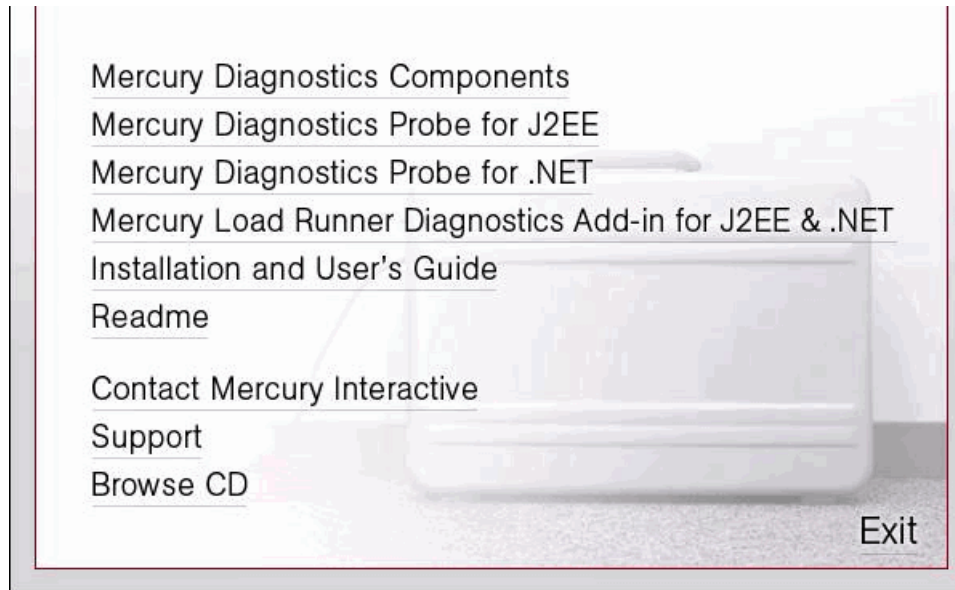
- 2 The installer should start automatically.

If the installer does not start, run the installer from the Windows Start menu. Click **Start > Run** and then type the location of your CD-ROM drive followed by the name of the installer program, **setup.exe**.

For example, if your CD-ROM drive letter is M, type:

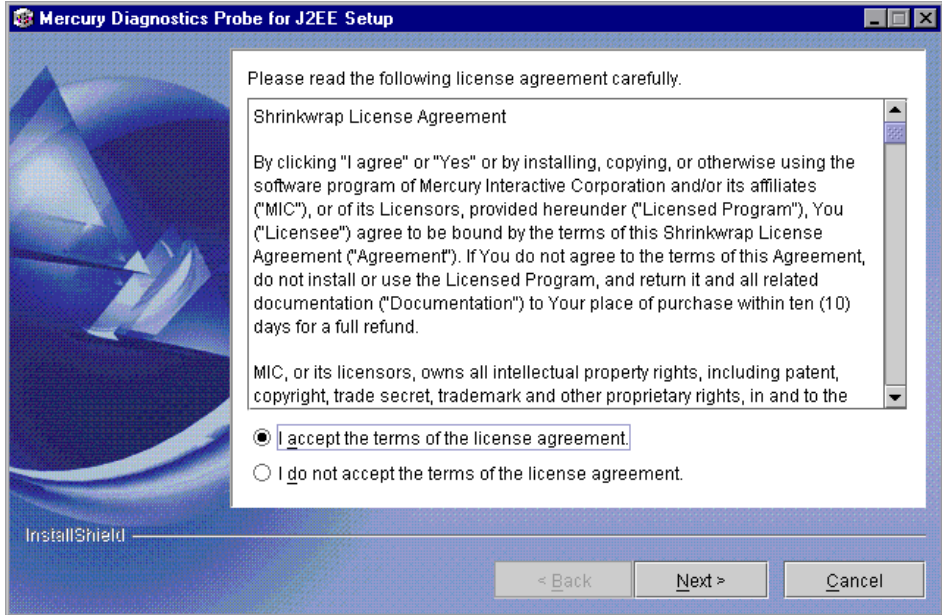
```
m:\setup.exe
```

- 3 The installer displays the Mercury Diagnostics for J2EE & .NET main installation menu.



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

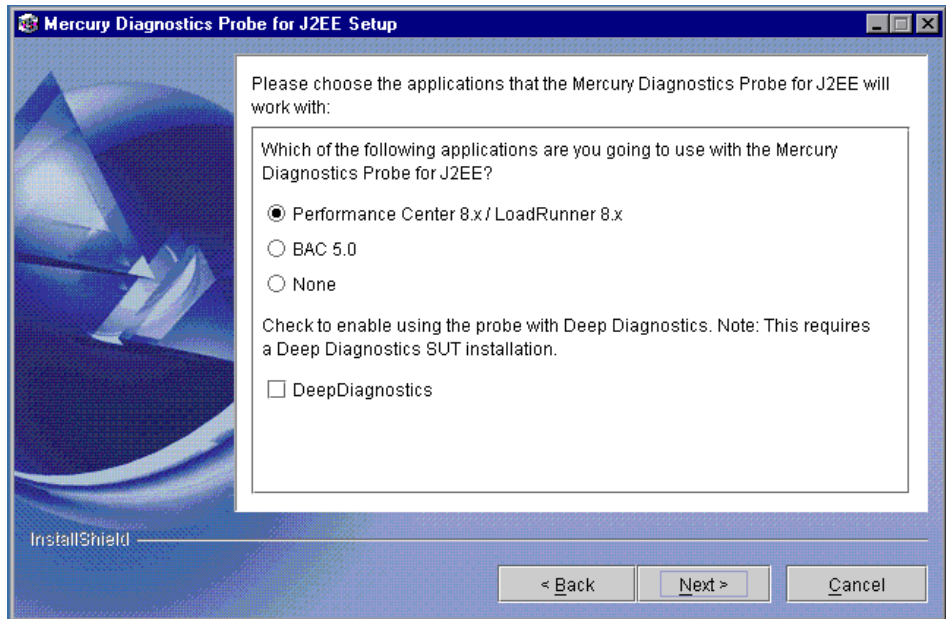
4 The installer displays the software license agreement.



Read the agreement. Select **I accept the terms of the license agreement** to accept the agreement.

Click **Next** to proceed with the installation.

- 5 The installer displays the dialog that allows you to select the Mercury Products with which the J2EE Probe will work.



Select **Performance Center 8.x / LoadRunner 8.x** and indicate whether you intend to use the Probe to collect data for Deep Diagnostics.

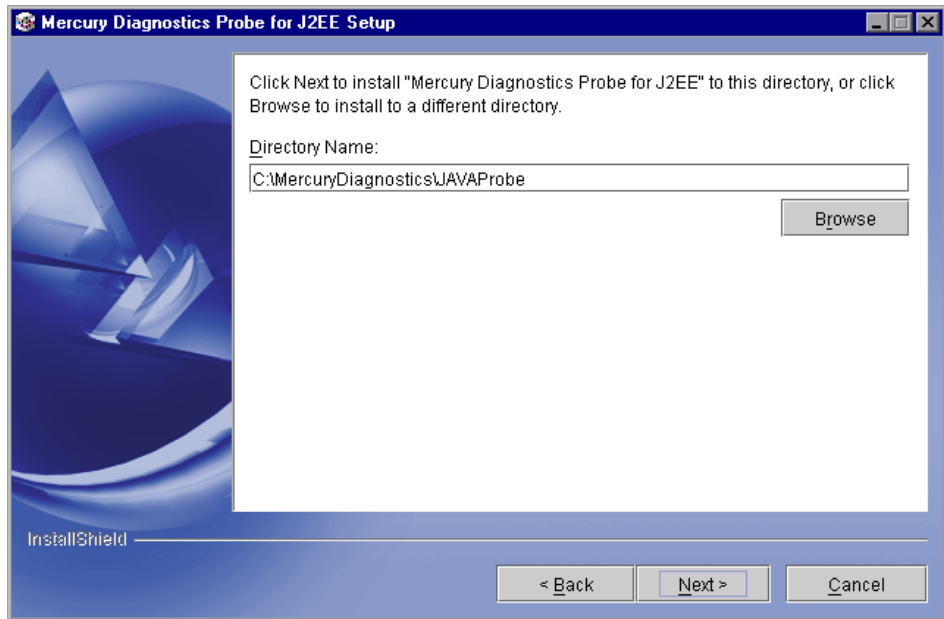
---

**Note:** This installation guide is written assuming that you are installing the J2EE Probe for use with LoadRunner 8.1 or Performance Center 8.1. If you are installing the J2EE Probe for use with the Mercury Business Availability Center 5.0, please see the installation guide for that product for further instructions on installing the Probe.

---

Click **Next** to proceed with the installation.

- 6 The installer displays the dialog for you to specify the path to the directory in which the Probe is to be installed.

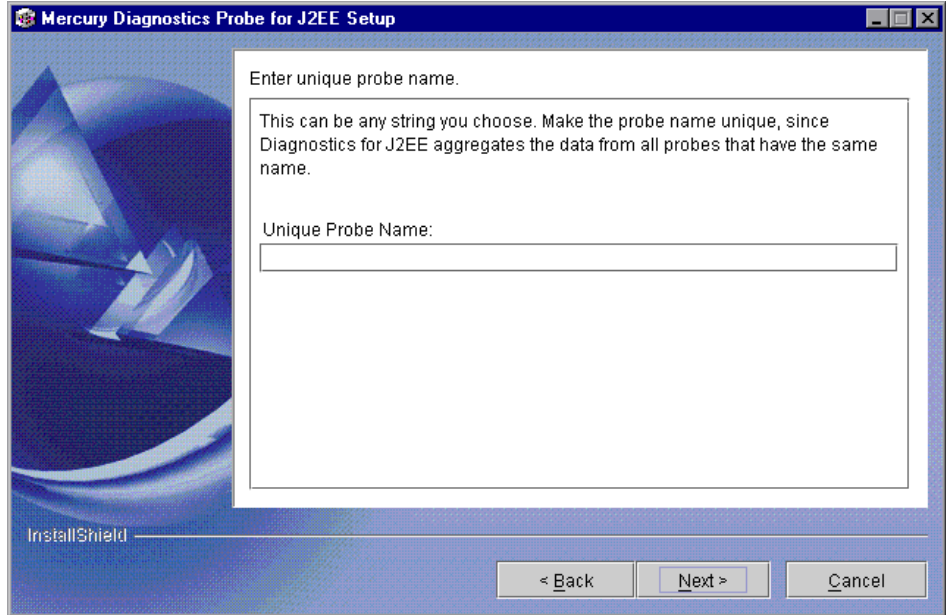


Ensure that the location where you want the J2EE Probe to be installed has been specified in the Directory Name text box either by typing in the path to the desired installation directory or by clicking Browse to navigate to the desired location.

Click **Next** when you are ready to proceed with the installation.



- 7 The installer displays the dialog for you to specify the Unique Probe Name.



Enter the name that is to be used to identify the Probe within Diagnostics for J2EE & .NET. The Probe Name can contain only the following valid characters: letters, digits, dash, underscore.

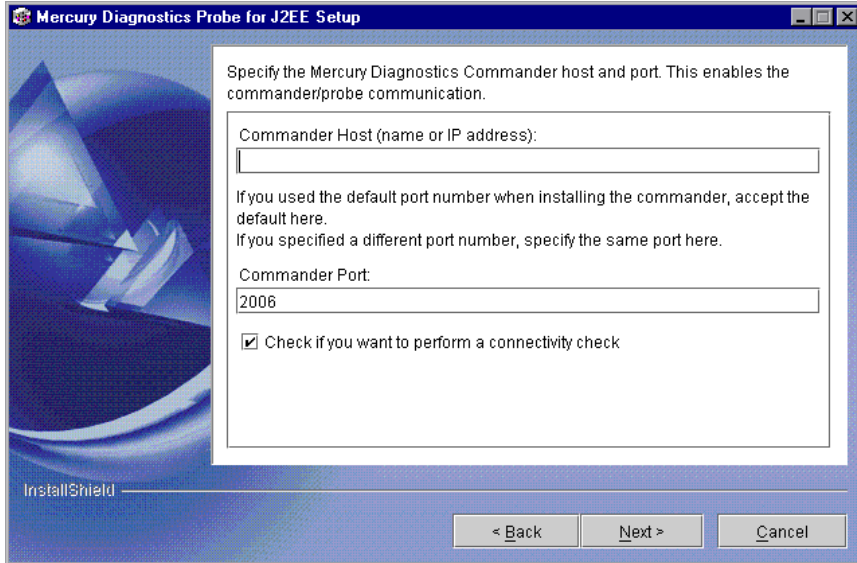
You should assign a Probe Name that will enable you to recognize the application that the Probe is monitoring and the type of probe (J2EE or .NET).

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

PetWorldJ2EEProbe1

Click **Next** when you are ready to proceed with the installation.

8 The installer displays the Diagnostics Commander Details dialog box.



- Enter the host name or IP address of the machine on which the Commander is installed.

---

**Note:** 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.

For information about ensuring that the correct Probe host name is used when there is a firewall or NAT in place or where your host machine is multi-homed see “Overriding the Default Probe Host Machine Name” on page 136.

---

- Enter the Commander’s port number. The default port number is 2006. If you specified a different port number when installing the Commander, specify the same port number here.

- Indicate whether you want to perform a connectivity check to make sure that the Commander's host name can be resolved. The connectivity check will let you know right away if you have made an error in the information that you provided about the Commander, or if there is a communication problem between the Commander's host machine and the Probe's host machine.

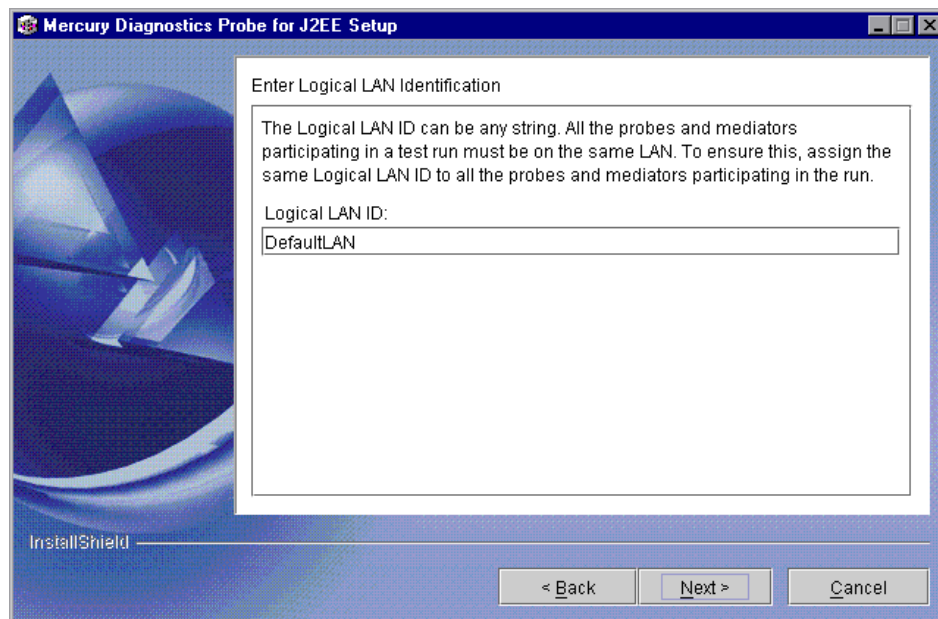
Click **Next** when you are ready to proceed with the installation.

---

**Note:** If the Commander's host name cannot be resolved, the installer will display an error message. If the host name is resolved, the Logical LAN Identification window opens.

---

- 9 The installer displays the Logical LAN ID dialog.



The network traffic between the Probe and the Mediator is high-volume. For this reason, the Mediator and the Probes that communicate with it must be located on the same LAN. The Logical LAN ID is not a physical LAN ID. The

value that you enter for the Mediator and each of the probes that you expect to be able to work with the Mediator must be exactly the same.

Enter the ID of the LAN on which the Probe and Mediator are running in to the **Logical LAN ID** box or accept the default.

---

**Note:** The Logical LAN ID is case-sensitive.

---

Click **Next** when you are ready to proceed with the installation.

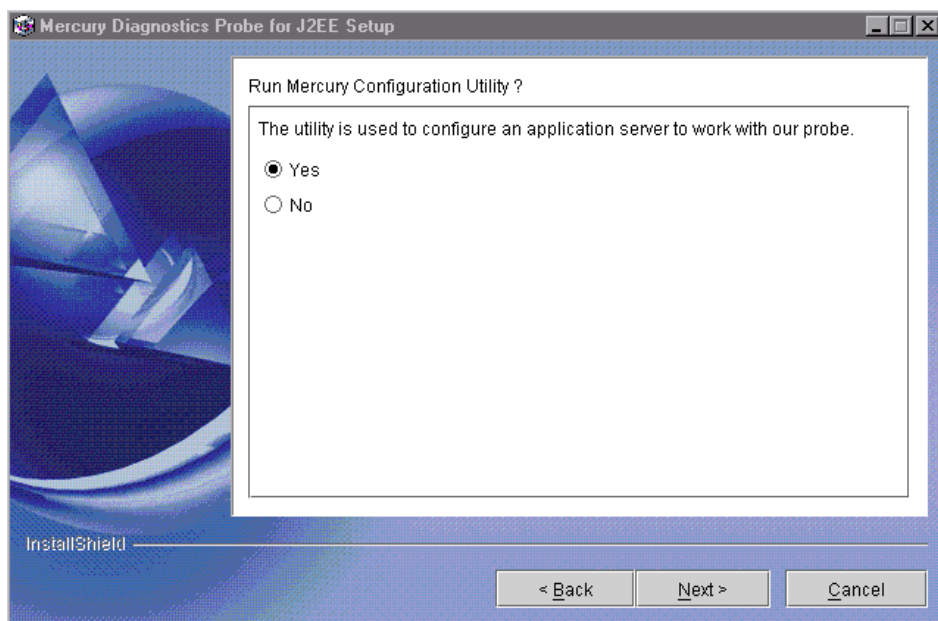
- 10** The installer displays a dialog for you to indicate if you want to configure an application server during the installation of the Probe.

---

**Note:** You can let the installer configure the application server now, or you can do it after the Probe has been installed using the process described in Part III, “Installation and Configuration of Diagnostics Probe for J2EE.”

If you are installing this Probe to work with SAP NetWeaver, select **No** on this dialog. You will configure SAP NetWeaver manually. See “Configuring the SAP NetWeaver Application Server” on page 192.

---

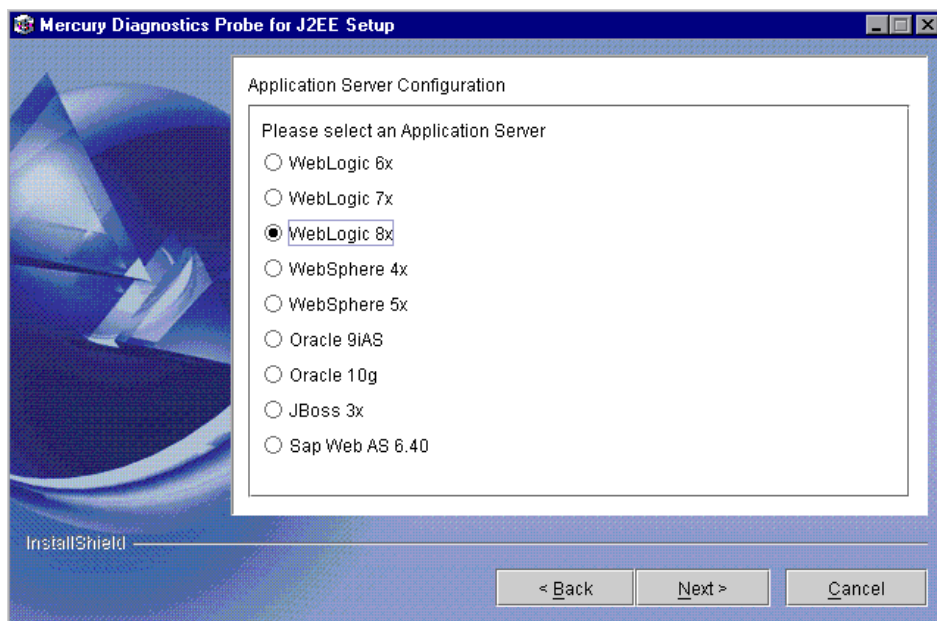


To request that the installer configure the application server for the Probe, choose **Yes**. The installer will continue as shown in step 11.

To skip the configuration of the application server for this probe choose **No**. If you chose to skip the application server configuration, the installer will continue as shown in step 13 on page 112.

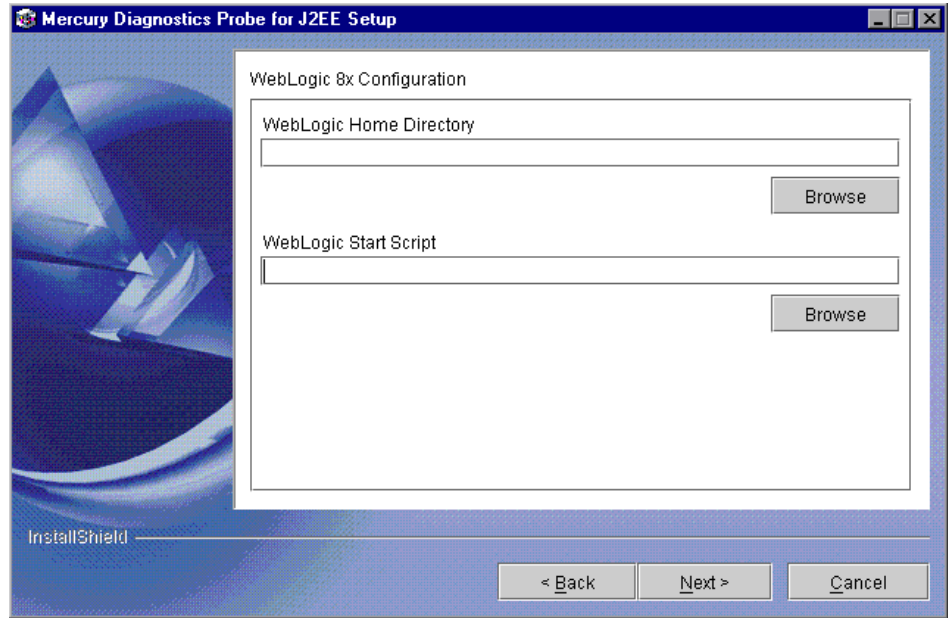
Click **Next** to proceed with the installation.

- 11 If you chose to configure the application server in step 10, the installer displays the Application Server Configuration window with a list of the available application servers.



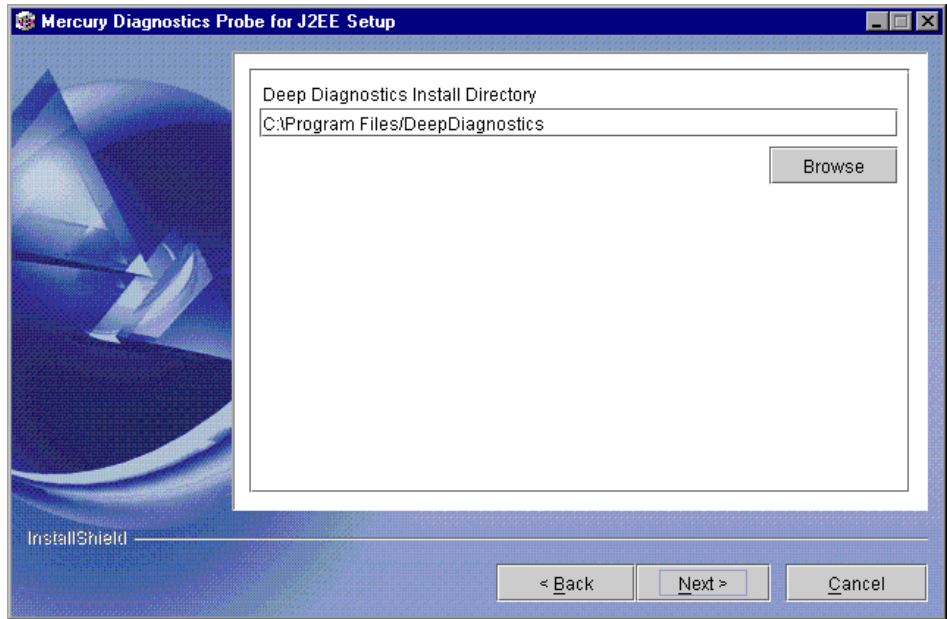
Choose the application server that you want to configure, and click **Next**.

- 12 The installer displays a dialog box that requests the application server configuration information for the application server that you selected on the previous dialog. The fields in this dialog will vary according to the application server that you selected.



Enter the requested information and click **Next**.

- 13 If you indicated that you will be using this Probe with Deep Diagnostics (see step 5), the installer asks you to enter the Deep Diagnostics installation directory.



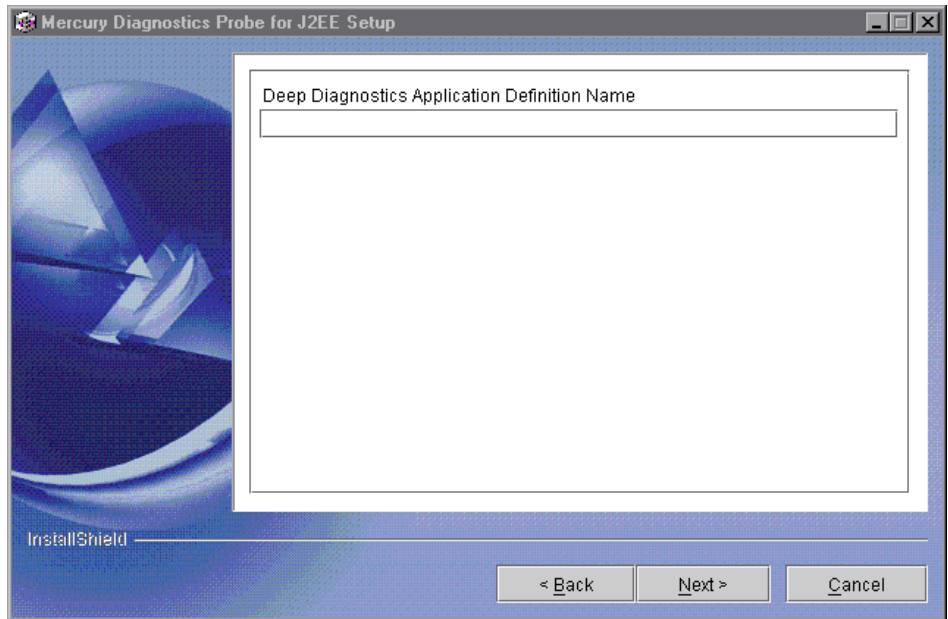
Enter the location where Deep Diagnostics was installed into the **Deep Diagnostics Install Directory** text box or click **Browse** to navigate to the location.

Click **Next** when you are ready to proceed with the installation.

- 14 In the dialog box that follows, enter the Deep Diagnostics Application Definition Name. The Deep Diagnostics Application Definition name must exactly match the name specified for the Application Definition in Deep

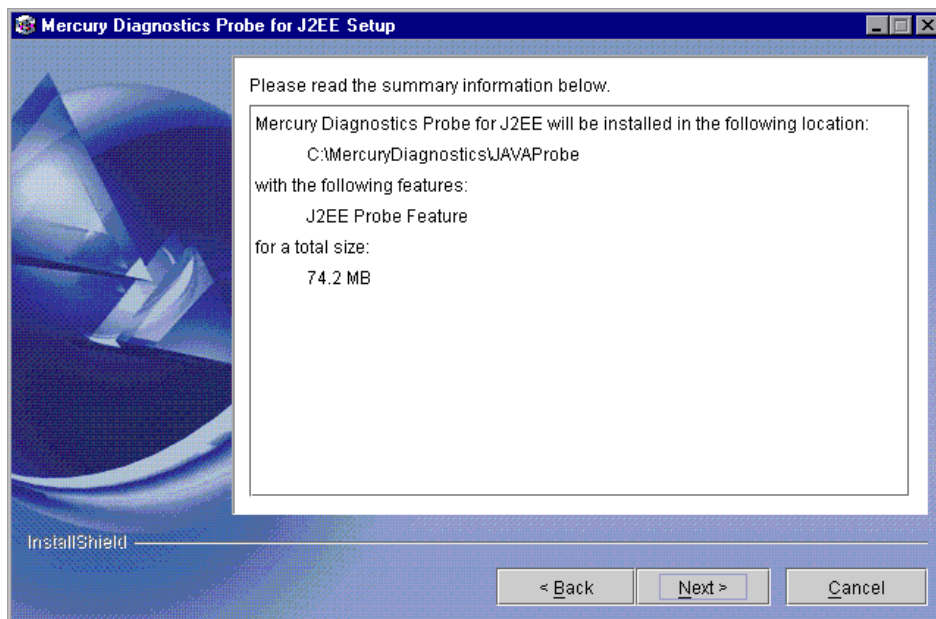


Diagnostics. See the Mercury Deep Diagnostics for J2EE Installation Guide for more information. Click **Next** to proceed with the installation.



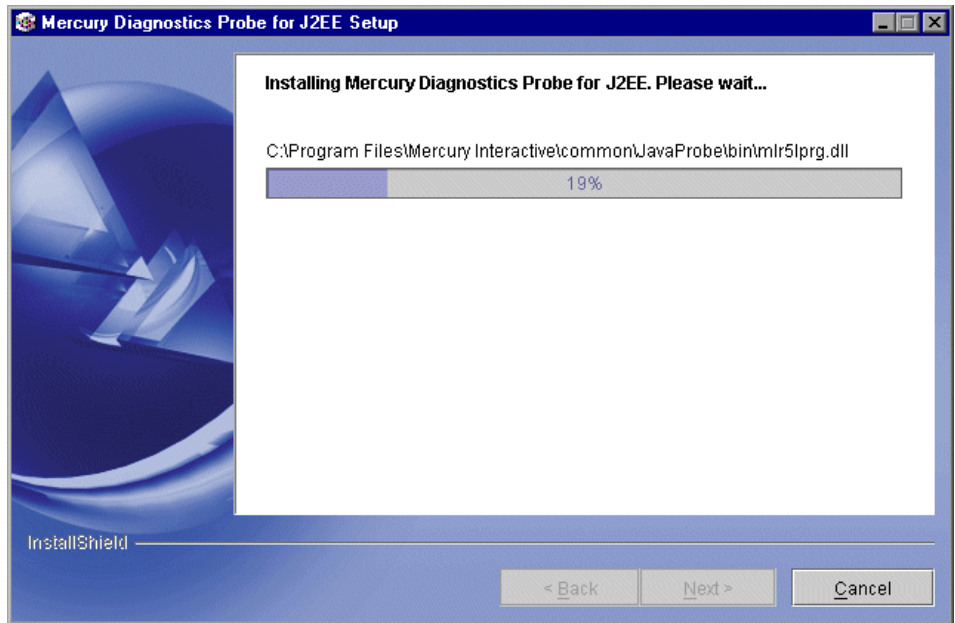
15

- 16** The installer then displays a dialog with the Pre Installation Summary Information for your review.

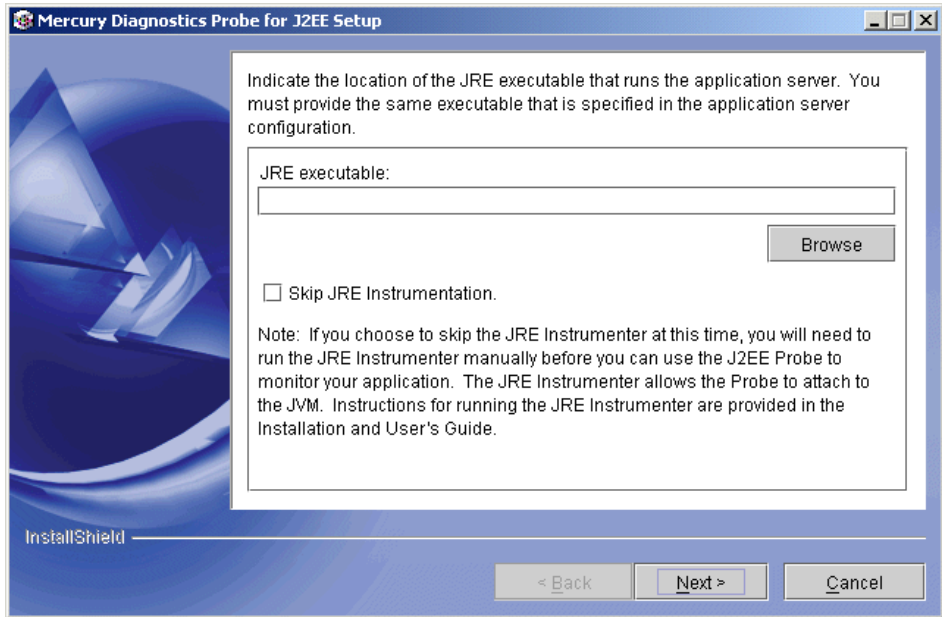


Review the information to make sure that you are satisfied. Click **Back** to make any changes, or click **Next** to proceed with the installation.

- 17 The installer begins the process of installing the J2EE Probe. The installer displays a progress bar to let you know how the installation is proceeding.



- 18** Following the progress bar dialog, the installer displays the JDK/JRE dialog box.



Enter the path to the JVM executable (java.exe) used by the application server that you are monitoring. For example, if you have installed WebLogic 6.1 in your **D:\bea** directory, the **java.exe** file can be found in **D:\bea\jdk131\jre\bin**:

To skip this step and perform it later, select **Skip JRE Instrumentation**.

Click **Next** to proceed with the installation.

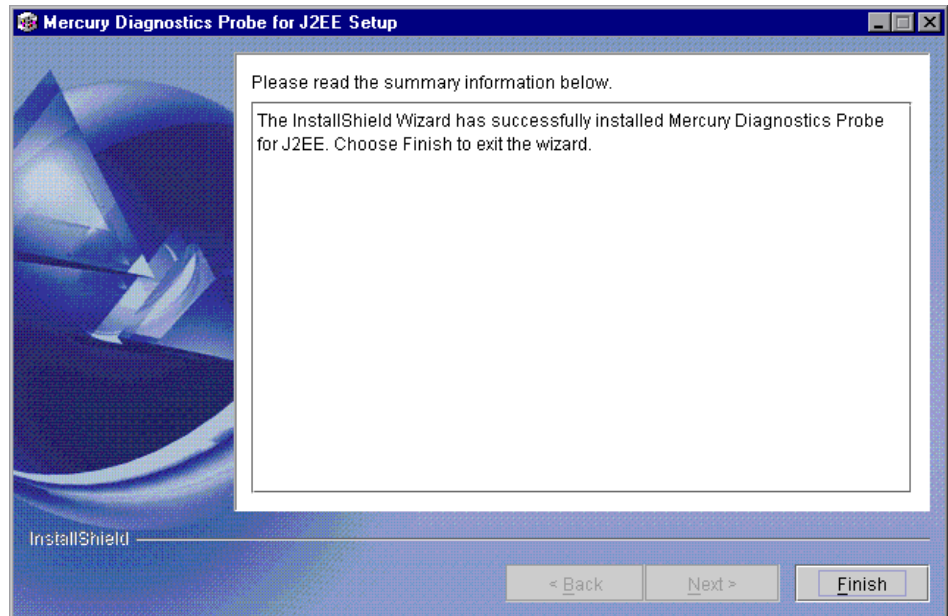
---

**Note:** If you are installing this Probe to work with SAP NetWeaver, select **Skip JRE Instrumentation**. You will configure SAP NetWeaver manually. See “Configuring the SAP NetWeaver Application Server” on page 192.

If you choose to skip this step now, you must run the JRE Instrumenter manually before you can use the Probe. For instructions, see “Running the JRE Instrumenter” on page 141.

---

- 19** The installer displays an installation status message to let you know that the installation was successfully completed.



Click **Finish** to end the installation.

- 20** Configure your application server so it can use the J2EE Probe. For more information, see “Introducing J2EE Probe Installation and Application Server Configuration” on page 97.
- 21** If you will be using Mercury Deep Diagnostics for J2EE, configure the J2EE Probe to collect data for Deep Diagnostics. For more information, see “Configuring the J2EE Probe and Application Server for Deep Diagnostics” on page 375.

## 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 information that helps you to install the J2EE Probe in most UNIX environments using either a graphics based installation or a console mode installation.

The installer screens that you will see in a graphics based installation are the same as those documented for the Windows installer in “Installing the J2EE Probe on a Windows Machine” on page 100.

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 133.

---

**Note:** The J2EE\_Probe directory includes folders for AIX, Generic, HP-UX, LINUX, and Solaris. Choose the appropriate installer for your environment and copy it to the UNIX machine that will host the Probe..

---

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

**To install the J2EE Probe on a UNIX machine:**

- 1** Insert the installation disc and locate the **J2EE\_Probe/<UNIX version>** directory. For example, for a Solaris install you would locate the directory **J2EE\_Probe/Solaris**.

---

**Note:** For LoadRunner, the UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury LoadRunner 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

For Performance Center, the UNIX installers are on the CD labeled *Mercury Diagnostics for J2EE & .NET 3.6 - Supporting Mercury Performance Center 8.1 - Unix Installation - Disc* that you received with your Mercury Diagnostics for J2EE & .NET package.

---

- 2** Copy the installer that is appropriate for your environment to the machine where the Probe will be installed.
- 3** Change the mode of the installer file to make it executable.
- 4** Execute the installer.
  - ▶ To run the installer in console mode enter the following at the UNIX command prompt:

```
./install.sh -console
```

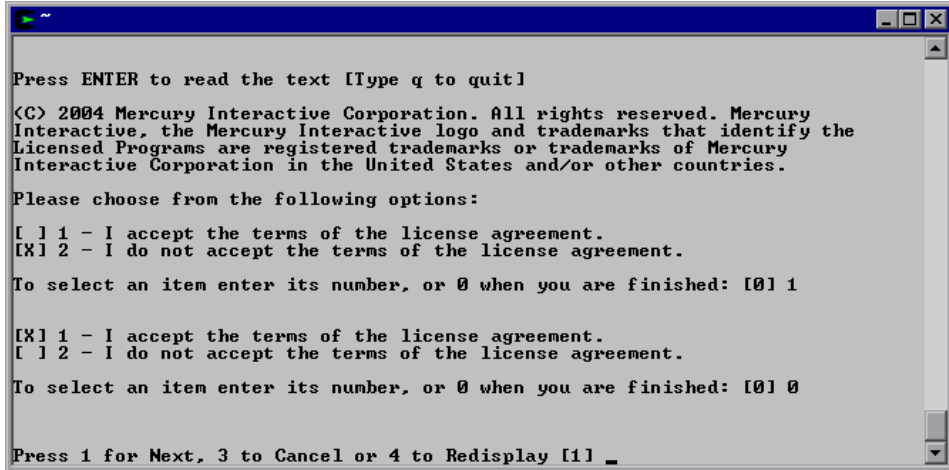
The installer will start and display the license agreement as shown in the following step.

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

```
./install.sh
```

The installer will display the same screens that are displayed for the Windows installer as shown in “Installing the J2EE Probe on a Windows Machine” on page 100.

5 The installer begins by displaying the software license agreement.



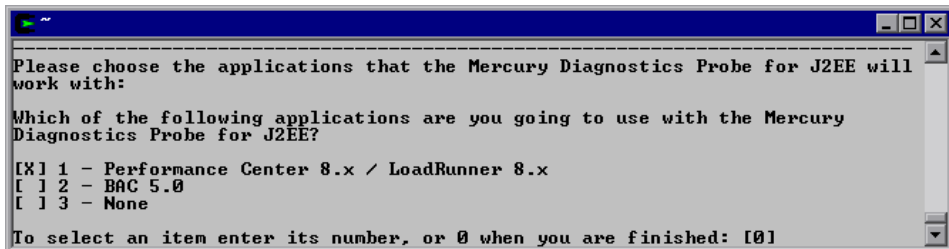
Read the agreement.

Press Enter to move to the next page of text or type **q** to jump to the end of the license agreement.

Enter **1** to accept the terms of the license agreement and then enter **0** to accept your selection.

Enter **1** to continue.

6 The installer prompts you to indicate the application with which the J2EE Probe is going to work.



Select **Performance Center 8.x / LoadRunner 8.x** by entering **1**. When you are satisfied with your selection enter **0** to proceed with the installation.

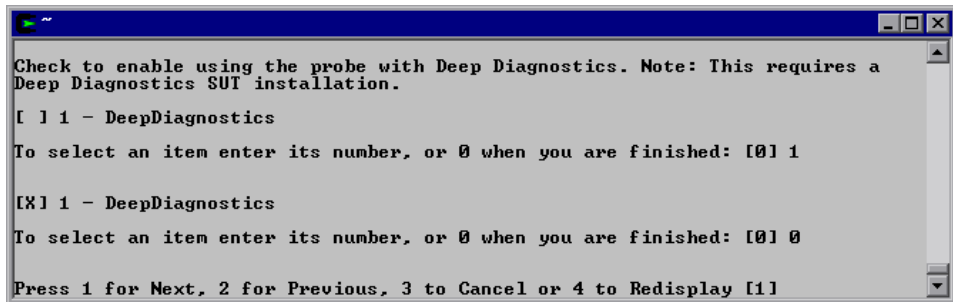


---

**Note:** This installation guide is written assuming that you are installing the J2EE Probe for use with LoadRunner 8.1 or Performance Center 8.1. If you are installing the J2EE Probe for use with the Mercury Business Availability Center 5.0, please see the installation guide for that product for further instructions on installing the Probe.

---

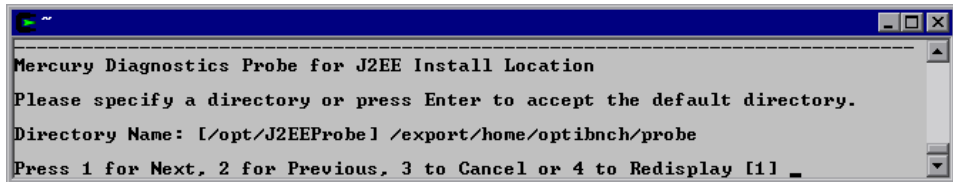
- 7** The installer asks if you want to use the J2EE Probe with Deep Diagnostics.



Enter **1** to select the option to use the Probe with Deep Diagnostics and then enter **0** to confirm your choice.

Enter **1** to continue.

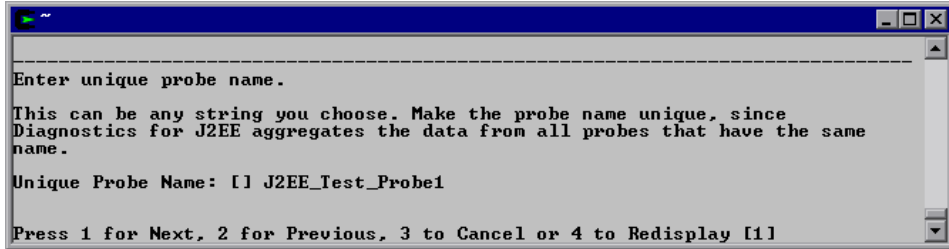
- 8** The installer prompts you to specify the path to the directory where the J2EE Probe is to be installed.



Specify the directory where you want the J2EE Probe to be installed or press Enter to accept the default directory.

Enter **1** to continue.

9 The installer asks for a unique probe name.



Enter the name that will be used to identify the Probe within Diagnostics for J2EE & .NET.

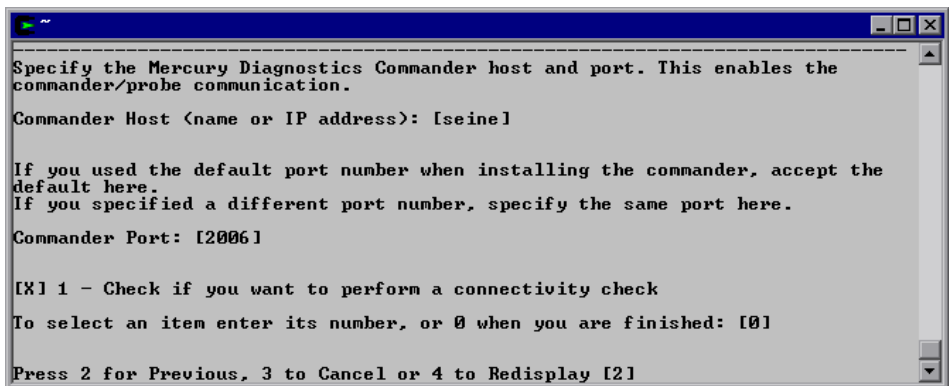
You should assign a Probe Name that will enable you to recognize the application that the Probe is monitoring and the type of probe (J2EE or .NET).

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

PetWorldJ2EEProbe1

Enter 1 to continue.

10 The installer asks for information about the machine that hosts the Commander.



- Enter the host name or IP address of the machine on which the Commander is installed.

**Note:** 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.

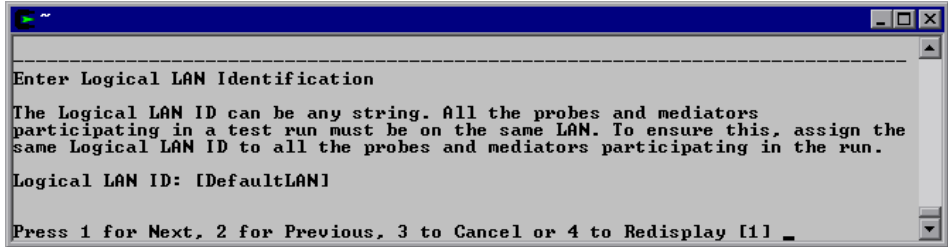
For information about ensuring that the correct Probe host name is used when there is a firewall or NAT in place or where your host machine is multi-homed see “Overriding the Default Probe Host Machine Name” on page 136.

---

- ▶ Enter the Commander’s port number.  
The default port number is 2006. If you specified a different port number when installing the Commander, specify the same port number here.
- ▶ Enter **1** to select the option to perform a connectivity check to make sure that the Commander’s host name can be resolved. The connectivity check will let you know right away if you have made an error in the information that you provided about the Commander, or if there is a communication problem between the Commander’s host machine and the Probe’s host machine. This selection will toggle off if you enter **1** again.
- ▶ Enter **0** when you are satisfied with your choice.

Enter **1** to continue.

**11** The installer asks for the Logical LAN ID.



The network traffic between the Probe and the Mediator is high-volume. For this reason, the Mediator and the Probes that communicate with it must be located on the same LAN. The Logical LAN ID is not a physical LAN ID. The value that you enter for the Mediator and each of the probes that you expect to be able to work with the Mediator must be exactly the same.

Enter the ID of the LAN on which the Probe and Mediator are running or accept the default.

---

**Note:** The Logical LAN ID is case-sensitive.

---

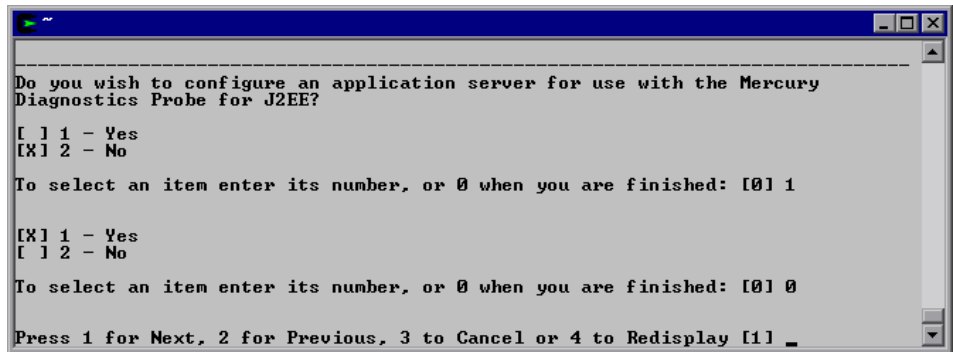
Enter **1** to continue.

**12** Indicate if you would like the installer to configure your application server.

---

**Note:** You can let the installer configure the application server now or you can do it after the Probe has been installed using the process described in Chapter 10, “Configuring the Application Server and Probe Using the Configuration Utility.”

---



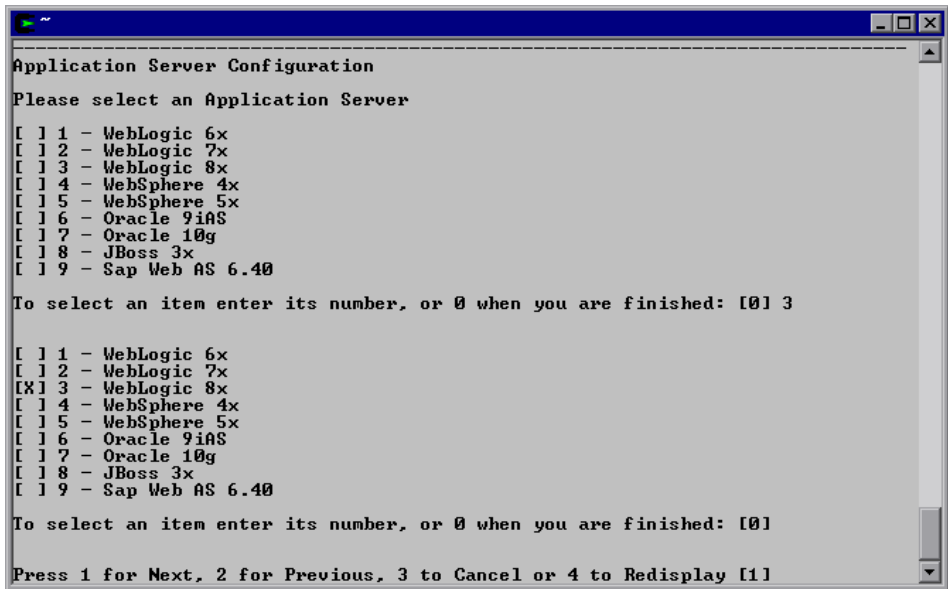
To request that the installer configure the application server for the Probe, choose **Yes** by entering **1**. The installer will continue as shown in step 11.

To skip the configuration of the application server for this probe choose **No** by entering **2**. If you chose to skip the application server configuration, the installer will continue as shown in step 13 on page 112.

Enter **0** when you are satisfied with your choice.

Enter **1** to continue.

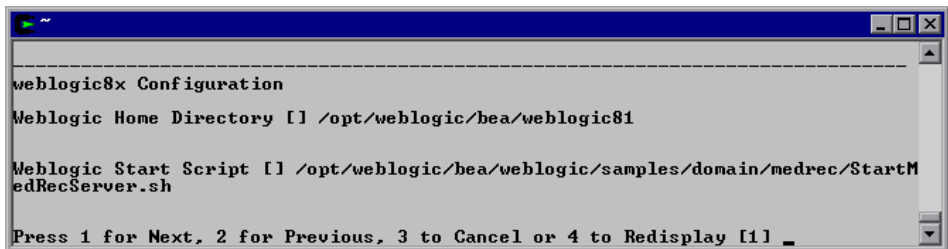
- 13 If you chose to configure the application server in step 12, the installer asks you to select the application server that you want to configure.



To make your selection, enter the number that corresponds to the application server to be configured. When you are satisfied with your choice, enter 0 to accept.

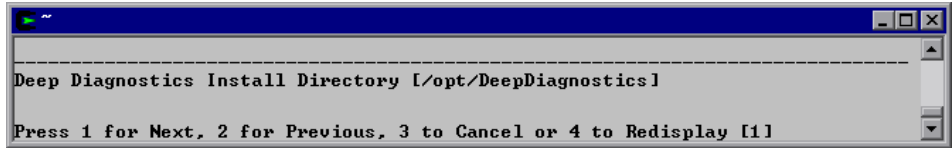
Enter 1 to continue.

- 14 The installer prompts you for information needed to configure the application server that you selected on the previous dialog. The fields in this dialog will vary according to the application server that you selected.



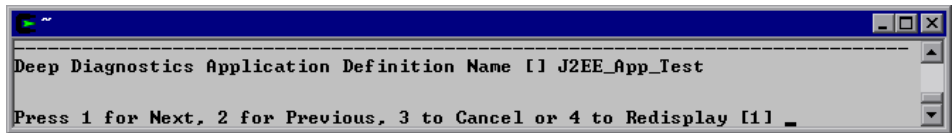
Enter the information requested and enter 1 to continue.

- 15** If you indicated that you will be using this Probe with Deep Diagnostics in step 7 on page 121, the installer asks you to enter the Deep Diagnostics installation directory.



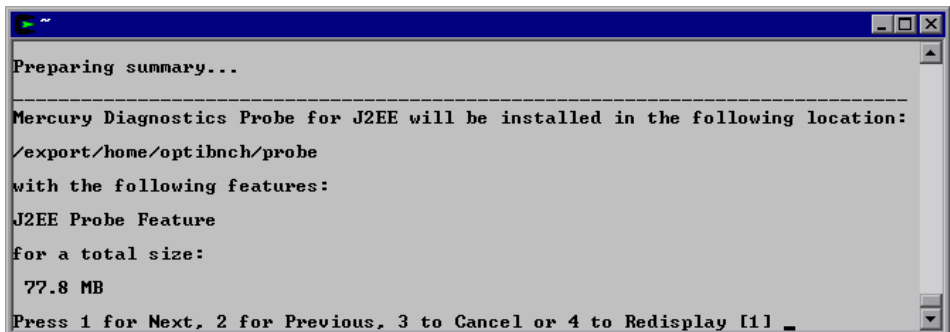
Enter the path to the installation directory for the Deep Diagnostics Server and then enter **1** to continue.

- 16** If you specified that the Probe will work with Deep Diagnostics, the installer asks for the Deep Diagnostics Application Definition name.



The Deep Diagnostics Application Definition name must exactly match the name specified for the Application Definition in Deep Diagnostics. Enter the Application Definition Name that was created in Deep Diagnostics and then enter **1** to continue.

- 17** The installer displays a Pre-Installation Summary.

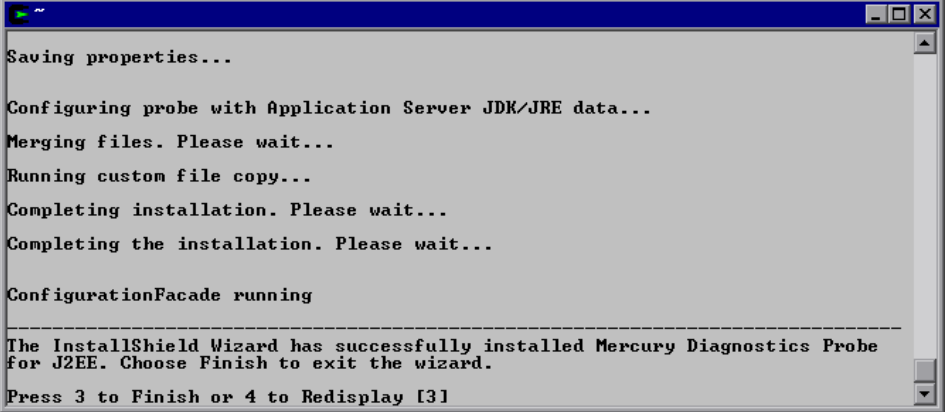


Review the information to make sure that you are satisfied. Enter **1** to start the Probe Installation.





- 20** The installer installs the J2EE Probe.



```
Saving properties...
Configuring probe with Application Server JDK/JRE data...
Merging files. Please wait...
Running custom file copy...
Completing installation. Please wait...
Completing the installation. Please wait...

ConfigurationFacade running

-----
The InstallShield Wizard has successfully installed Mercury Diagnostics Probe
for J2EE. Choose Finish to exit the wizard.
Press 3 to Finish or 4 to Redisplay [3]
```

- 21** After installation is complete, enter **3** to exit the installer.
- 22** Verify the Probe installation as described in “Verifying the J2EE Probe Installation” on page 134.
- 23** Configure your application server so it can use the J2EE Probe. For more information, see “Introducing J2EE Probe Installation and Application Server Configuration” on page 97.
- 24** If you will be using Mercury Deep Diagnostics for J2EE, configure the J2EE Probe to collect data for Deep Diagnostics. For more information, see “Configuring the J2EE Probe and Application Server for Deep Diagnostics” on page 375.

## Installing the J2EE Probe on a z/OS Mainframe

Instructions for installing the J2EE Probe from the tar file that is included on the product CD and from a pax archive that was created on a z/OS machine have been provide in this section.

---

**Note:** The J2EE Probe only captures the Deep Diagnostics Java metrics on a z/OS mainframe. The Probe does not capture the system level metrics for CPU, memory and network latencies. Load test metrics are not captured for z/OS.

---

### Editing Property Files on a z/OS Mainframe

The following tips have been provided to help you to update the property files in a z/OS environment and make sure that the updates are stored in the character set that can be used by the Probe.

The diagnostics property files are stored in ASCII format. In order to edit these files in the z/OS environment you must convert them from ASCII to EBCDIC. When you have completed your edits, the file must be converted back to ASCII.

---

**Note:** If you have access to an ASCII error such as viascii you do not have to worry about converting to EBCIDIC.

---

Use the following command to convert a file from ASCII to EBCDIC:

```
iconv -f ISO8859-1 -t IBM-1047 'ASCII_FILE' > 'EBCDIC_OUTPUT_FILE_NAME'
```

Use the following command to convert a file from EBCDIC to ASCII:

```
iconv -f IBM-1047 -t ISO8859-1 'EBCDIC_FILE' > 'ASCII_OUTPUT_FILE_NAME'
```

## Installing the J2EE Probe on z/OS from the product CD

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

- 1 Upload J2EEProbe-<version\_nbr>.tar to the z/OS machine.
- 2 Run `pax -rf J2EEProbe-<version_nbr>.tar` from the POSIX shell. This creates a directory called J2EEProbe.
- 3 Open the property file `<probe_install_dir>/etc/probe.properties` using `viascii` or another ASCII editor. Set the `id` property to a unique name that has not been assigned to any other probe as follows:  
`id=<unique_probe_name>`
- 4 Ensure that the permissions bits are set correctly on the `<probe_install_dir>/log` directory so that the user that starts the application server has write (`o+w`) permissions (`chmod 644 <logs>`).
- 5 Run the JRE Instrumenter as documented in “Running the JRE Instrumenter” on page 141.
- 6 Configure the application server to load the probe files. For instructions on configuring the application server to load the probe files and other parameters, see Chapter 10, “Configuring the Application Server and Probe Using the Configuration Utility.”

---

**Note:** You can view the system log by accessing the primary operator’s console in SDSF.

---

- 7 Verify the Probe installation as described in “Verifying the J2EE Probe Installation” on page 134.

## Installing the J2EE Probe on z/OS From a Pax Archive

This section describes how to deploy the J2EE Probe from a pax archive that was created on a z/OS machine. You may want to use this method to install the Probe on multiple z/OS machines after you have successfully installed and configured the first Probe on a z/OS machine.

### To install the J2EE Probe on a z/OS mainframe using a pax archive:

- 1 Copy the compressed pax archive to the machine where the J2EE Probe is to be installed.
- 2 Uncompress the pax archive using the following command:

```
uncompress 'COMPRESSED_PAX_ARCHIVE
```

This creates an uncompressed pax file in the current working directory.

- 3 Extract the archive into the directory where you want the Probe installed using the following command:

```
pax -r * < 'FULL_PATH_TO_PAX_FILE
```

---

**Note:** This command extracts the archive into the current directory. Make sure that you run the command from the directory where you want the Probe installed.

---

- Convert the modified file from EBCDIC back to ASCII.
- 4 Verify the Probe installation as described in “Verifying the J2EE Probe Installation” on page 134.
  - 5 Configure your application server so it can use the J2EE Probe. For more information, see “Introducing J2EE Probe Installation and Application Server Configuration” on page 97.
  - 6 If you will be using Mercury Deep Diagnostics for J2EE, configure the J2EE Probe to collect data for Deep Diagnostics. For more information, see

“Configuring the J2EE Probe and Application Server for Deep Diagnostics” on page 375.

## Installing the J2EE Probe Using the Generic Installer

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 will work with other platforms that have not yet been certified. A generic installer has been provided to allow you to install the probe on these other platforms.

To get the Probe to work on the platforms that are not supported in the regular installer, you must run the generic installer and then 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 a platform that is not certified:**

- 1** Locate the generic installer on the Mercury Diagnostics for J2EE & .NET CD at:  
  
CD2/Probe/GenericProbeTar/J2EEProbe-3.3.11.0.tar
- 2** Unzip/untar `J2EEProbe-3.3.11.0.tar` to the probe installation directory that you want to use.
- 3** Manually configure the Probe for the product mode that is appropriate by following the Probe configuration instructions documented in “Configuring the J2EE Probe for Various Mercury Products” on page 370.
- 4** Run the JRE Instrumenter for the application server that the Probe will be monitoring by following the instructions documented in “Running the JRE Instrumenter” on page 141.
- 5** If the Probe will be used with Deep Diagnostics, update the `pfmt.godel.probedir` property in `$DD/etc/pfmt.godel.expert.properties`.
- 6** To configure the Probe so that it can register with the Commander, set the host name and the port using the `registrar.url` property which can be found in the property file:

<probe\_install\_dir>\etc\dispatcher.properties

The following is an excerpt from the dispatcher.properties file showing the registrar.url property.

---

**Note:** Make sure that you update the registrar.url property in the property file that is appropriate for your network configuration. The property registrar.url exists in two different property files in <probe\_install\_dir>\etc. The property in dispatcher.properties is used to provide the url for the registrar when there is not a proxy. The 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.

---

- 7 Verify the Probe installation as described in “Verifying the J2EE Probe Installation” on page 134.

## Verifying the J2EE Probe Installation

Use the System Health Monitor to verify the installation of the J2EE Probe. For instructions on how to use the System Health Monitor, see Appendix A, “Using the System Health Monitor.”

---

**Note:** The Probe does not register with the Commander until it is started. The Probe is started when the instrumented application server is started. Therefore, you will not be able to verify the installation of the Probe using the System Health Monitor until you have configured the Probe and Application Server as described in “Introducing J2EE Probe Installation and Application Server Configuration” on page 97.

---

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

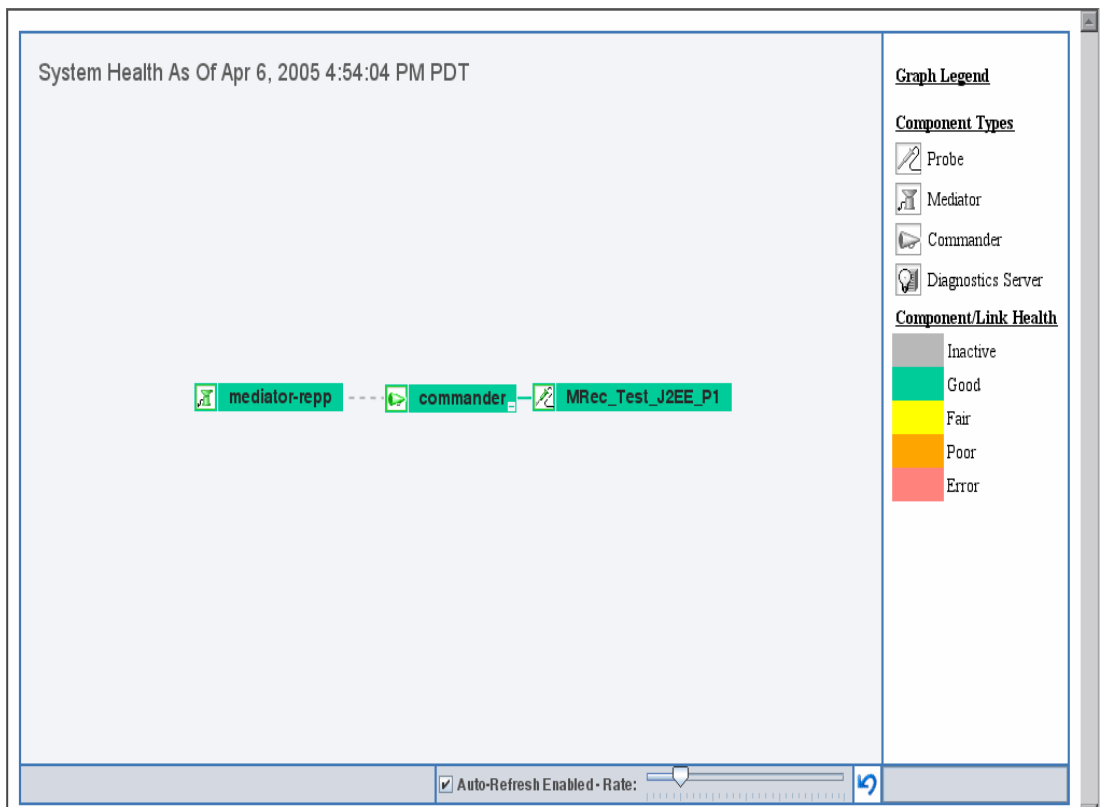
- ▶ The Mediator was successfully installed and has established connectivity with the Commander.
- ▶ The J2EE Probe was successfully installed and has established connectivity with the Commander.

The new J2EE Probe is shown as a child of the Commander on the System Health Monitor.

---

**Note:** The J2EE Probe will appear colored grey in the System Health Monitor when the application that it is monitoring has been stopped. Once the application has been started the Probe will be colored green.

---



## Using the J2EE Probe with Deep Diagnostics

Mercury Deep Diagnostics for J2EE does not include a probe in its installation; instead, it uses the J2EE Probe installed with LoadRunner 8.1 or Performance Center 8.1. For instructions about configuring the Probe for use with Deep Diagnostics, see “Configuring the J2EE Probe and Application Server for Deep Diagnostics” on page 375.

## Overriding the Default Probe Host Machine 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 Commander to communicate with the Probe using the host name that was assigned when the Probe was installed. The `registered_hostname` property allows you to override the default host machine name that the Probe uses to register itself with the Commander.

To override the default host machine name for a Probe, set the `registered_hostname` property located in `<probe_install_dir>/etc/dispatcher.properties` to an alternate machine name or IP Address that will let the Commander communicate with the Probe.

## Determining the Version of the J2EE Probe That is Installed

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

### To determine the J2EE Probe version:

Locate the version file `<probe_install_dir>\dat\version.txt`. The file contains the four digit version number, as well as the build number.



## Upgrading to a Newer Version of the J2EE Probe

The following instructions will guide you in upgrading from an older version of the Mercury Diagnostics Probe for J2EE.

---

**Note:** The installer does not upgrade the Probe. The following instructions are manual steps that you must take to make sure that the upgrade works as expected.

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 LoadRunner 8.1, Performance Center 8.1 and BAC 5.x installations and will not work with earlier versions of these products. Likewise, the previous versions of the Probe will 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 configurations such as instrumentation changes, buffering changes, port numbers, mediator & commander locations, probe.id and product mode.

---

- 4** Uninstall the Probe. See “Uninstalling the J2EE Probe” on page 140 for instructions on uninstalling the Probe.
- 5** Install the new version of the J2EE Probe as instructed in this guide in Chapter 2, “Preparing to Install Mercury Diagnostics for J2EE & .NET.”

**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 continue with the next steps.

---

- 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 back-up information to configure the new probe.
  - ▶ 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.
  - ▶ Rename the <probe\_install\_dir>/etc/auto\_detect.points file that was just installed to new\_auto\_detect.points.
  - ▶ Copy the backup copy of auto\_detect.points into <probe\_install\_dir>/etc
  - ▶ 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.
  - ▶ 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.

---

- ▶ In `capture.properties` add the following property:  
`ac.register.sink=true`
  - ▶ In `dispatcher.properties` add the following properties:  
`dispatcher.properties.file.name=dispatcher.properties`  
`registrar.mediator_ping.time=90s`  
`ac.autostart=true`
  - ▶ 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.*`
  - ▶ 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 Chapter 7, “Introducing J2EE Probe Installation and Application Server Configuration.”

---

**Note:** If a firewall separates your probe and mediator 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. See “Configuring Diagnostics Components to Work with a Firewall” on page 399 for more information.

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 execute **uninstall.exe** which is located in the <probe\_install\_dir>\Probe\\_uninst directory.
- ▶ On a Unix machine execute **uninstall\*** which is located in the <probe\_install\_dir>\Probe\\_uninst directory.

# 9

---

## Running the JRE Instrumenter

This chapter explains the process for configuring the application server on which your applications run and the J2EE Probe that will monitor your application.

This chapter includes the following sections:

- ▶ About the JRE Instrumenter
- ▶ Running the JRE Instrumenter

### About the JRE Instrumenter

When you install a Probe, you must run the JRE Instrumenter to prepare your application for the instrumentation that allows the Probe to monitor the processing. The JRE Instrumenter will run automatically during the Probe installation unless you elect not to run it at that time. If you elect not to run the JRE Instrumenter during the installation of the Probe, you can run it manually as described in “Running the JRE Instrumenter” on page 142.

When the JDK (java.exe executable) used by your application server changes, you must manually run the JRE Instrumenter again in order for the Probe to be able 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. See “Configuring the Probes for Multiple Application Server Instances” on page 196 for details.

---

---

**Note:** In this chapter “<probe\_install\_dir>” is used to indicate the directory where the J2EE Probe was installed.

---

## Running the JRE Instrumenter

To run the JRE Instrumenter:

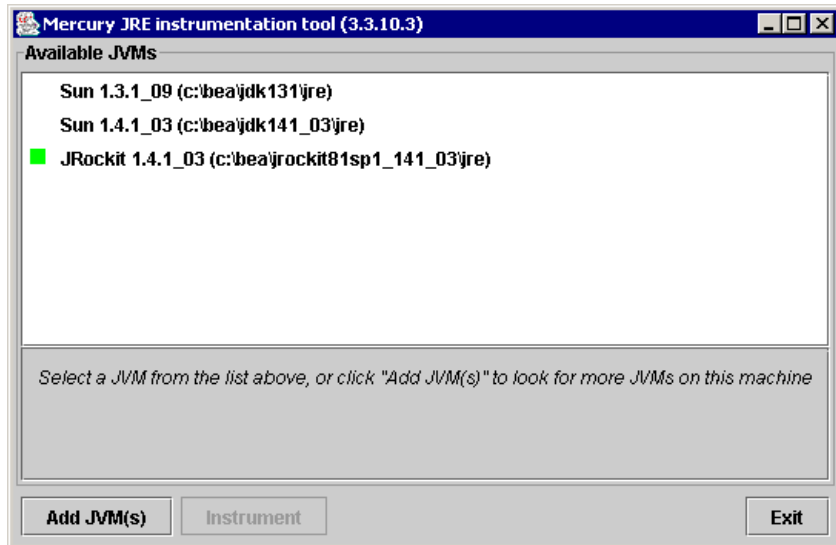
- 1 Open <probe\_install\_dir>\bin. This is the directory where the JRE Instrumenter executable is located. Replace <probe\_install\_dir> with the path to the directory where the Probe was installed.
- 2 Execute the jreinstrumenter.cmd command if the Probe is installed on a Windows machine, or jreinstrumenter.sh if the Probe is installed on a UNIX machine.

To execute the JRE Instrumenter from the command prompt in Windows use the following command:

```
java -jar jreinstrumenter.jar
```

When the instrumenter starts, the Mercury JRE Instrumentation Tool dialogue box opens.

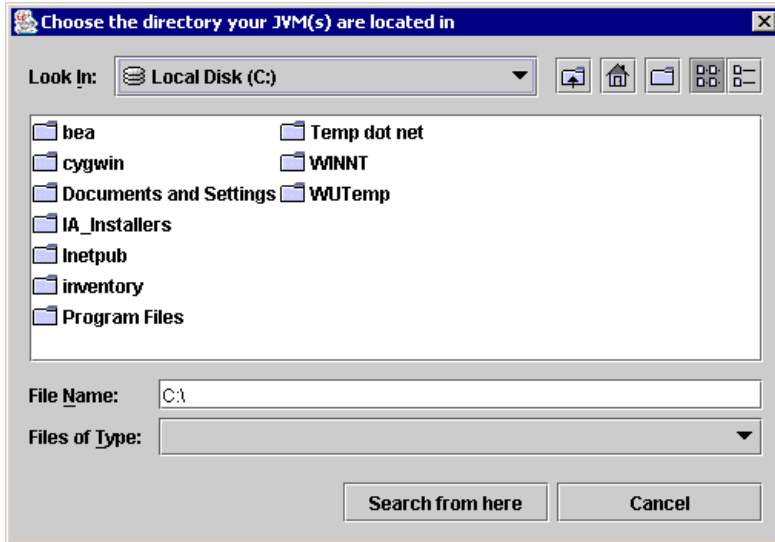
The JVMs that are available for instrumentation are listed in the **Available JVMs** list box. The JVMs that have already been instrumented are listed with green square preceding the name of the JVM.



**To add a JVM to the Available JVM List:**

- 1** Click **Add JVM(s)**.

The following dialogue box opens:



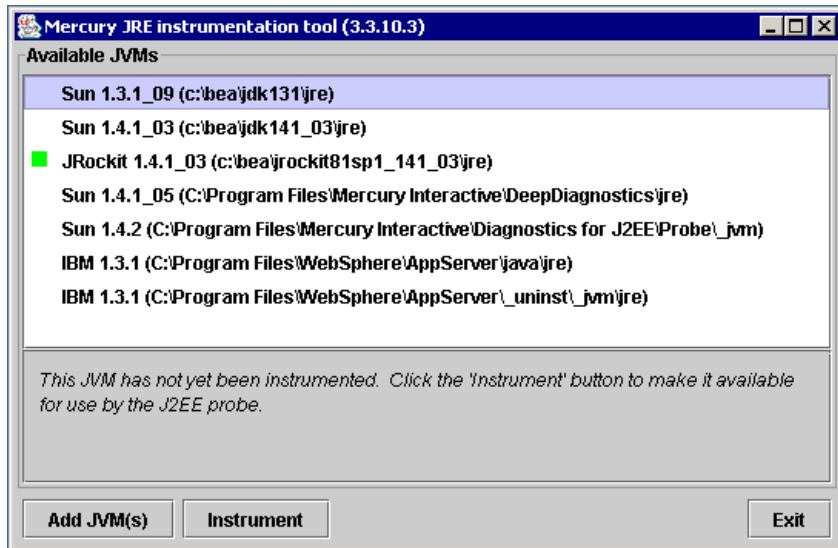
- 2** Navigate to the location where you would like to begin searching for JVMs using the standard Windows Explorer type navigation tools.
- 3** Select the file where you would like to begin the search so that its name appears in the File Name text box.



- 4 Click **Search from here** to instruct the instrumenter to begin searching for JVMs.

The dialog box closes and the Mercury JRE Instrumentation Tool dialogue box opens again, with the command buttons disabled while the tool searches for JVMs.

As the tool locates JVMs, it lists them in the Available JVMs list box.



### To instrument a JVM:

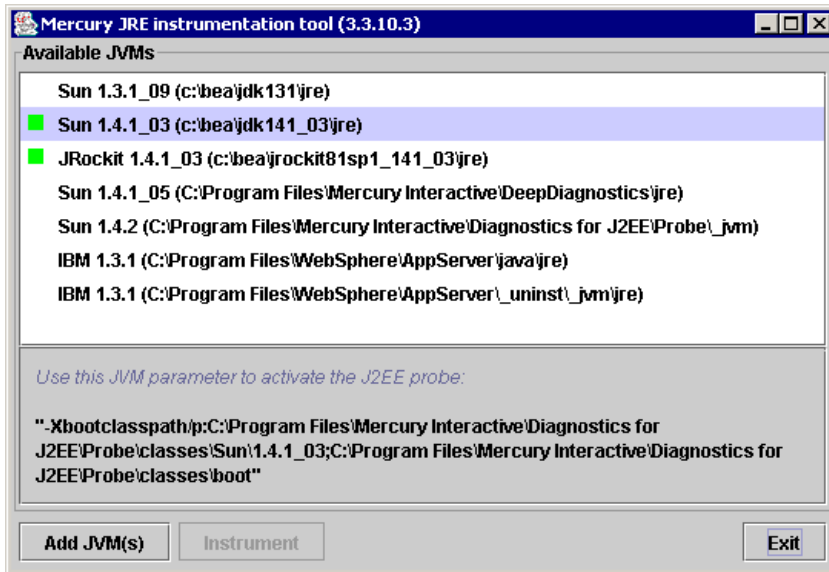
Select a JVM from the Available JVMs list and click **Instrument**.

The instrumenter prepares the JVM to work with the Probe and displays the JVM parameter that you must include in the application server's startup script in order to activate the J2EE Probe in the box above the command buttons. The green icon is displayed in front of the JVM in the Available JVMs list to indicate that the JVM has been instrumented.

---

**Note:** When you select an instrumented JVM from the Available JVMs list, the JVM parameter that needs to be added to the application server's startup script for that JVM is displayed in the box above the command buttons.

---



# 10

---

## Configuring the Application Server and Probe Using the Configuration Utility

This chapter describes how to use the Configuration Utility to configure the Application Server and the J2EE Probe.

This chapter includes the following sections:

- About the Configuration Utility
- Starting the Mercury Configuration Utility
- Defining and Configuring Application Server Instances
- Configuring Application Server Instance Startup Scripts
- Removing the Probe Configuration from the Application Server Instance Startup Script
- Deleting an Application Server Definition
- Reconfiguring a Probe

### About the Configuration Utility

The Mercury Configuration Utility is a tool that helps you configure the certified application servers and the way that they are instrumented for the J2EE Probe. You use the **Define and Configure Application Servers** menu item to set the system properties associated with the application and update the application server startup script to include the path to the Probe's boot classes. The **Reconfigure Unified Probe** menu option lets you reset the Probe properties that were set when you installed the Probe.

---

**Note:** Instructions for configuring the applications servers manually have been provided in Chapter 11, “Configuring the Application Servers Manually.”

---

## Starting the Mercury Configuration Utility

To start the Mercury Configuration Utility:

- 1 On the computer on which the Probe is installed, open `<probe_install_dir>\util\bin`, where `<probe_install_dir>` is the directory where the Probe is installed.
- 2 Execute the appropriate command to run the configuration utility.
  - UNIX

```
config.sh
```

- Windows:

```
config.cmd
```

The Mercury Configuration Utility window opens.

```
=====
Mercury Configuration Utility
=====

Welcome to the Mercury Configuration Utility. This program will allow you to
configure various properties of your Mercury products. Please choose an
option.

  1. Define and Configure Application Servers
  2. Reconfigure Mercury Lifecycle Probe
  3. Quit

Choice:
```

## Defining and Configuring Application Server Instances

To define and configure the application server instance:

- 1 From the Configuration Utility main menu, select the **Define and Configure Application Servers** option by typing **1** and pressing Enter. The following screen is displayed.

```

=====
Define and Configure Application Servers
=====
Use the following options to configure or unconfigure an Application Server for
use with the Mercury Lifecycle Probe. You must provide information about your
Application Server <option 1> before configuring it <option 2>.

  1. Provide information about your Application Server
  2. Apply configuration changes to your Application Server to work
     with the Mercury Lifecycle Probe
  3. Revert configuration changes to your Application Server
  4. Delete information about your Application Server
  5. Return to previous menu

Choice: 1_

```

Select the **Provide information about your Application Server** menu option by typing **1** and pressing Enter.

- 2 The following screen, providing the application server information, is displayed.

```

=====
Provide Application Server Information
=====
You will now be asked about your Application Server. The information you enter
will be stored internally. Once the data is entered, you will be able to apply
and revert the changes necessary to use the Mercury Lifecycle Probe quickly and
easily.

[In] or [Enter] to go to next page or [c] to cancel/exit:

```

Press Enter to continue.

- 3 The Configuration Utility prompts you to enter an ID for your application server data. This ID will uniquely identify the application configuration information that you are about to enter.

```
=====
Create an ID for your Application Server data
-----
The ID will be associated internally with your Application Server data.  A good
ID might be something like Weblogic_Petstore.

Enter your new Application Server data ID: Weblogic_Petstore
```

Enter an ID (for example, Weblogic\_Petstore), press Enter, and then press Enter again.

- 4 The Configuration Utility displays the screen to determine whether the Probe for this application server is to be used with Deep Diagnostics.

```
=====
Application Server Configuration Information
-----
Please answer the following questions about your Application Server.

If you will be capturing data using Mercury's Deep Diagnostics for J2EE, you
will be prompted below for an Application Definition Name.  The name you supply
will need to be the same as the name of the Application Definition you create
in Deep Diagnostics.  For more information, please see the Deep Diagnostics
User Guide.

Will you be capturing data using Mercury's Deep Diagnostics for J2EE?(n)
```

Enter **y** if the probe will be used with Deep Diagnostics. Enter **n** if it will not be used with Deep Diagnostics. If you leave this value blank, the default is **n**.

- 5 In the next screen you select the type of application server that you are configuring.

```
If you will be capturing data using Mercury's Deep Diagnostics for J2EE, you
will be prompted below for an Application Definition Name. The name you supply
will need to be the same as the name of the Application Definition you create
in Deep Diagnostics. For more information, please see the Deep Diagnostics
User Guide.

Will you be capturing data using Mercury's Deep Diagnostics for J2EE?(n)

Please select an Application Server type:
 1. BEA WebLogic 6.x
 2. BEA WebLogic 7.x
 3. BEA WebLogic 8.x
 4. IBM WebSphere 4.x
 5. IBM WebSphere 5.x
 6. Oracle 9ias
 7. Oracle 10g
 8. JBoss 3.x
 9. SAP NetWeaver

Choice: 1
```

Enter the appropriate number for your application server and press Enter.

---

**Note:** The next screen displayed by the Configuration Utility will depend upon the application server you selected. This document provides instructions for WebLogic 6.1. Other application servers have similar procedures.

---

- 6 If you chose WebLogic 6.1 on the previous screen, the Configuration Utility prompts you to enter the full path to the WebLogic Home Directory. Enter the path, including the letter of the hard drive, as in the following example:

```
c:\bea\wlserver6.1
```

Press Enter.

- 7 The Configuration Utility then prompts you to enter the full path to the WebLogic Server Instance startup script, including the letter of the drive (in Windows) and the name of the script executable. Enter the path as in the following example:

```
c:\bea\wlserver6.1\config\petstore\startPetStore.cmd
```

Press Enter twice.

- 8 Press Enter to proceed with defining the Application Server Instance. The following screen is displayed.

Press Enter as prompted to monitor the progress and review the summary of the results.

```
=====
Defining Your Application Server Instance
=====
This wizard will now define your Application Server instance.
At anytime during the execution, you may press 's' then <Enter> to stop the
execution.
- Defining Weblogic_Petstore
  |.....| Succeeded
Press <Enter> to continue...
```



## Configuring Application Server Instance Startup Scripts

To configure an application server instance startup script:

- 1 From the Configuration Utility main menu, select the **Define and Configure Application Servers** option by typing **1** and pressing Enter. The following screen is displayed:

```

=====
Define and Configure Application Servers
=====
Use the following options to configure or unconfigure an Application Server for
use with the Mercury Lifecycle Probe. You must provide information about your
Application Server (option 1) before configuring it (option 2).

  1. Provide information about your Application Server
  2. Apply configuration changes to your Application Server to work
    with the Mercury Lifecycle Probe
  3. Revert configuration changes to your Application Server
  4. Delete information about your Application Server
  5. Return to previous menu

Choice: 1_

```

- 2 Select the **Apply configuration changes to your Application Server...** menu option by typing **2** and pressing Enter.
- 3 The Configuration Utility starts the Configure Application Server Instance wizard and displays the following prompt:

```

=====
Configure Application Server Instance
=====
This wizard will guide you through the process of configuring a previously
defined Application Server instance.

[In] or [Enter] to go to next page or [c] to cancel/exit:

```

Press Enter to continue.

- 4 The following screen lists the defined application server instances.

```

=====
Application Server Instance
=====
Select the Application Server instance that should you would like to
configure.

Instance Name:
  1. web_05
  2. web_07
  3. web_09

Choice: _

```

Enter the number of the application server instance that you would like to configure and press Enter twice.

- 5 The following screen asks you to specify a directory where the Configuration Utility can store the application server startup script after it has been modified to use the probe's boot classes.

```
=====
Select Modified File Output Directory
=====
In order to properly configure your Application Server for use with the Mercury
Lifecycle Probe, some files of your Application Server installation may need to
change. You can enter a directory below where you would like the modified
files to be written.

If you enter nothing, the modified files will be written to the original
location.

If you enter a directory, you will need to manually copy the files from this
directory to their original location. If you know that you do not have
permission to overwrite files in the original location, you should enter a
directory below.

Enter a directory where modified files should be written: c:\bea\temp_
```

Enter the path to the temporary directory after the prompt.

Press Enter to proceed with the configuration.

- 6 Press Enter at each of the following prompts to initiate the processing to configure the application server startup scripts.
- 7 Rename the original copy of the startup script so that you can keep it as a backup.
- 8 Copy the modified startup script from the temporary directory to the folder where the application server startup script is stored.

## Removing the Probe Configuration from the Application Server Instance Startup Script

To restore an application server instance startup script's settings:

- 1 From the Configuration Utility main menu, select the **Define and Configure Application Servers** option by typing **1** and pressing Enter. The following screen is displayed:

```

=====
Define and Configure Application Servers
=====
Use the following options to configure or unconfigure an Application Server for
use with the Mercury Lifecycle Probe. You must provide information about your
Application Server (option 1) before configuring it (option 2).

  1. Provide information about your Application Server
  2. Apply configuration changes to your Application Server to work
    with the Mercury Lifecycle Probe
  3. Revert configuration changes to your Application Server
  4. Delete information about your Application Server
  5. Return to previous menu

Choice: 1_

```

- 2 Select the **Revert configuration changes to your Application Server** menu option by typing **3** and pressing Enter.
- 3 The Configuration Utility starts the Un-configure Application Server Instance wizard and displays the following prompt.

```

=====
Un-configure Application Server Instance
-----
This wizard will guide you through the process of un-configuring an
[In] or [Enter] to go to next page or [c] to cancel/exit:
=====
Application Server Instance
-----
Select the Application Server instance that should you would like to
unconfigure.

Instance Name:
  1. web_05
  2. web_07
  3. web_09

Choice: 1_

```

Press Enter after the prompt. The Configuration Utility displays a list of the available application server instances.

The screen lists the application server instances.

Enter the number of the application server instance that you would like to un-configure and press Enter twice.

- 4 The wizard displays the following screen to ask you to specify a directory where the configuration utility can store the application server startup script after it has been modified to use the probe's boot classes.

```
=====
Select Modified File Output Directory
=====
In order to properly configure your Application Server for use with the Mercury
Lifecycle Probe, some files of your Application Server installation may need to
change. You can enter a directory below where you would like the modified
files to be written.

If you enter nothing, the modified files will be written to the original
location.

If you enter a directory, you will need to manually copy the files from this
directory to their original location. If you know that you do not have
permission to overwrite files in the original location, you should enter a
directory below.

Enter a directory where modified files should be written: c:\bea\temp_
```

- 5 Enter the path to the temporary directory after the prompt.  
Press Enter to proceed with the configuration.
- 6 Press Enter at each of the prompts to initiate the processing to configure the application server startup scripts.

## Deleting an Application Server Definition

To delete information about an application server:

- 1 From the Configuration Utility main menu, select the **Define and Configure Application Servers** option by typing **1** and pressing Enter. The following screen is displayed:

```

=====
Define and Configure Application Servers
=====
Use the following options to configure or unconfigure an Application Server for
use with the Mercury Lifecycle Probe. You must provide information about your
Application Server (option 1) before configuring it (option 2).

  1. Provide information about your Application Server
  2. Apply configuration changes to your Application Server to work
     with the Mercury Lifecycle Probe
  3. Revert configuration changes to your Application Server
  4. Delete information about your Application Server
  5. Return to previous menu

Choice: 1_

```

- 2 Select the **Delete information about your Application Server** menu option by typing **4** and pressing Enter.
- 3 The Configuration Utility starts the Configure Application Server Instance wizard and displays the following prompt.

```

=====
Delete Application Server Instance Definition
=====
This wizard will guide you through the process of deleting an Application
Server instance that was previously defined.

[In] or [Enter] to go to next page or [c] to cancel/exit: _

```

Press Enter to continue.

- 4 The wizard displays a list of defined application server instances.

```

=====
Application Server Instance
=====
Select the Application Server instance that should you would like to delete.

Instance Name:
  1. web_05
  2. web_07
  3. web_09

Choice: 2_

```

Enter the number of the application server instance that you want to delete and press Enter twice.

- 5 The process for deleting the application server instance definition begins.

Press Enter at each of the prompts to initiate the processing. If the deletion succeeds, the following screen is displayed:

```
=====
Deleting the Application Server Instance Definition
=====
This wizard will now delete the selected Application Server instance
definition.

At anytime during the execution, you may press 's' then <Enter> to stop the
execution.

- Deleting web_0?
  [.....] Succeeded

Press <Enter> to continue...
```

## Reconfiguring a Probe

The Configuration Utility allows you to reconfigure the J2EE Probe so that you do not have to update the Probe properties manually or reinstall the Probe from the installation CD.

To reconfigure the probe:

- 1 From the Configuration Utility main menu, select the **Reconfigure Unified Probe** option by typing **2** and pressing Enter. The following screen is displayed.
- 2 The Reconfigure Mercury Lifecycle Probe menu lists the properties that you can manipulate using the Configuration Utility.

```
=====
Reconfigure Mercury Lifecycle Probe
=====
Select one of the following Mercury Lifecycle Probe properties to configure:

  1. Product Mode = AD
  2. Probe ID = benzi_s_probe
  3. Commander Host Name = straw
  4. Commander Port = 2006
  5. Return to previous menu

Choice:
```

The following properties are displayed on the Reconfigure Mercury Lifecycle Probe menu:

► **Product Mode:**

The value of the Product Mode property should be **AD** when you are using the Probe with LoadRunner 8.1 or Performance Center 8.1 unless you will also be using the Probe with Deep Diagnostics. In this case the value of Product Mode should be **AD,DeepDiagnostics**.

The value of this property should be **AM** when you are using the Probe with BAC 5.0 unless you will also be using the Probe with Deep Diagnostics. In this case the value of Product Mode should be **AM,DeepDiagnostics**.

The value of this property should be **AC** when you are using the Probe with Mercury Diagnostics Profiler.

- **Probe ID:** The probe ID must be unique for each probe that is involved in a run. When you are using the Probe with LoadRunner, Performance Center, or BAC, you should make sure that the probe ID provides some indication of whether the probe is a .NET Probe or a J2EE Probe. (For example, if the probe is a J2EE Probe, you might name it PetStoreJ2EE01.) This will make it easier to interpret the monitoring results.
- **Commander Host Name:** To change the Commander, choose **4** and press Enter. The installer prompts you to enter the host name of the Commander machine.

---

**Note:** Mercury Diagnostics Profiler does not use the Commander component.

---

- **Commander Port:** This property specifies the port the probe will use to communicate with the Commander. To change the Commander port, choose **5** and press Enter. The installer prompts you to enter the port number on the Commander machine. Enter the port number and press Enter.

---

**Note:** Mercury Diagnostics Profiler does not use the Commander component.

---

- 3** To exit the probe configuration menu, choose **6** and press Enter. The Mercury Configuration Utility menu is displayed.
- 4** To exit the Mercury Configuration Utility, choose **3** in the Mercury Configuration Utility menu.



# 11

---

## Configuring the Application Servers Manually

This chapter explains the process for configuring the application server on which your applications run and the J2EE Probe that is to monitor your application.

This chapter includes the following sections:

- ▶ About Configuring the Application Server
- ▶ Configuring WebSphere Application Servers
- ▶ Configuring WebLogic Application Servers
- ▶ Configuring the Oracle9i Application Server
- ▶ Configuring the JBoss Application Server
- ▶ Configuring the SAP NetWeaver Application Server
- ▶ Configuring a Generic Application Server
- ▶ Configuring the Probes for Multiple Application Server Instances

### 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. There are two ways to configure the application servers:

- ▶ Using the automated tool, Mercury Configuration Utility. For more information see “Configuring the Application Server and Probe Using the Configuration Utility,” on page 147.

- By manually updating the application server startup scripts. Instructions are provided in this chapter for updating the certified application servers using manual processes. The first set of instructions can be found in “Configuring WebSphere Application Servers,” on page 163, and instructions for the other certified application servers follow.

It is possible that your site administrator has site-specific methods for making these configuration modifications. If this is the case, the generic procedure described in “Configuring a Generic Application Server” on page 194 should provide the information that the site administrator needs to implement the required configuration changes.

---

**Note:** Please see the section appropriate for your particular application servers. If there is no section for your specific application server, follow the procedure in “Configuring a Generic Application Server” on page 194.

---

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 Instances” on page 196.

---

**Note:** In this chapter “<probe\_install\_dir>” is used to indicate the directory where the J2EE Probe was installed.

---

## Configuring WebSphere Application Servers

Supported versions	Operating systems
3.5, 4.x, 5.x	AIX 4.3 Windows 2000/2003 Solaris 8

WebSphere servers are controlled using the WebSphere Advanced 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 and 5.x versions. It allows you to add the Xbootclasspath property. You may also add any additional arguments to the JVM command line that may be needed.

Note that the appearance of the Console may differ for different versions of WebSphere. The following instructions show different screens for each of the certified WebSphere Application Server versions. Note that the changes are implemented in slightly different ways in each version. The examples shown in this section may not correspond exactly to your WebSphere version, but the principle is the same, that is, the parameters must be inserted in the appropriate fields.

---

**Note:** In order to apply your changes, you must click **Apply** in each tab in the Administrative Console before moving to another tab.

---

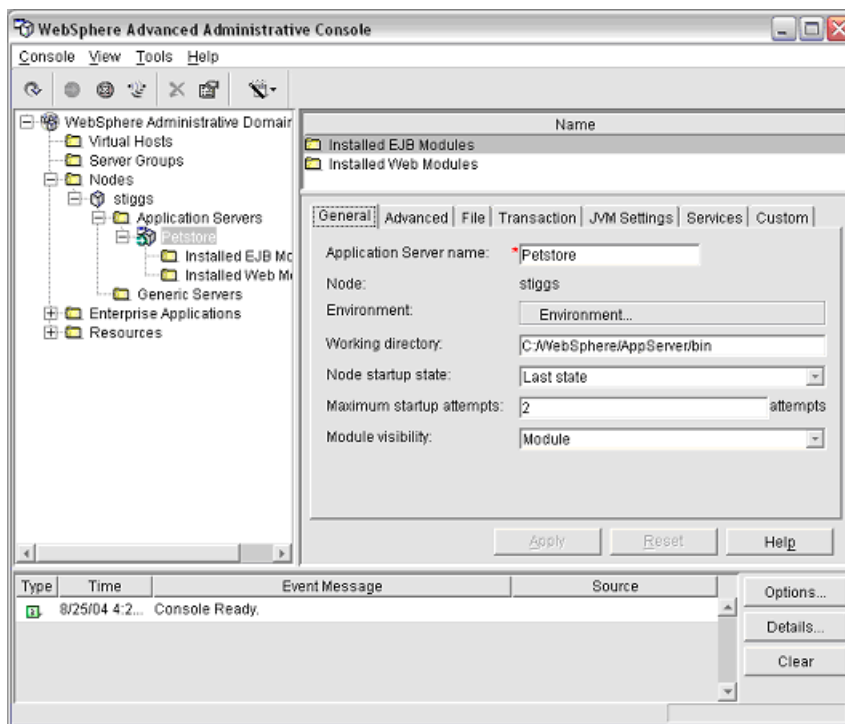
### WebSphere 4.0

To configure a WebSphere 4.0 application server:

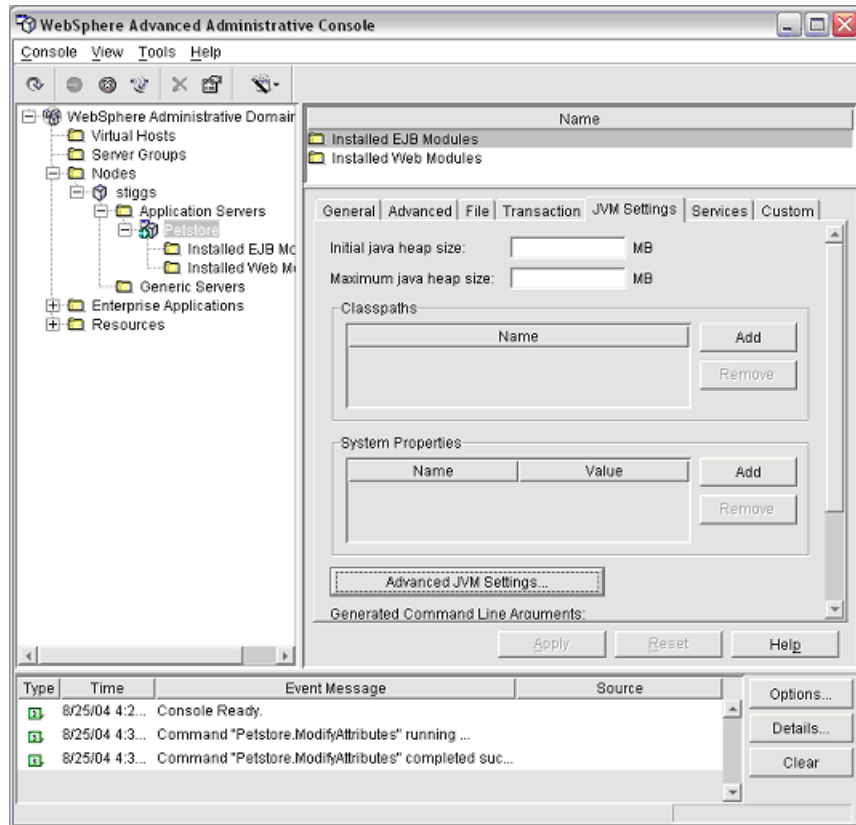
- 1 Use your Web browser to access the Web page for the WebSphere Administrative Console for the application server instance for which the probe was installed:

```
<WebSphere_Install_Dir>\AppServer\bin\adminclient.sh(bat)
```

- 2 Select the application server from the tree view under the **Nodes** parent. 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:




---

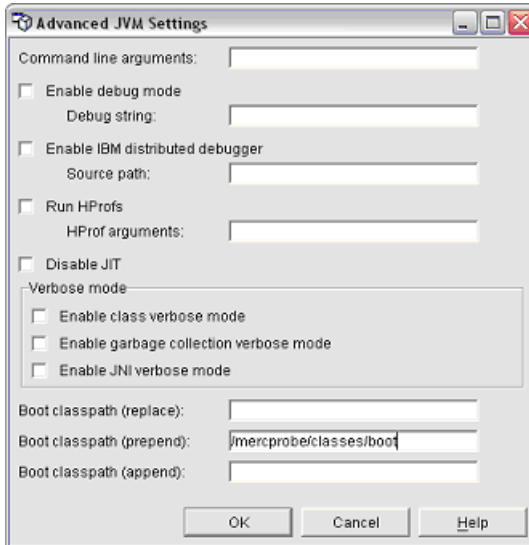
**Note:** If your JVM uses a JIT option, such as `-hotspot` or `-classic`, make sure (in the Java Command Line arguments on the Console) that the `-Xbootclasspath` option is entered following the JIT option.

---

- 4 Click **Advanced JVM Settings** to open the Advanced JVM Settings dialog box.

- 5 Enter the boot classpath property in the **Boot classpath (prepend)** field as follows, where **<probe\_install\_dir>** is the path to the directory where the probe was installed:

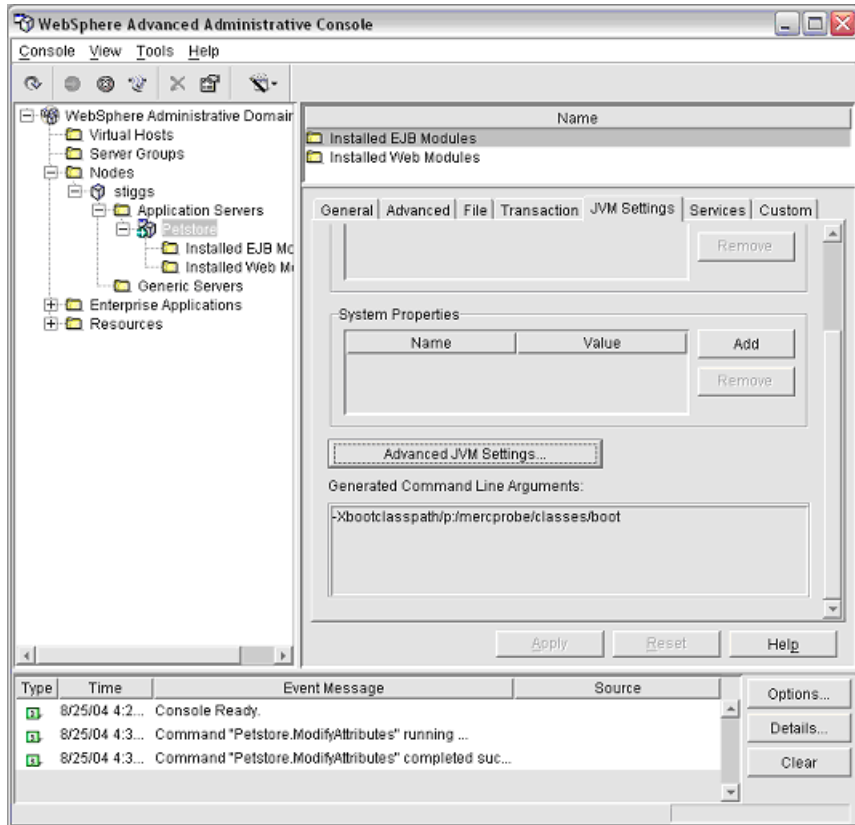
**<probe\_install\_dir>\classes\boot**



- 6 Click **OK** to close the Advanced JVM Settings dialog box. The JVM Settings tab is again displayed.

- 7 In **Generated Command Line Arguments** area, verify that the Xbootclasspath property was created properly.

The settings should be as in the following example:



- 8 Click **Apply** to save your changes so that they will be in effect when you restart the application server.
- 9 Restart the WebSphere application server from the Console menu. You do not need to restart the host machine for your application server.

- 10** 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 you did not run the JRE Instrumenter or did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

You can also refer to the Websphere **stdout.log** and **stderr.log** files for troubleshooting any problems arising from this configuration.



## WebSphere 5.0

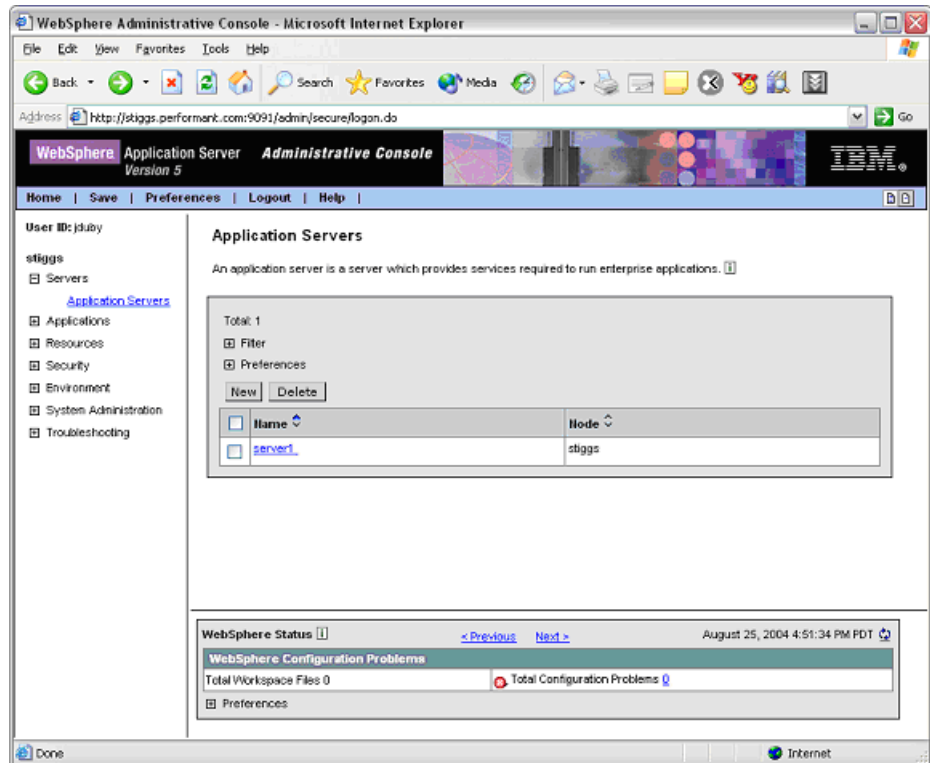
To configure a WebSphere 5.0 application server:

- 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:

```
http://<App_Server_Host>:9090/admin
```

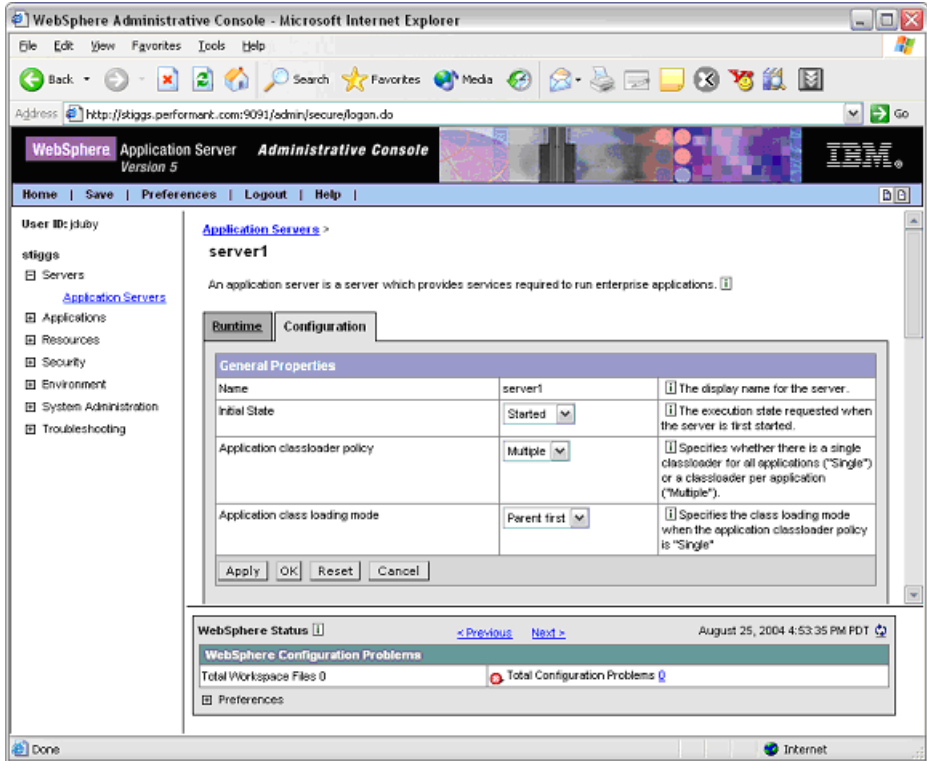
Replace **<App\_Server\_Host>** with the machine name for the application server host.

The Websphere Application Server Administrative Console is displayed.



- 2 In the left panel, select **Servers** and then select **Application Servers**.
- 3 From the list of application servers in the right panel, select the name of the server that you want to configure to be monitored by the J2EE Probe. (In the screen image above, the server name is **server1**.)

The Configuration tab for the selected application server is displayed.



- 4 Scroll down in the **Configuration** tab until the **Process Definition** property is visible in the General Properties column.

Click **Process Definition**.

The screenshot shows the WebSphere Administrative Console interface in a Microsoft Internet Explorer browser window. The address bar shows the URL: `http://stiggs.performant.com:9091/admin/secure/login.do`. The page title is "WebSphere Application Server Administrative Console Version 5". The left navigation pane shows a tree structure with "Process Definition" highlighted in red. The main content area displays a list of configuration properties with their descriptions:

<a href="#">Custom Properties</a>	Additional custom properties for this runtime component. Some components may make use of custom configuration properties which can be defined here.
<a href="#">Administration Services</a>	Specify various settings for administration facility for this server, such as administrative communication protocol settings and timeouts.
<a href="#">Diagnostic Trace Service</a>	View and modify the properties of the diagnostic trace service.
<a href="#">Debugging Service</a>	Specify settings for the debugging service, to be used in conjunction with a workspace debugging client application.
<a href="#">IBM Service Logs</a>	Configure the IBM service log, also known as the activity log.
<a href="#">Custom Services</a>	Define custom service classes that will run within this server and their configuration properties.
<a href="#">Server Components</a>	Additional runtime components which are configurable.
<a href="#">Process Definition</a>	A process definition defines the command line information necessary to start/initialize a process.
<a href="#">Performance Monitoring Service</a>	Specify settings for performance monitoring, including enabling performance monitoring, selecting the PMI module and setting monitoring levels.
<a href="#">End Points</a>	Configure important TCP/IP ports which this server uses for connections.
<a href="#">Classloader</a>	Classloader configuration
<a href="#">Extended Messaging Service</a>	The Extended Messaging Service provides runtime service for the support of Container Managed Messaging.
<a href="#">Staff Service</a>	The Staff Service manages Staff Plugin resources used by a given server.
<a href="#">Activity Session Service</a>	A unit of work service that can be used to coordinate one-phase resources or for extending the activation transaction of an app.

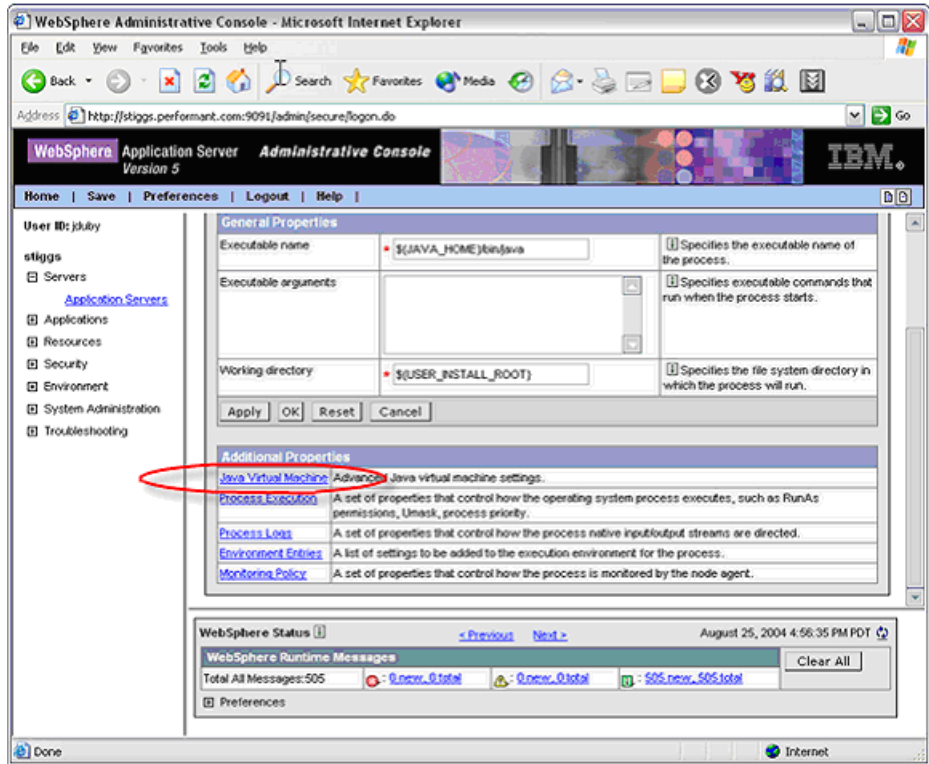
At the bottom of the console, the "WebSphere Status" section shows the date and time: "August 25, 2004 4:54:35 PM PDT". Below this, the "WebSphere Runtime Messages" section displays:

Total All Messages: 505    0 new, 0 total    0 new, 0 total    505 new, 505 total

A "Clear All" button is visible next to the message counts.

- 5 Scroll down in the right panel until the **Java Virtual Machine** additional property is visible.

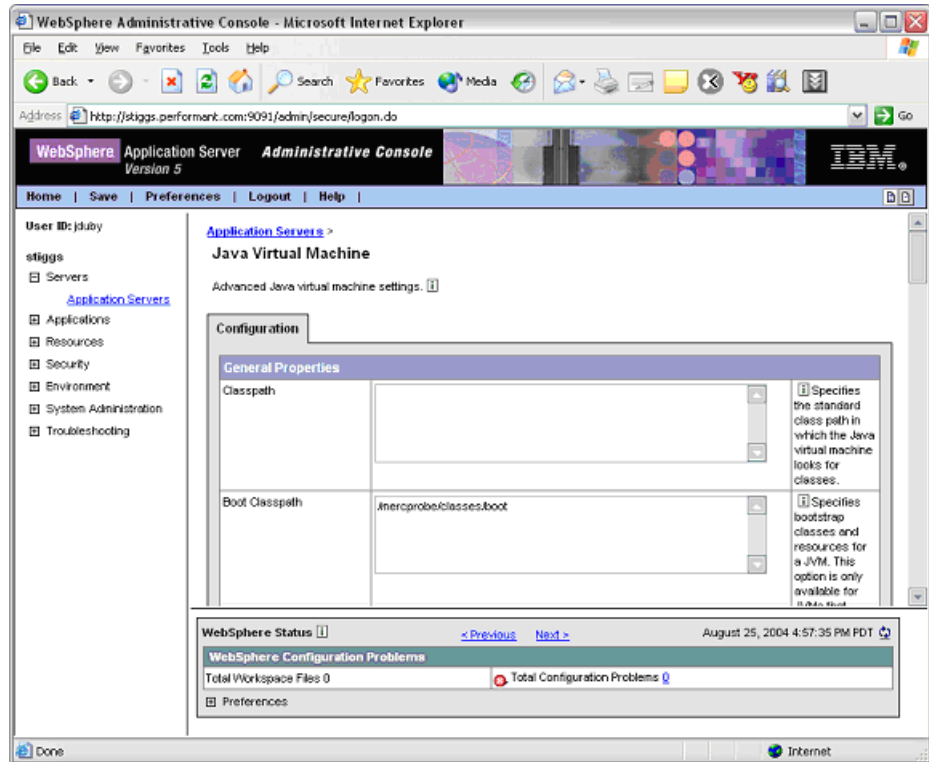
Click **Java Virtual Machine**.



6 The Configuration tab for the Java Virtual Machine is displayed.

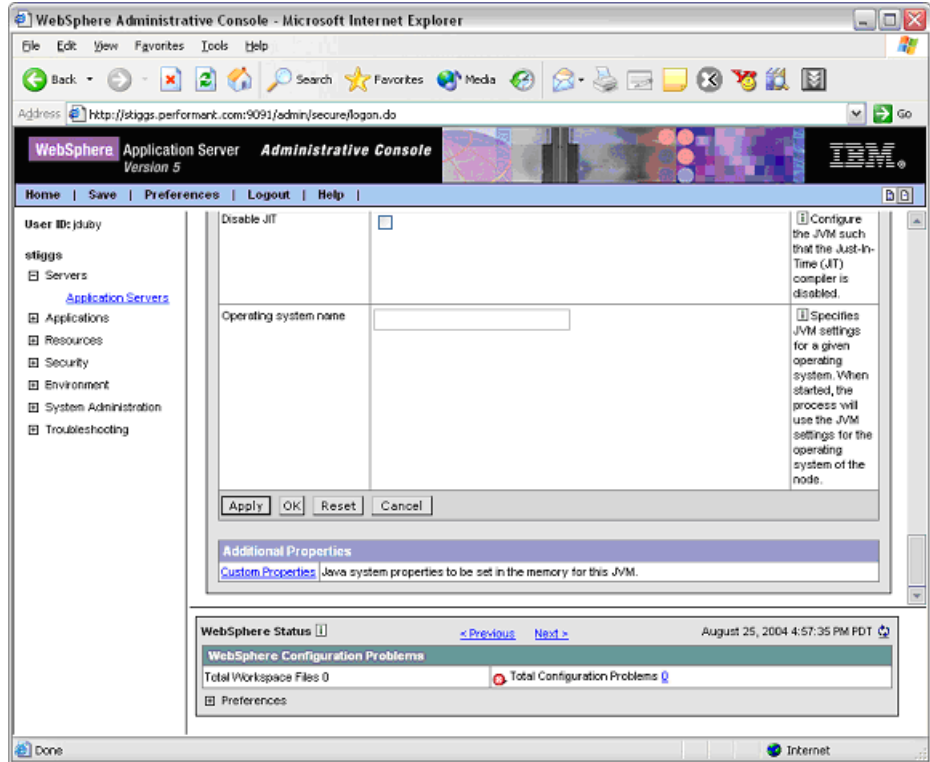
In the **Boot Classpath** box, enter the path to the boot directory for the J2EE Probe as follows, where `<probe_install_dir>` is the path to the location where the Probe was installed:

`<probe_install_dir>\classes\boot`



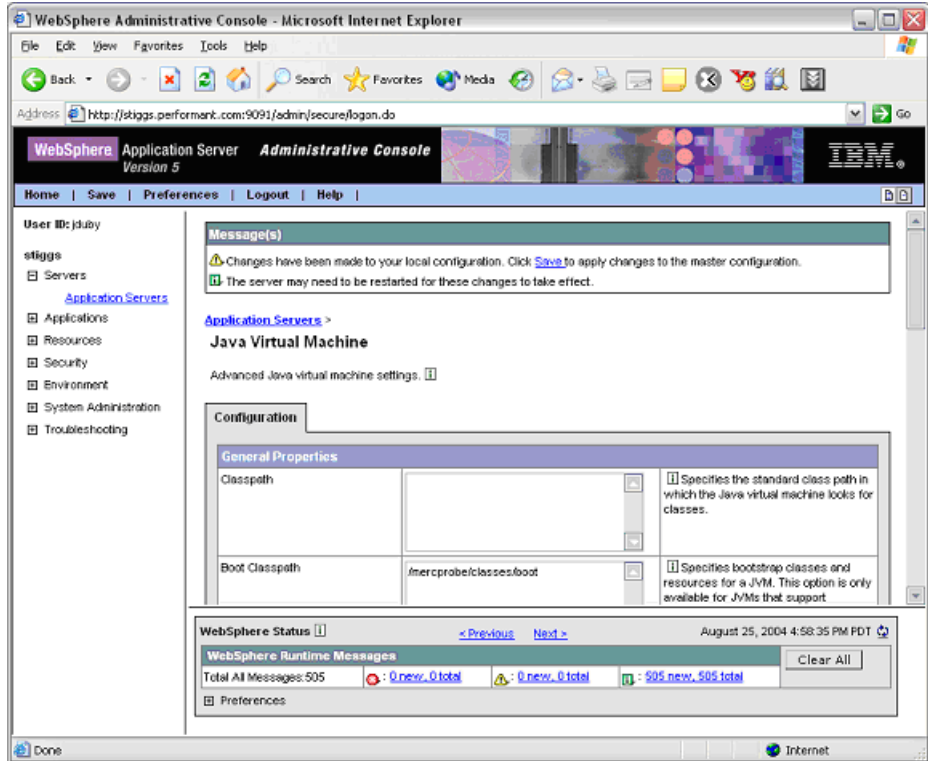
- 7 Scroll to the bottom of the **Configuration** tab until the command buttons are visible, as shown in the following screen.

Click **Apply** to save your changes to your local configuration.

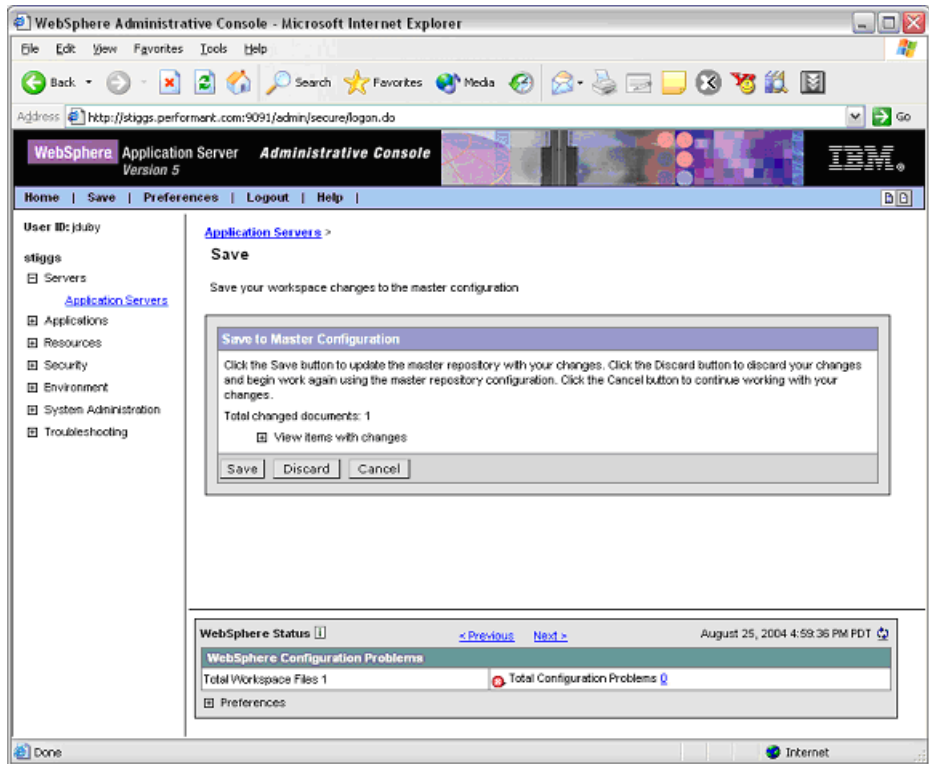


8 When your changes have been applied, the following window is displayed.

Click the **Save** menu option to save the changes to the master configuration.



- 9 The following window is displayed. Click **Save** in the Save to Master Configuration frame.



- 10 Restart the WebSphere application server. You do not need to restart the host machine for your application server.
- 11 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 you did not run the JRE Instrumenter or did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see "Running the JRE Instrumenter" on page 141.



## Configuring WebLogic Application Servers

Supported versions	Operating systems
6.1, 7.0, 8.1	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<Your_Dom_Name>.cmd
```

Replace **<Your\_Dom\_Name>** by the name of the script that starts your application.

- 2 For example, if your domain name is `petstore`, the path looks like this:

```
D:\bea\wlserver6.1\config\petstore\startPetStore.cmd
```

- 3 Create a back-up copy of the startup script prior to making any changes to the script.
- 4 Use your editor to open the startup script.
- 5 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 where `<probe_install_dir>` is to be replaced with the path to the directory where the Probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>/classes/boot
```

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\wlserver6.1/lib/weblogic.policy" weblogic.Server
```

- 6** Save the changes to the startup script.
- 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>.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. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

## 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 JRocket JVM, on page 183.

---

### 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\<Your_Dom_Name>\start<Your_Dom_Name>.cmd
```

Replace `<Your_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 back-up 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.

---

#### Windows:

```
set JAVA_OPTIONS=-Xbootclasspath/p:"C:\Program Files\Mercury Interactive\
common\JavaProbe\classes\boot" %JAVA_OPTIONS%
@rem Call WebLogic Server
call "C:\bea7\weblogic700\server\bin\startWLS.cmd"
```

#### UNIX:

```
export JAVA_OPTIONS=-Xbootclasspath/p:"/opt/Diagnostics/common/
JavaProbe/classes/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 restart 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>.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. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

## 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\\start<Your_Dom_Name>.cmd
```

Replace **<Your\_Dom\_Name>** is replaced by the name of the script that starts your application.

- 2 For example, if your domain name is `petstore`, the path looks like this:

```
D:\bea\weblogic81\config\petstore\startPetStore.cmd
```

- 3 Create a back-up copy of the startup script prior to making any changes to the script.
- 4 Use your editor to open the startup script.
- 5 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 where **<probe\_install\_dir>** is to be replaced with the path to the directory where the probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>\classes\boot
```

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\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
```

- 6** Save the changes to the startup script.
- 7** Restart the WebLogic application server. You do not need to restart the application server host machine.
- 8** 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 you did not run the JRE Instrumenter, or did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

**To configure a WebLogic 8.1 application server for the JRockit JVM:**

- 1** Locate the command file that the user uses to invoke 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 back-up 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.
  - ▶ Locate the line in the command file where the value of the **JAVA\_VENDOR** parameter is set.
  - ▶ 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.
  - ▶ Locate the line in the command file which begins as follows:

```
%JAVA_HOME%\bin\java %JAVA_VM% %JAVA_OPTIONS% .....
```

- ▶ 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.JRocketManagement
```

- ▶ 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 where `<probe_install_dir>` is to be replaced with the path to the directory where the probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>\classes\boot
```

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>.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. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

## 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 back-up 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 added to the "java-option value."

The following is an example of the **Xbootclasspath** parameter where **<probe\_install\_dir>** is to be replaced with the path to the directory where the probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>\classes\boot
```

Following 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" rmi="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" rmi="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>
```

Following 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 restart the host machine for the application server.
- 7 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 you did not run the JRE Instrumenter or did not enter the Xbootclasspath correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

## Configuring the JBoss Application Server

Supported versions	Operating systems
3.0.4	Windows 2000 Server
3.2.1	Linux Debian

This section explains how to configure the JBoss 3.2.1 application server.

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 back-up 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 where **<probe\_install\_dir>** is to be replaced with the path to the directory where the probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>\classes\boot
```

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:"C:\Program Files\Mercury Interactive\common\
JavaProbe\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 machine.
- 7** 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 you did not run the JRE Instrumenter or did not enter the **Xbootclasspath** correctly. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

## Configuring the SAP NetWeaver Application Server

Supported versions	Operating systems
3.0.4	Windows 2000 Server
3.2.1	Linux Debian

The following instructions describe how to configure the SAP NetWeaver application server so that your applications can be monitored by the J2EE Probe.

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 J2EE Probe. You should always create a backup of files before making any changes.

---

### To configure a SAP NetWeaver application server:

- 1** Run JRE Instrumenter as described in Chapter 9, “Running the JRE Instrumenter.”
- 2** Add the JVM that runs the NetWeaver application server. (See instructions for adding a JVM to the Available JVM List in “Running the JRE Instrumenter” on page 142.)
- 3** Instrument the JVM. (For instructions see “Running the JRE Instrumenter” on page 142.) The JRE Instrumenter provides the `Xbootclass` parameter. This parameter must be added to the NetWeaver Configtool’s JVM parameters window as described in the next step.



- 4** Run the NetWeaver application server configuration tool. The configuration tool is called **configtool.bat** and it is located in the **usr\sap\j2e\jc00\j2ee\configtool** directory.
- 5** 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.
- 6** Save your changes and exit the configuration tool.
- 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>.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. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

---

**Note:** If you detect capturing problems, 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
```

---

## Configuring a Generic Application Server

---

**Note:** You should only use the instructions for the 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. If this is the 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 back-up 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, where **<probe\_install\_dir>** is the path to the directory where the Probe was installed:

```
-Xbootclasspath/p:<probe_install_dir>\classes/boot
```

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:"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\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>.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. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

**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.

---

## Configuring the Probes for Multiple Application Server 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 a couple of options; you may install one Probe for each JVM on a host machine 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:

- ▶ Establishing communication between the JVM and the Probe
- ▶ Identification of the Probe by the JVM
- ▶ Instrumenting the JVM to instruct the Probe about the performance metrics it is to monitor

#### To configure a J2EE Probe to work with multiple JVMs:

- 1** When a Probe is being used to monitor multiple JVMs, the JRE Instrumenter must be run once for each JVM to enable the Probe to monitor the events of the applications that are running on the JVM.

If you have not already run the JRE Instrumenter for each of the JVMs that the Probe will be working with, do so now. See “Running the JRE Instrumenter” on page 141 for the instructions.

---

**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 the 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. See the following 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 50 JVMs. The range must be large enough so that there is are two available ports for each of the Probes.

---

**Note:** If a firewall separates your Probe and Mediator from the other Diagnostics components, you must configure the firewall to allow communications using the ports in the range that you specify. See "Configuring Diagnostics Components to Work with a Firewall" on page 399 for more information.

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.

---

- ▶ Locate the dispatcher.properties file in the folder `<probe_install_dir>/javaprobe/etc`.
- ▶ 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:

```
webserver.port=35000
```

The maximum port in the port number range is set using the following property:

```
webserver.maxPort=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 names 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 (note that the startup script must be entered on one line, without any breaks):

```
"%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
```

- 4 Specify the points file that LoadRunner / Performance Center will use. You may want to specify that an alternate 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 some custom hooks to 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 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 141.

- 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 Set the "id" property to a name 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 will be created in the `<probe_install_dir>/javaprobe/log` directory where the log messages for the Probe will be 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.

---

### **Configuring Multiple JVMs to Use a Single Probe for Deep Diagnostics**

When a Probe is being used by multiple JVMs, each instance of the application server must identify with an application definition in Deep Diagnostics. This is accomplished by defining the application definition using the Deep Diagnostics Console. The application definition associates the port number that the Probe will use to communicate with Deep Diagnostics and the host name of the computer on which the Probe is installed.

---

**Note:** This setup is the same if the application server instances are sharing a common JRE (for example, when multiple WebLogic JVM instances are using a shared file system).

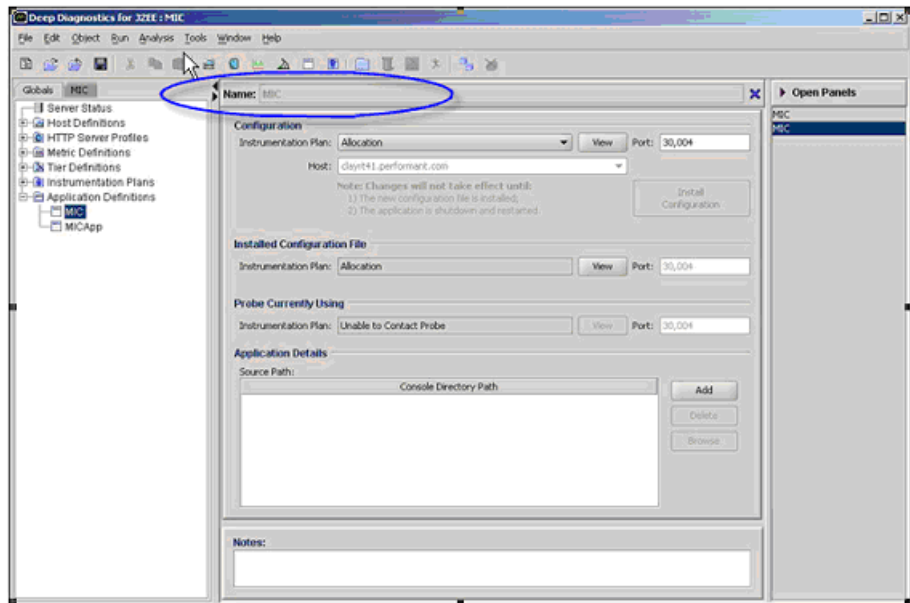
---



**To configure Deep Diagnostics and the JVM:**

- 1 Create a new application definition, or access an existing application definition from the Deep Diagnostics console.
  - Access the Application Definition window by selecting an application definition from the tree view on the Globals tab.
  - Enter the name for the Application Definition in the Name field.
- 2 Explicitly specify the port that the Probe is to use for communicating with the Deep Diagnostics server.

To specify the port, enter the number of the port that the Probe will use to communicate with Deep Diagnostics in the Port field of the Configuration section of the Application Definition window as shown below. Deep Diagnostics updates the port property in the points file.



- 3 Instrument each JVM that is being monitored by the Probe.
  - For each JVM version that is being monitored by the Probe:
    - Run the JRE Instrumenter. For details on running the JRE Instrumenter, see “Running the JRE Instrumenter” on page 141.

- Copy the `<probe_install_dir> classes/boot` directory and rename the directory to a name that indicates the JVM version that you ran the JRE Instrumenter for.
- For example, if you just ran the JRE Instrumenter for the WebLogic JVM you may want to rename the boot directory to:

`<probe_install_dir>\classes\bootwls`

- ▶ Add the `Xbootclasspath` parameter to the Java command line that starts the application server. Make sure that the `Xbootclasspath` parameter points to the `<probe_install_dir> classes/boot` version that you created for the application server type that is appropriate for the startup script that you are modifying.

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 where `<probe_install_dir>` is to be replaced with the path to the directory where the Probe was installed. Note that the boot directory is the copy of the directory that was created for the JVM.

```
-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"
```

---

**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 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
```

The following is an example of a WebLogic startup script after adding the Xbootclasspath parameter (note that the startup script must be entered on one line, without any breaks):

```
"%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%"
-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"
```

- 4 Point each of the application server instances to the Probe directory, and assign a unique Probe Identifier that will be used by the application server to reference the Probe. To define the unique Probe Identifier for each application server, add the following system property to the startup script for each instance, and to the JavaProbe/etc/probe.properties file:

```
-Dprobe.id="<Unique_DD_Probe_Name>"
```

Following is an example of a WebLogic startup script before adding the probe.id parameter:

---

**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.

---

```
"%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\Mer-
cury Interactive\Diagnostics for J2EE\Probe\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
```

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_DD_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
```

The system creates a separate log file for each probe name, so the log messages generated by the Probe for a particular application server appear in its own log file.

# Part IV

---

## Using Mercury Diagnostics for J2EE & .NET



# 12

---

## Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1

This chapter describes how to configure LoadRunner 8.1 so that Mercury Diagnostics for J2EE & .NET can be enabled for use in a load test.

This chapter includes the following sections:

- About Setting Up Diagnostics for J2EE & .NET
- Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1
- Configuring LoadRunner Scenarios to use Diagnostics for J2EE & .NET

---

**Note:** For information about configuring Performance Center 8.1 for use with Diagnostics for J2EE & .NET see Chapter 13, “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1.”

---

## About Setting Up Diagnostics for J2EE & .NET

LoadRunner and Diagnostics for J2EE & .NET are integrated products that have been designed to work together to provide you with the information that helps you to understand and improve the performance characteristics of your applications. However, you must provide LoadRunner with some information about your Diagnostics configuration to enable the product integration.

Before you can use Diagnostics for J2EE & .NET with LoadRunner 8.1, you must provide LoadRunner with the information that it needs so that it can communicate with the Diagnostics components.

Before you can gather Diagnostics metrics in a particular load test, you must select the LAN and probes that will be involved in the test and provide instructions for the scope and breadth of the monitoring that you would like Diagnostics to perform.



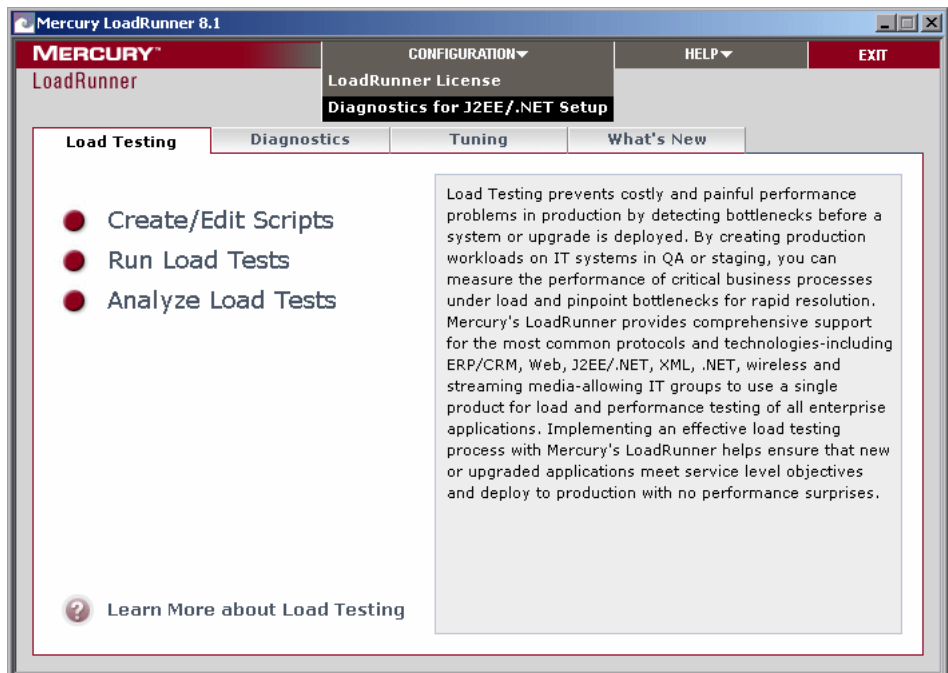
## Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1

The first time you use LoadRunner 8.1 to capture J2EE or .NET diagnostics data, you must tell LoadRunner the machine on which the Diagnostics Commander is running, and the port the Commander is using for communication with LoadRunner.

You must update this information if you want to integrate with a different Diagnostics Commander, or if you change the port the Commander is using.

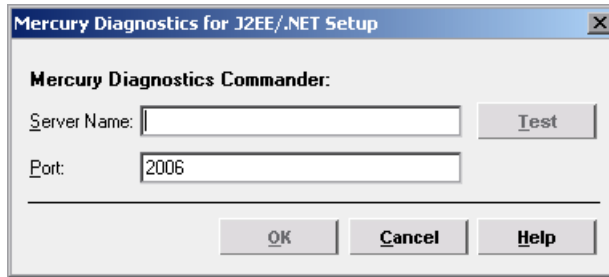
**To update the LoadRunner configuration settings for Diagnostics for J2EE & .NET:**

- 1 From the Start menu, choose **Programs > Mercury LoadRunner > LoadRunner** to open the Mercury LoadRunner launcher window.



- 2 From the Mercury LoadRunner launcher window menu, choose **Configuration > Diagnostics for J2EE & .NET Setup**.

- 3 The Diagnostics for J2EE/.NET Setup screen is displayed.



Enter the **Server Name** and **Port** for the Diagnostics Commander in the text boxes provided.

You may click **Test** to make sure that you entered the correct information for the Diagnostics Commander and that there is connectivity between the Diagnostics Commander and LoadRunner.

- 4 When you are satisfied with the information that you provided, click **OK** to finish the configuration process.

## Configuring LoadRunner Scenarios to use Diagnostics for J2EE & .NET

When you want to capture Diagnostics metrics in a load test, you need to configure the Diagnostics parameters for the scenario to enable Diagnostics and to let LoadRunner know which LAN and which Probes will be included in the load test scenario.

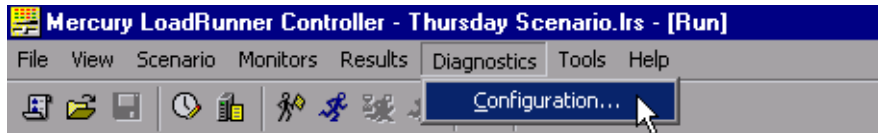
---

**Note:** You must configure the Diagnostics settings each time you run a load test unless you have saved a scenario with the Diagnostics settings already configured.

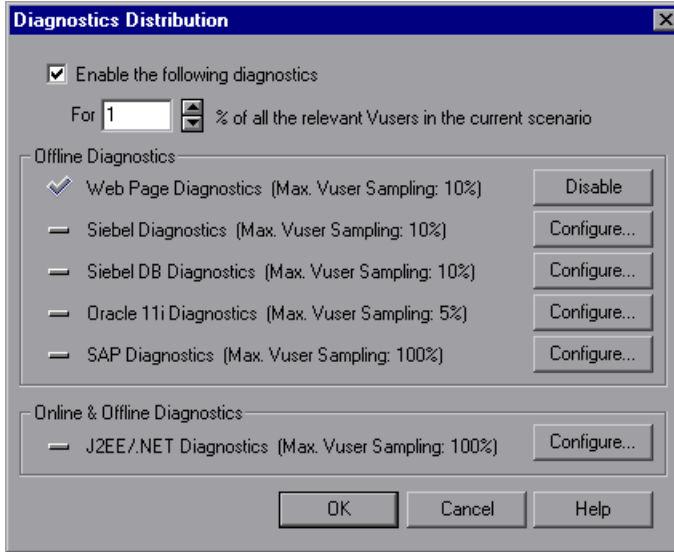
---

**To configure the diagnostics parameters for a load test scenario:**

- 1** Make sure that the application server has been started before configuring your scenario for Diagnostics from the LoadRunner Controller.
- 2** From the Start menu, choose **Programs > Mercury LoadRunner > Applications > Controller** to launch the Mercury LoadRunner Controller.
- 3** From the LoadRunner Controller menu bar, select **Diagnostics > Configuration**.



The Diagnostics Distribution dialog box opens.



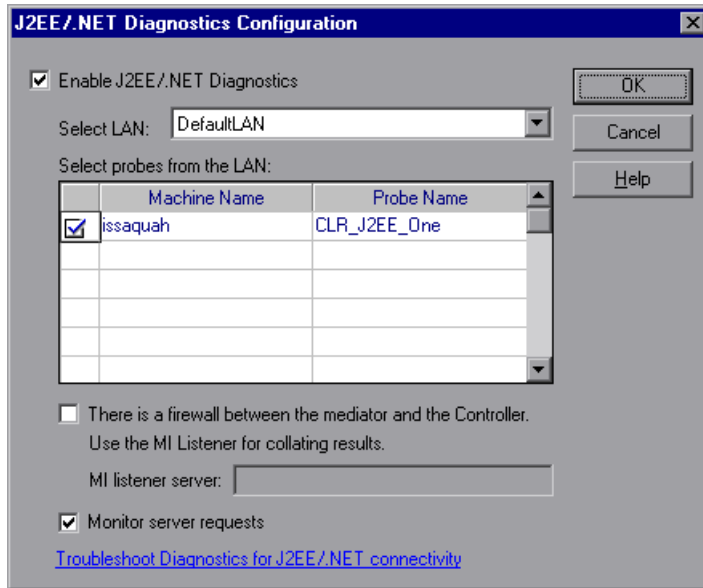
- 4** Enable Diagnostics by selecting the check box, **Enable the following diagnostics**.
- 5** Set the percentage of Vusers to participate in the monitoring. The maximum percentage of J2EE Vusers on which breakdown can be performed is 100%.

---

**Note:** If you have enabled other types of diagnostics, the percentage of Vuser participation cannot exceed the maximum of any of the selected diagnostics types.

---

- 6 In the Online & Offline Diagnostics section of the Diagnostics Distribution dialog, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.




---

**Note:** This dialog box is read-only while a scenario or session step is running.

---

- 7 Enable J2EE/.NET Diagnostics by checking the **Enable J2EE/.NET Diagnostics** box.
- 8 Choose the LAN that will be involved in the load test scenario from the **Select LAN** drop-down list. The selected LAN must contain the probes which were installed on the application servers you want to monitor.

---

**Warning:** The next time you open the J2EE/.NET Diagnostics Configuration dialog box, the LAN that you see may be a different one. This is because the dialog box usually shows the default LAN. Make sure you choose the LAN you chose before. Otherwise, clicking **OK** will disable the Probes that you selected.

---

When you choose a LAN, the list of Probes in the **Select Probes from the LAN** list box is updated with the Probes that are in the selected LAN.

---

**Note:** All of the Probes in a load test must be from the same LAN.

---

- 9** Choose the Probes that are to be monitored during the load test from **Select probes from the LAN** list box. Select the check box adjacent to each probe that you want to monitor. A check mark appears, indicating that the probe is enabled.

To disable a probe for the duration of a scenario or session, toggle the check box by selecting it again.

---

**Note:** You must enable at least one probe.

---

- 10** If the mediators are located behind a firewall, check **There is a firewall between the mediator and the Controller**, and enter the name of the MI listener server in the **MI listener server** field. For more information see Appendix F, “Configuring Diagnostics Components to Work with a Firewall.”
- 11** To capture a percentage of server requests which occur outside the context of any Virtual User transaction select the **Monitor server requests** check box.

The server requests will be captured at the same percentage as was selected for the percentage of Vusers on Diagnostics Configuration dialog.

**Note:** Enabling this functionality imposes an additional overhead on the Probe.

---

The benefit of enabling this functionality is that calls into a “back-end” VM can be captured even in the case where:

- ▶ The Probe is not capturing RMI calls
  - ▶ RMI calls cannot be captured (perhaps because an unsupported application container is being used)
  - ▶ The application uses some other mechanism for communications between multiple VMs.
- 12** Investigate any issues that you have with the connections between the Diagnostics components by clicking the **Troubleshoot** Diagnostics for J2EE & .NET **connectivity** hyper link. This will open the System Health Monitor in a new browser window as shown below.

For details of how to use the System Health Monitor, see Appendix A, “Using the System Health Monitor.”

- 13** When you have finished making your selections, click **OK** to accept your changes and close the J2EE/.NET Diagnostics Configuration dialog. Click **OK** on the Diagnostics Distribution dialog to complete the configuration.





# 13

---

## Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1

This chapter describes how to configure Performance Center 8.1 so that Mercury Diagnostics for J2EE & .NET can be enabled for use in a load test.

This chapter includes the following sections:

- ▶ About Setting Up Diagnostics for J2EE & .NET
- ▶ Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1
- ▶ Configuring Performance Center Load Tests to use Diagnostics for J2EE & .NET

---

**Note:** For information about configuring LoadRunner 8.1 for use with Diagnostics for J2EE & .NET see Chapter 12, “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1.”

---

## About Setting Up Diagnostics for J2EE & .NET

Performance Center and Diagnostics for J2EE & .NET are integrated products that have been designed to work together to provide you with the information that helps you to understand and improve the performance characteristics of your applications. However, you must provide Performance Center with some information about your Diagnostics configuration to enable the product integration.

Before you can gather Diagnostics metrics in a particular load test, you must select the LAN and probes that will be involved in the test and provide instructions for the scope and breadth of the monitoring that you would like Diagnostics to perform.

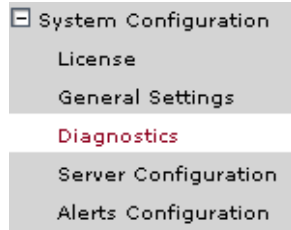
## Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1

The first time you use Performance Center 8.1 to capture J2EE or .NET diagnostics data, you need to specify the machine on which the Diagnostics Commander is running, and the port that the Commander is using for communication with Performance Center.

You provide this information on the Diagnostics page of the Performance Center Administration Site. You must update this information if you want to integrate with a different Diagnostics Commander, or if you change the port the Commander is using.

**To update the Performance Center configuration settings for Diagnostics for J2EE & .NET:**

- 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 **Server Name** and **Port Number** for the Diagnostics Commander.

**Setting J2EE/.Net Environment**

---

J2EE/.Net Diagnostics Server Name

Server Name:

Port Number:

- 4 Click **Save** to update the information.

## Configuring Performance Center Load Tests to use Diagnostics for J2EE & .NET

When you want to capture Diagnostics metrics in a load test, you need to configure the Diagnostics parameters for the load Test to enable Diagnostics and you need to specify which LAN and which Probes will be included in the load test. You provide this information on the Load Test Configuration page of the Performance Center User Site.

**Note:** You must configure the Diagnostics settings each time you run a load test unless you have saved a load test with the Diagnostics settings already configured.

To configure the Diagnostic parameters for a load test:

- 1 Log on to the Performance Center User Site.
- 2 From the left navigation menu, choose **Load Tests > Create/Edit** to open the Load Test Configuration page.

The screenshot shows the Mercury Performance Center interface. The top navigation bar includes the Mercury logo and the text 'Performance Center'. Below this, the user is logged in as 'Admin' and the project is 'Default'. The main heading is 'Load Tests'. A 'New Load Test' button is present. Below the button is a table listing existing load tests.

Name ( # of Runs )	Last Modified Date
<a href="#">asd (0)</a>	11-Jul-2005
<a href="#">666 (0)</a>	11-Jul-2005
<a href="#">sched1 (0)</a>	4-Jul-2005
<a href="#">44 (0)</a>	28-Jun-2005
<a href="#">+ dashboard (2)</a>	28-Jun-2005
<a href="#">+ empty transaction test (2)</a>	4-Jul-2005

- 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 select the **Enable Diagnostics** check box to enable transaction breakdown monitoring.

General Design Groups Scheduler Monitors **Diagnostics**

Select the Mercury Diagnostics tools that you want to use to identify and pinpoint performance problems in your Web, ERP/CRM, and J2EE/.NET applications.

Enable diagnostics

Perform diagnostics breakdown on  % of all relevant Vusers participating in the current Load Test

**Offline Diagnostics**

- Web Page Breakdown (Max. Vuser Sampling: 10%)
- Siebel Diagnostics (Max. Vuser Sampling: 10%)
- Siebel DB Diagnostics (Max. Vuser Sampling: 10%)
- Oracle 11i Diagnostics (Max. Vuser Sampling: 5%)
- SAP Diagnostics (Max. Vuser Sampling: 100%)

**Offline & Online Diagnostics**

- J2EE/.NET Diagnostics (Max. Vuser Sampling: 100%)

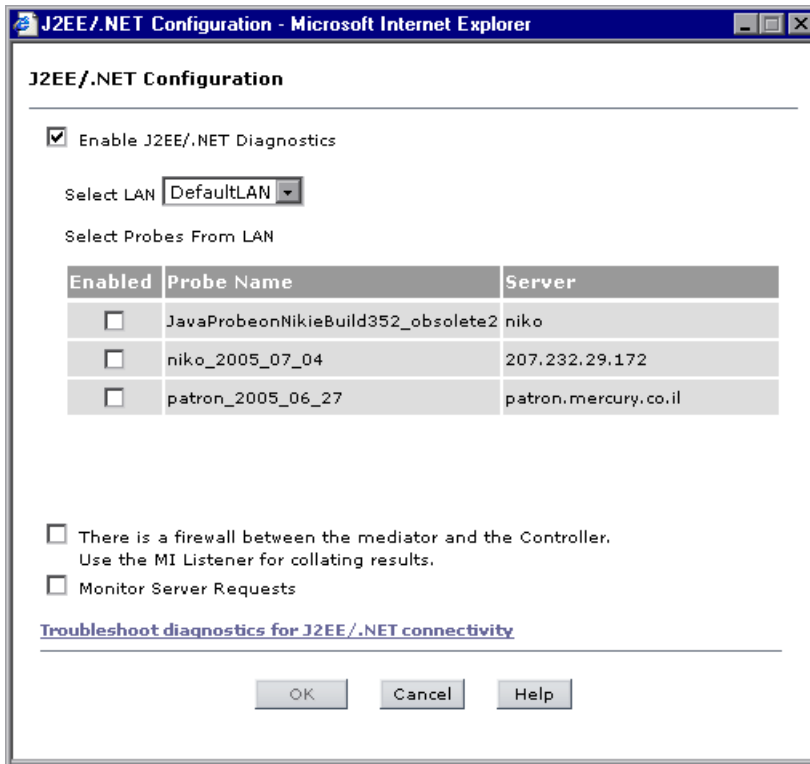
- 5 Set the percentage of Vusers to participate in the monitoring in the **% of all relevant Vusers** box. The maximum percentage of J2EE/.NET Vusers on which breakdown can be performed is 100%.

---

**Note:** If you have enabled other types of diagnostics, the percentage of Vuser participation cannot exceed the maximum of any of the selected diagnostics types.

---

- 6 In the **Offline and Online Diagnostics** section, click **Configure** to open the J2EE/.NET Diagnostics Configuration dialog box.



---

**Note:** This dialog box is read-only while a load test is running.

---

- 7 Select the **Enable J2EE/.NET Diagnostics** check box to enable all the other fields in the dialog box.
- 8 In the **Select LAN** box, select the LAN that will be involved in the load test. The selected LAN must contain the probes which were installed on the application servers you want to monitor.

**Warning:** The next time you open the J2EE/.NET Diagnostics Configuration dialog box, the LAN that you see may be a different one. This is because the dialog box usually shows the default LAN. Make sure you choose the LAN you chose before. Otherwise, clicking **OK** will disable the Probes that you selected.

---

When you choose a LAN, the list of Probes in the **Select Probes from the LAN** list box is updated with the Probes that are in the selected LAN.

---

**Note:** All of the Probes in a load test must be from the same LAN.

---

- 9** In the **Select probes from LAN** section, select the check box adjacent to each probe that you want to monitor. A check mark appears, indicating that the probe is enabled.

To disable a probe for the duration of a load test, clear the check box.

---

**Note:** You must enable at least one probe in order to be able to save the J2EE/.NET configuration.

---

- 10** If the mediators are located behind a firewall, check **There is a firewall between the mediator and the Controller**.

If you are monitoring over a firewall, make sure that you have installed an MI Listener on a machine outside of the firewall, and that you specify the IP address of the MI Listener machine on the Performance Center Administration Site, on the **General Settings** page, in the **Firewall Diagnostics Communicator** field. For installation instructions, refer to the *Mercury Performance Center System Configuration and Installation Guide*.

For more information about configuring Diagnostics to work with a firewall see Appendix F, “Configuring Diagnostics Components to Work with a Firewall.”

- 11** To capture a percentage of server requests which occur outside the context of any Virtual User transaction select the **Monitor server requests** check box.

The server requests will be captured at the same percentage as was selected for the percentage of Vusers on Diagnostics Configuration dialog.

---

**Note:** Enabling this functionality imposes an additional overhead on the Probe.

---

The benefit of enabling this functionality is that calls into a “back-end” VM can be captured even in the case where:

- ▶ The Probe is not capturing RMI calls
  - ▶ RMI calls cannot be captured (perhaps because an unsupported application container is being used)
  - ▶ The application uses some other mechanism for communications between multiple VMs.
- 12** Investigate any issues that you have with the connections between the Diagnostics components by clicking the **Troubleshoot** Diagnostics for J2EE & .NET **connectivity** hyper link. For details of how to use the System Health Monitor, see Appendix A, “Using the System Health Monitor.”
  - 13** When you have finished making your selections, click **OK** to accept your changes and to close the J2EE/.NET Diagnostics Configuration dialog.
  - 14** Click **Save** in the Diagnostics tab to save and to validate your settings and to complete the configuration.



# 14

---

## Introducing Diagnostics for J2EE & .NET Screens

This chapter introduces the screens that are used to present the Diagnostics for J2EE & .NET performance metrics. A high level overview of the screens and instructions for using them are provided. The next chapter provides a detailed description of each screen along with the graphs and tables that appear on the screens. See “Using the Diagnostics for J2EE & .NET Screens” on page 239 for more information.

The following topics are included in this chapter:

- ▶ Viewing Diagnostics Data from the Controller
- ▶ Introducing the Diagnostics Screens
- ▶ Drilling Down Into the Diagnostics Metrics
- ▶ Using the Diagnostics Navigation and Display Controls

### Viewing Diagnostics Data from the Controller

Once you have configured LoadRunner / Performance Center and Diagnostics for J2EE & .NET as described in Chapter 12, “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1” and in Chapter 13, “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1,” and you have initiated a load test, you may view the Diagnostics metrics as the load test executes.

**To view the Diagnostics metrics in LoadRunner:**

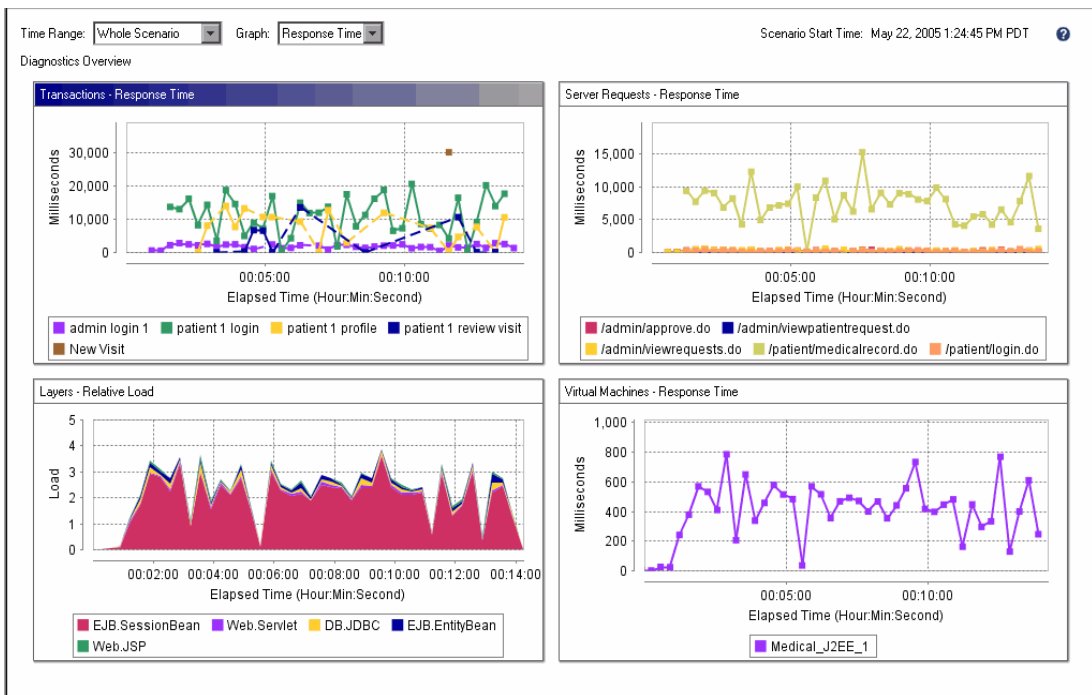
Click the **Diagnostics for J2EE/.NET** tab at the bottom of the Mercury LoadRunner Controller window to open the Diagnostics Overview screen.

**To view the Diagnostics metrics in Performance Center:**

Click the **View Diagnostics** tab on the Load Test Run page to open the Diagnostics Overview screen.

The Diagnostics Overview screen is displayed as shown below. From this screen, you can navigate and drill down into all of the performance metrics provided by Diagnostics.

A high level description of this screen and the other Diagnostics screens are provided in the remainder of this chapter. For a more detailed description of the screens see “Using the Diagnostics for J2EE & .NET Screens” on page 239.



## Introducing the Diagnostics Screens

The Diagnostics performance metrics are presented on a series of screens in both graphical and tabular formats so that you can analyze a particular aspect of your application's performance and drill down into the details of the system behavior that is contributing to the observed performance.

---

**Note:** Response time on the Diagnostics screens refers to the server response time as measured on the server side. Response time on the LoadRunner / Performance Center screens refers to the server response time as measured on the client side. For this reason, comparison of response times between a Diagnostics screen and a LoadRunner / Performance Center screen may not match.

The transaction time that is displayed on the Diagnostics screens includes only the time that the transaction spends in the J2EE/.NET server. This means that it does not include the user's think time in the midst of the transaction.

---

The Diagnostics screens are:

► Diagnostics Overview

The top five most notable instances for Transactions, Server Requests, Layers, and Virtual Machines data are displayed on a single screen to provide a broad overview of the performance characteristics of your application. The most notable instance would be the 5 slowest response times when you are viewing the response times graphs. The 5 highest counts would be the most notable when viewing the count graphs.

For more information on this screen see "Analyzing Performance Using the Diagnostics Overview Screen" on page 240.

► Transactions

The response times or the requests per second for the captured transactions across all Virtual Machines are displayed in graphical and tabular formats. For more information on this screen see "Analyzing Performance with the Transaction Trends Screen" on page 244.

► Server Requests

The response times or the requests per second for the captured server request calls are displayed in graphical and tabular formats. For more information on this screen see “Analyzing Performance with the Server Request Screens” on page 249.

► Layers

The average response time for the captured layers are displayed in graphical and tabular formats. For more information on this screen see “Analyzing Performance with the Layer Screens” on page 255.

► Virtual Machines

The average response time or the requests per second for all server requests on the selected virtual machines are displayed in graphical and tabular formats. For more information on this screen see “Analyzing Performance with the Virtual Machines Screen” on page 261.

► Aggregate Profile

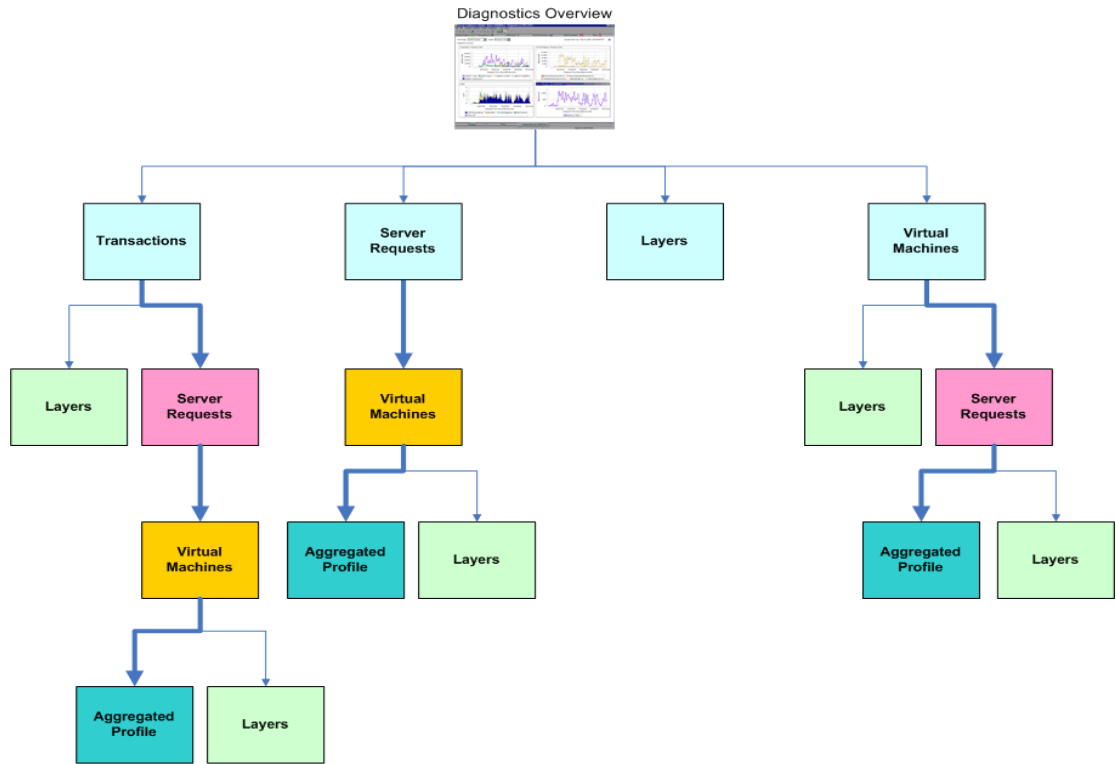
The method call profile behind the processing metric that you are viewing on one of the other screens is presented in a graphical and in a call tree format. For more information on this screen see “Analyzing Performance with the Aggregate Profile Screen” on page 267.

## Drilling Down Into the Diagnostics Metrics

When you analyze the performance of your application, you begin by using the Diagnostics graphs found on the Diagnostics Overview screen. Mercury Diagnostics allows you to drill down into the information presented on the screens to uncover the root cause of the application performance that is being portrayed in the higher level graphs.

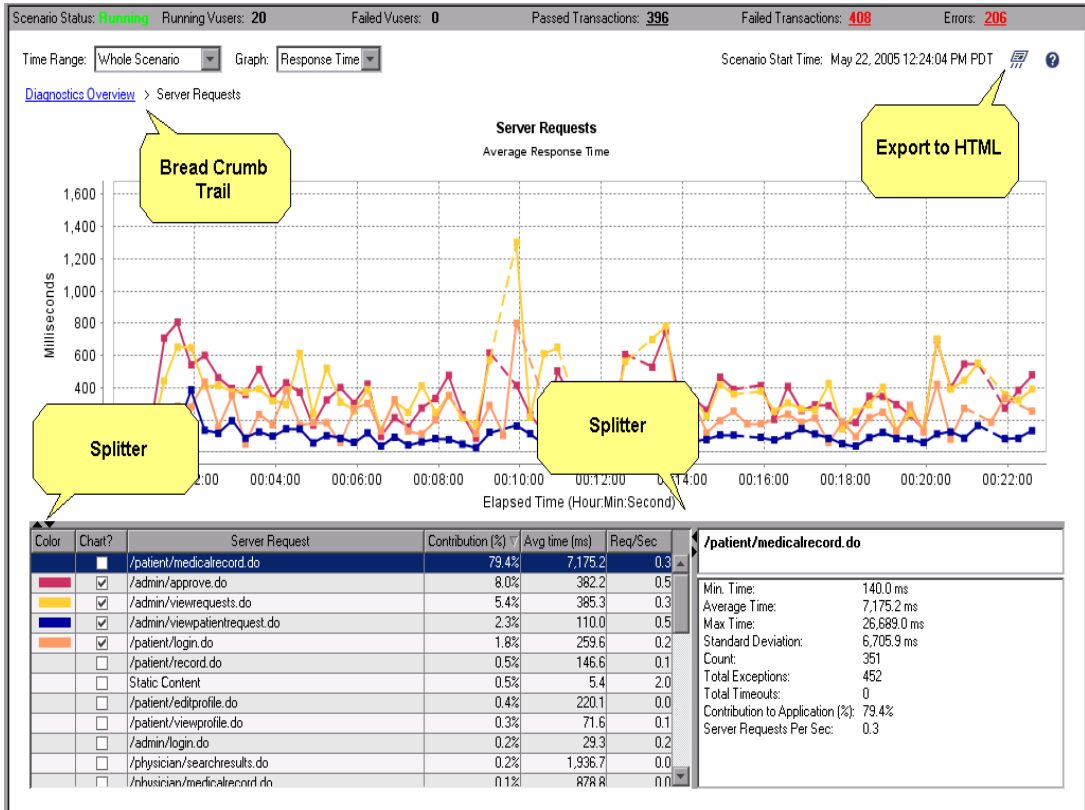
The following diagram illustrates the paths that you can use as you drill down into the metrics. As you drill down, the data displayed on the lower level screens is filtered based upon each of the preceding drill down levels.

The default path that will be taken when you double click on an item in the table is shown with a bold line.



## Using the Diagnostics Navigation and Display Controls

This section provides an overview of the common navigation and display controls that appear on the Diagnostics screens. More details about these controls have been provided in the description of each screen when the functionality varies significantly from what is described in this section.



### Dashed Lines Verses Solid Lines In Graph

When there is a solid line between two data point on the graph, the two data points were consecutive queries. This means that line illustrates the actual trend of the data between those two points.

When there is a dashed line between two data points on the graph, the two data points were not consecutive queries. This means that there was a query in between the two data points that returned no measurable data and

therefore the actual line between the two graphed points illustrates an approximation of the actual trend.

### **Adjusting the Reported Time Range**

You may control the breadth of time for which performance data is displayed in many of the graphs, tables and charts by selecting the desired length of time from the **Time Range** drop down list. The choices in the drop down list are:

- ▶ Previous 5 Minutes
- ▶ Previous 30 Minutes
- ▶ Previous 4 Hours
- ▶ Whole Scenario (This is the default selection.)

### **Adjusting the Graphed Metric**

You may control which performance metrics will be graphed on many of the Diagnostics screens by using the **Graph** drop down list. Once you make a selection from this drop down list, the selection will be applied to all subsequent screens until another selection is made. The choices that are available in the drop down list are:

- ▶ Response Time - the graphed metrics will depict the response time between the time when a call was made and when the processing for the call was ended.
- ▶ Count - the graphed metrics will depict the average number of calls per second.

### **Screen Context - Bread Crumb Trail**

The “bread crumb” trail at the top of many of the diagnostics screens provides a convenient reminder of the context for the information that is presented on the screen. It also provides a handy method of navigation when you want to retrace your steps in your performance analysis.

Clicking on a level in the bread crumb trail will take you back to the screen that displays the metrics for the selected higher level. The selections that you made from the drop-down lists in the lower level screens to change the time period or type of graph and changes that you make to the table including the sort order and the rows that are to appear in the graph will be retained.

---

**Note:** When you return to the Diagnostics Overview screen the drop-down list selections are retained but the changes that you made to the sort order for the table or the rows to display in the graph will be lost.

---

The last level displayed in the bread crumb is not a link because it represents the current diagnostics screen that has been displayed. The bread crumb has been highlighted with a callout in the previous screen image.

### Exporting Metrics to HTML Report



You may capture the processing metrics displayed in the graph and table on a diagnostics screen by clicking the icon on the top right-hand side of the screen. The information displayed on the screen is exported to an HTML report file. The HTML reports can be viewed in your browser and can be used to share metrics for interesting or perplexing system performance with others.

---

**Note:** There is not HTML Report Export from the Diagnostics Overview Screen.

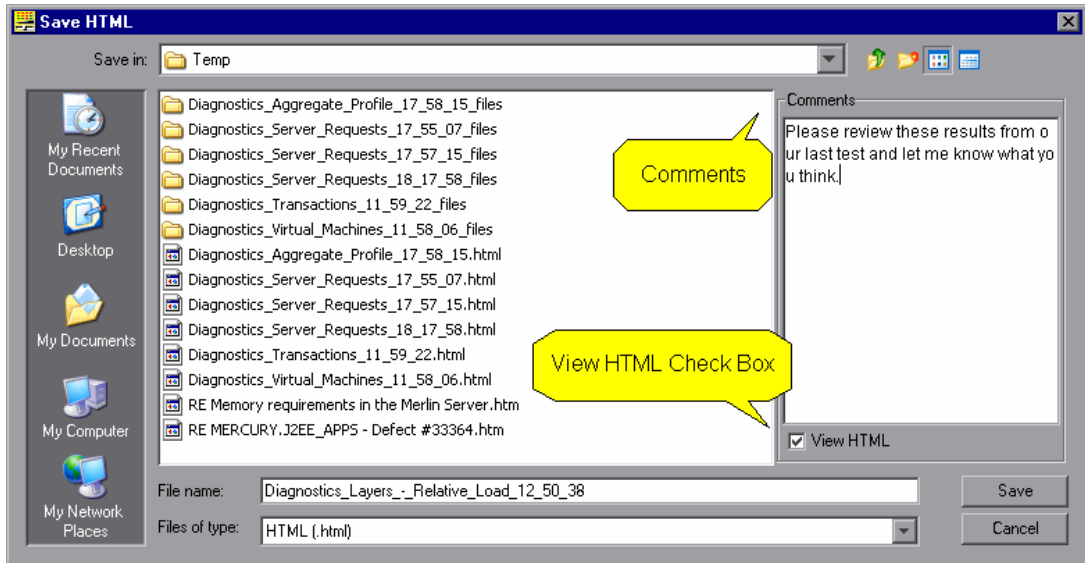
---

#### To export to an HTML Report:

- 1 Click the export icon. The Save HTML screen is displayed.
- 2 Enter the file name that you want the exported file to be saved as into the **File name** field.
- 3 Enter any comments that you would like to have appear on the report into the Comments text box.



- 4 If you would like to view the HTML file, select the **View HTML** check box. This will cause the HTML file that is being saved to be displayed in your browser after it has been saved.



### Definitions for the Columns in the Diagnostics Table

The table at the bottom of the Diagnostics screens contains a list of the items for which processing metrics are being reported. On the Server Requests screen the table lists server requests and on the Transactions screen the table lists transactions.

The table lists the items that apply to the context displayed in the bread crumbs. The metrics reported in the table are filtered based upon the time period specified in the **Time Range** drop-down.

The table is made up of the following columns. Some of the screens do not display all of these columns.

- **Color** - When the Chart check box for the item has been selected so that a trend line for the item will be included in the graph this column contains a colored block. The color of the block is the same color as the trend line for the item.

- ▶ **Chart** - A check box that you may toggle to indicate whether the item should be included in the graph. When the box is checked, a trend line for the item will appear in the graph.
- ▶ **Item** - The name of the item that is being reported in the table. On the Virtual Machines screen this column has a heading of “Virtual Machine” and contains the name of the virtual machines for which metrics are reported.
- ▶ **Contribution** - The amount of time and percentage of the parent’s total time that the selected row contributed to the item. The time is reported in milliseconds.
- ▶ **Avg Time** - The average response time for the item across all instances. The time is reported in milliseconds.
- ▶ **Req /Sec** - The number of server requests that occurred each second for the selected time period.
- ▶ **Txn /Sec** - The number of transactions that occurred each second for the selected time period.

### **Sorting the Table**

When viewing the Response Time graph, the table is sorted in descending order by the Contribution column. When viewing the Count graph, the table is sorted in descending order by the Txn/Sec or Req/Sec column.

You may change the order in which information is displayed in the tables on the Diagnostics screens by clicking the column headers in the table. Diagnostics will sort the rows in the table in ascending order according to the values in the table column that you selected. Clicking the same column header again will toggle the sort order to descending order.

---

**Note:** The table on the Aggregate Profile screen cannot be sorted.

---

### **Adding and Removing Items From the Graph**

On many of the screens you may control which items from the table appear in the graph by toggling the selection in the Chart column of the table.

- ▶ To add an item to the graph select the check box in the **Chart** column for the row in the table. When you click anywhere on a row in the table that is not already included in the graph, it will be added to the graph as well.
- ▶ To remove an item from the graph deselect the check box in the **Chart** column for the row in the table.

### **Highlighting the Trend Line or Area For a Detail Line in the Table**

You can cause the trend line or area for a detail line in the table to stand out in a bold font by selecting the row in the table. The graph line for the previously selected row will be changed to a normal font and the trend line for the row that you selected will stand out with a bolder font.

### **Selecting the Trend Line in the Table for a Graph Line**

You can determine which detail line in the table is associated with a given graphed trend line by clicking on a data point on the trend line. When the data point is clicked, the focus in the table will be shifted to the row associated with the graphed line.

### **Viewing Additional Details in the Details Panel**

You may view additional information for the items listed in the table by clicking on the item to select the row in the table. The details for the item in the selected row will be displayed in the Details Panel in the bottom right corner of the screen.

---

**Note:** The fields that are displayed in the Details Panel will vary depending on the metrics that are being displayed on the screen and time range that you selected in the **Time Range** drop down.

---

**Note:** The time measurements may not correspond to the actual time for an event when the beginning or end of an event has not been captured.

---

### **Definitions for the Rows in the Details Panel**

The Details Panel lists the following information for the selected row of the Diagnostics Table.

- ▶ **Name** - The name of the item that was selected from the Diagnostics table and whose information is displayed in the Details Panel.
- ▶ **Min Time** - The elapsed time for the instance of the item selected from the diagnostics table that has the shortest duration of all of the aggregated instances. Time is measured from the start of the first captured point until the end of the last captured point.
- ▶ **Average Time** - The average (mean) elapsed time taken by an item. Average time is calculated by dividing the item's total time by the item's count.
- ▶ **Max Time** - The elapsed time for the instance of the item selected from the diagnostics table that has the longest duration of all of the aggregated instances. Time is measured from the start of the first captured point until the end of the last captured point.
- ▶ **Standard Deviation** - A measure of the dispersion of the total time taken by all of the instances of the selected item.
- ▶ **Count** - The total number of times an item was invoked.
- ▶ **Total Exceptions** - The number of exceptions that were captured for all of the aggregated instances of the selected item.
- ▶ **Total Timeouts** - The number of times that an instance of the selected item failed in an unknown manner.
- ▶ **Host Name** - The machine name that is the host for the virtual machine where the item is being executed.
- ▶ **Platform** - The platform where the application is running.

- ▶ **Contribution to Parent (Time)** - The amount of time that this item contributed to the parents total time.
- ▶ **Contribution to Parent (%)** - The percentage of the time that this item contributed to the parents total time.
- ▶ **Contribution to Application** - The percentage of time that this item contributed to the applications total time.
- ▶ **Server Requests per Sec** - The number of events per second for the selected time frame.
- ▶ **Transactions per Sec** - The number of transactions per second for the selected time frame.

### **Drilling Down into Rows in the Table**

Many of the Diagnostics screens allow you to drill down into the metrics presented in the tables to see the metrics for the underlying processing. There are two ways provided for you to drill down on the diagnostics screens.

---

**Note:** Not all screens have drill down navigation options.

---

- ▶ Right-click on a row in the table to display a pop-up menu with the available navigation options.
- ▶ Double click on a row in the table to drill down directly into the default lower level graph. This is a shortcut that allows you to drill down without stopping at the pop-up menu displayed when you right-click.

### **Zooming in on Sections of the Screen**

Splitter controls have been provided on most of the screens to allow you to minimize and maximize areas of the screen. See the arrows that have been highlighted with the square in the above example for an example of splitter controls. This makes it possible for you to highlight areas of the screen that are of interest to you by maximizing that part of the screen and minimizing other parts of the screen that are of less interest.



# 15

---

## Using the Diagnostics for J2EE & .NET Screens

This chapter provides a detailed description of the screens, graphs, tables, and charts that are used to present the diagnostic performance metrics for the application that is being analyzed. The presentation of each monitored metric is described along with the ways that you can drill down from the higher level information to reveal the specific part of your application that is contributing to the observed application performance. For a high level introduction to the Diagnostic screens see “Introducing Diagnostics for J2EE & .NET Screens” on page 225.

This chapter contains the following topics:

- Analyzing Performance Using the Diagnostics Overview Screen
- Analyzing Performance with the Transaction Trends Screen
- Analyzing Performance with the Server Request Screens
- Analyzing Performance with the Layer Screens
- Analyzing Performance with the Virtual Machines Screen
- Analyzing Performance with the Aggregate Profile Screen

## Analyzing Performance Using the Diagnostics Overview Screen

### Purpose of Diagnostics Overview Screen

The top five most notable instances for Transactions, Server Requests, Layers, and Virtual Machines data are displayed on a single screen to provide a broad overview of the performance characteristics of your application. The most notable instance are the 5 slowest response times when you are viewing the response times graphs. The 5 highest counts would be the most notable when viewing the count graphs. From the Diagnostics Overview screen, you can drill down into the details of the performance portrayed in the graphs along with the other less noteworthy instances of the performance metrics.

### Accessing the Diagnostics Overview Screen

**To access the Diagnostics Overview Screen from LoadRunner:**

Click the **Diagnostics for J2EE/.NET** tab at the bottom of the Mercury LoadRunner Controller window.

**To access the Diagnostics Overview Screen from Performance Center:**

Click the **View Diagnostics** tab on the Load Test Run page.

**To access the Diagnostics Overview Screen from the any of the other Diagnostics screens:**

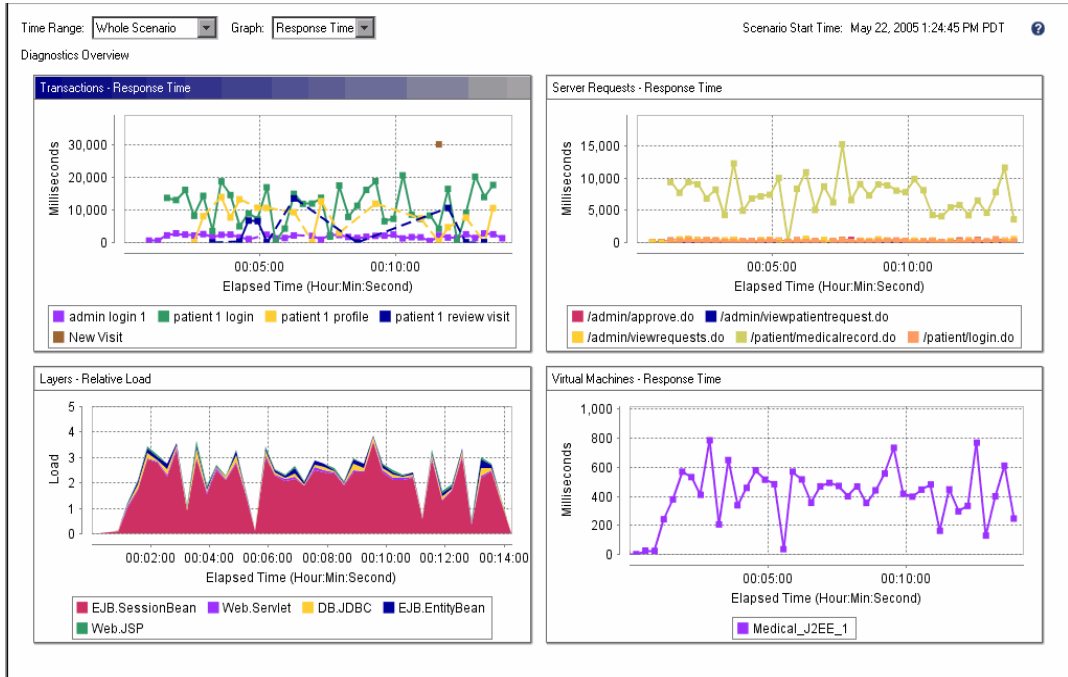
Click the **Diagnostics Overview** bread crumb at the top of the screen.

The Diagnostics Overview screen is displayed with the four graphs containing the information for the worst performing transactions, server requests, layers, and virtual machines.



## Description of Diagnostics Overview Screen

The Diagnostics Overview screen contains the Transactions graph, the Server Requests graph, the Layers graph and the Virtual Machines graph as shown in the following example.



### Transactions graph

This graph shows the transaction metrics for the five transactions with the longest response times. This is the same graph that is presented on the Transactions screen when it is first displayed. See “Transactions: Response Time Graph” on page 246 for more information.

### Drilling Down into the Transaction Metrics

You may drill down into the transaction information displayed on the Diagnostics Overview screen. To see more information on the transactions displayed in the Transactions graph, double click the graph. The Transactions screen is displayed where you can view detailed metrics for the transactions displayed on the Diagnostics Overview screen as well as the rest

of the transactions that were not displayed on the Overview screen. See “Analyzing Performance with the Transaction Trends Screen” on page 244 for more information.

### **Server Requests Graph**

This graph shows the server request metrics for the five server requests with the longest response times. This is the same graph that is displayed on the Server Requests screen. See “Server Requests: Response Time Graph” on page 253 for more information.

### **Drilling Down into the Server Request Metrics**

You may drill down into the server request information displayed on the Diagnostics Overview screen. To see more information on the server requests displayed in the Server Requests graph, double click the graph. The Server Requests screen will be displayed where you can view detailed metrics for the server requests displayed on the Diagnostics Overview screen as well as the rest of the Server Requests that were not displayed on the Overview screen. See “Analyzing Performance with the Server Request Screens” on page 249 for more information on these choices.

### **Layers Graph**

This graph shows the performance metrics for the captured layers with the worst response times. This is the same graph that is displayed on the Layers screen.

### **Drilling Down into the Layer Response Time Metrics**

You may drill down into the Layer performance metrics displayed on the Diagnostics Overview screen. To see more information on the performance metrics displayed in the Layers graph, double click the graph. The Layers screen is displayed where you can view detailed metrics for the Layer performance metrics displayed on the Diagnostics Overview screen as well as the rest of the VM that were not displayed on the Overview screen. See “Analyzing Performance with the Server Request Screens” on page 249 for more information on these choices.

## Virtual Machines graph

This graph shows the processing metrics for the five virtual machines with the worst response times. This is the same graph that is displayed on the Virtual Machines screen. See “Virtual Machines graph” on page 243 for more information.

## Drilling Down into the Virtual Machine Response Time Metrics

You may drill down into the VM information displayed on the Diagnostics Overview screen. To see more information on the performance metrics displayed in the Virtual Machines graph, double click the graph. The Virtual Machines screen will be displayed where you can view detailed metrics for the VM displayed on the Diagnostics Overview screen as well as the rest of the VM that were not displayed on the Overview screen. See “Analyzing Performance with the Virtual Machines Screen” on page 261 for more information on these choices.

## Refining the View In the Diagnostics Overview Screen

Use the following controls to adjust the amount of data and the type of data that is displayed on the Diagnostics Overview screen.

- Choose whether to graph the response times or the call counts for the transactions as described in “Adjusting the Graphed Metric” on page 231. This will impact the metrics that are displayed in the Transactions, Server Requests, and Virtual Machines graphs displayed on this screen as well as the graphs that are displayed on any screen that is displayed as a result of drilling down into these graphs.

The Layers graph will show the Relative Load no matter what is selected from the Graph drop-down list. There is no version of this graph that shows requests per second counts.

- Adjust the Time Range for which transaction metrics are displayed as described in “Viewing Additional Details in the Details Panel” on page 235. This will impact the metrics that are displayed in the graphs on this screen as well as any screens that are displayed as a result of drilling down into the content of the screen.

## Analyzing Performance with the Transaction Trends Screen

### Purpose of the Transactions Screen

This screen displays performance metrics for the captured transactions across all Virtual Machines. The graph on the screen can display either transaction response times or the execution counts for each transaction. The tables on the screen display the metrics for each transaction that was captured for the context that is being displayed along with additional information that help you to understand and improve the performance of your applications.

### Accessing the Transactions Screen

**To get to the Transactions Screen from the Diagnostics Overview Screen:**

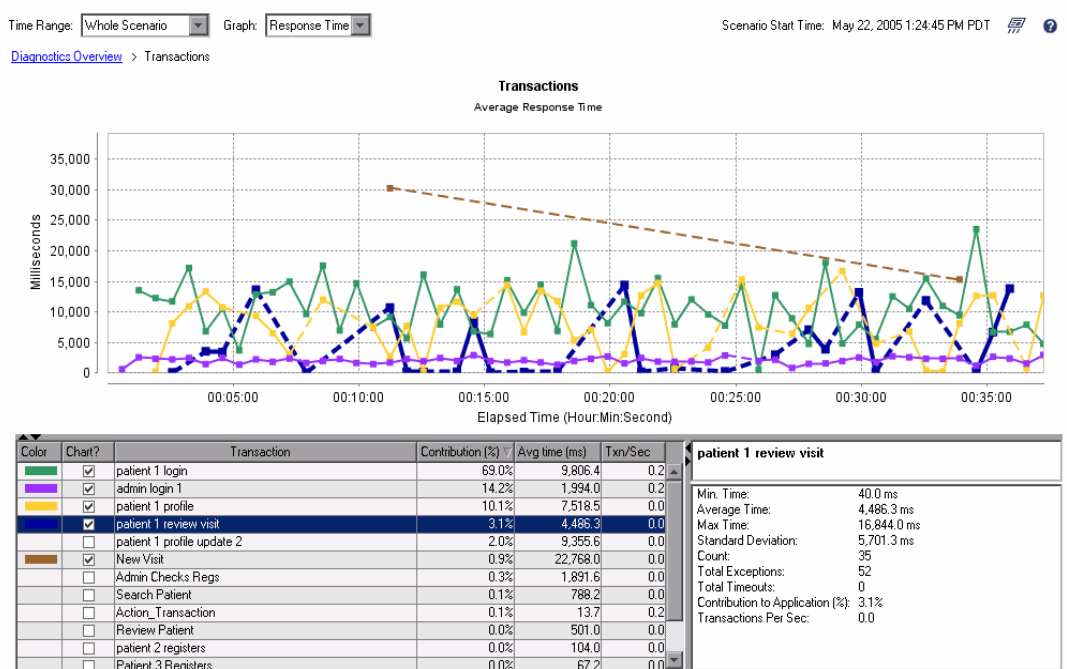
- 1** Double click the Transactions graph.
- 2** The Transaction Trends screen is displayed. The graph should contain the same transactions that were displayed on the Diagnostics Overview screen, but may differ slightly if new data was received while drilling down."

**To get to the Transaction screen from the other Diagnostics screens:**

When you have navigated to another screen by drilling down from the Transaction screen, you can return to the Transaction screen by clicking on the Transaction bread crumb at the top of the screen.

## Description of Transactions Screen

Reference the following screen image when reading the information in this section.



### Screen Context - Bread Crumb Trail

You may return to the Diagnostics Overview screen by clicking on the **Diagnostics Overview** bread crumb.

**Note:** When you return to the Diagnostics Overview screen the drop-down list selections are retained but the changes that you made to the sort order for the table or the rows to display in the graph will be lost.

For more information on using the bread crumbs to navigate and to determine the context for the information displayed on the screen see “Screen Context - Bread Crumb Trail” on page 231.

### **Transactions: Response Time Graph**

When **Response Time** has been selected from the **Graph** drop-down, the graph on the Transactions screen shows the transaction response times for the transactions with the highest overall response time for the run.

The x-axis of the graph shows the elapsed time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows the total Response Time in milliseconds.

### **Transactions: Count Graph**

When the **Count** selection is made from the **Graph** drop-down, the graph on the Transactions screen shows the number of transaction calls that are made to a transaction each second.

The x-axis of the graph shows the elapsed time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows the count of the transactions per second.

### **Transactions Table**

The Transactions table lists the transactions for the run. The metrics reported in the table are filtered based upon the time period specified in the **Time Range** drop-down. See “Definitions for the Columns in the Diagnostics Table” on page 233 for a description of the columns in this table.

### **Details Panel**

The Details Panel lists more information for a selected row of the Transactions table. See “Definitions for the Rows in the Details Panel” on page 236 for a description of the metrics in this panel.

### **Refining the View In the Transactions Screen**

You may use the standard Diagnostics screen controls to adjust the amount of data and the type of data that is displayed on the Transactions Screen. For more information about the ways that you can control the how information is presented on this screen see “Using the Diagnostics Navigation and Display Controls” on page 230.

### **Drilling Down Into the Transaction Performance Metrics**

From the Transactions screen you may drill down into the Layers, and Service Requests that are behind the transactions.

You may drill down into the transactions using one of the following methods:

- Double click on a transaction listed in the Transaction Table at the bottom of the screen. The server requests metrics for the selected transaction are displayed on the Server Requests screen. See “Description of Server Request Breakdown Screen” on page 251 for more information on this screen.
- Right-click on a transaction listed in the Transaction Table at the bottom of the screen. A pop-up navigation menu for the selected transaction is displayed with the following options:

### **View Server Requests**

To view a breakdown of a selected transaction based upon the processing time for each server request, select the **View Server Requests** menu item from the pop-up menu. The server requests metrics for the selected transaction are displayed in a stacked area chart on the Server Request screen. See “Description of Server Request Breakdown Screen” on page 251 for more information on this screen.

### **View Layer Breakdown**

To view a breakdown of a selected transaction based upon the processing time for each layer where the processing is taking place, select the **View Layer Breakdown** menu item from the pop-up menu. The layer metrics for the selected transaction are displayed in a stacked area chart on the Layer Breakdown screen. See “Description of the Layer Breakdown Screen” on page 258 for more information on this screen.

---

**Note:** When you drill down into the performance metrics for a transaction that uses RMI, the breakdowns of the transaction processing that are displayed in the stacked area graphs will not sum to the total processing time for the transaction because the RMI server requests are double counted. They are counted in the "Remote to:" server requests and are also included in the callee server requests as well.

On the Server Request Breakdown screen for the transaction, if you deselect the Chart checkbox for the "Remote to" server requests, the stacked area chart of the server request processing will correctly sum up to equal the transaction processing time.

---



## Analyzing Performance with the Server Request Screens

### Purpose of Server Request Screens

The Server Request Screens display the server request performance metrics. The Server Requests Trends screen displays the metrics for Server Requests across all virtual machines and transactions. The Server Requests Breakdown screen displays the metrics for a selected transaction or virtual machine.

### Accessing the Server Request Screens

**To get to the Server Requests screen from the Diagnostics Overview screen:**

- 1** Double click the Server Requests graph.
- 2** The Server Requests screen is displayed. The graph should contain the same Server Requests that were displayed on the **Diagnostics Overview** Screen but, may differ slightly if new data was received while drilling down.

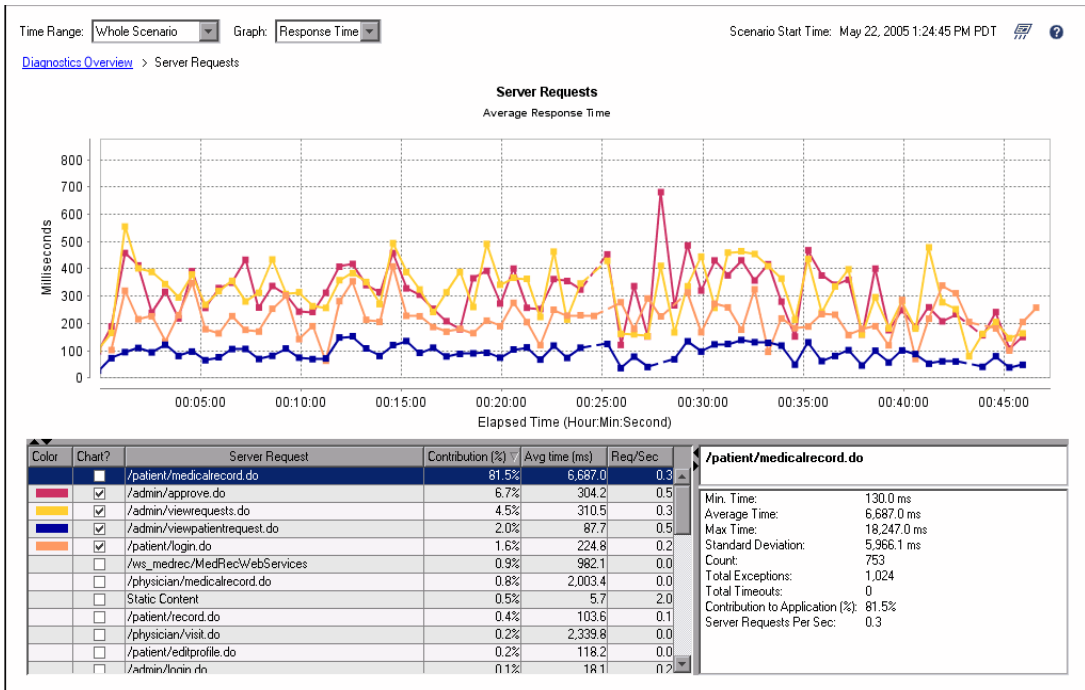
**To get to the Server Request Breakdown screen:**

You can access the Server Requests screen by drilling down into the metrics reported on the Transactions screen and the Virtual Machines screen.

- ▶ Instructions for drilling down to the Server Requests screen for a particular transaction on the Transactions screen can be found at “Drilling Down Into the Transaction Performance Metrics” on page 247.
- ▶ Instructions for drilling down to the Server Requests screen for a particular Virtual Machine on the Virtual Machines screen can be found at “Drilling Down Into the Virtual Machine Metrics” on page 265.

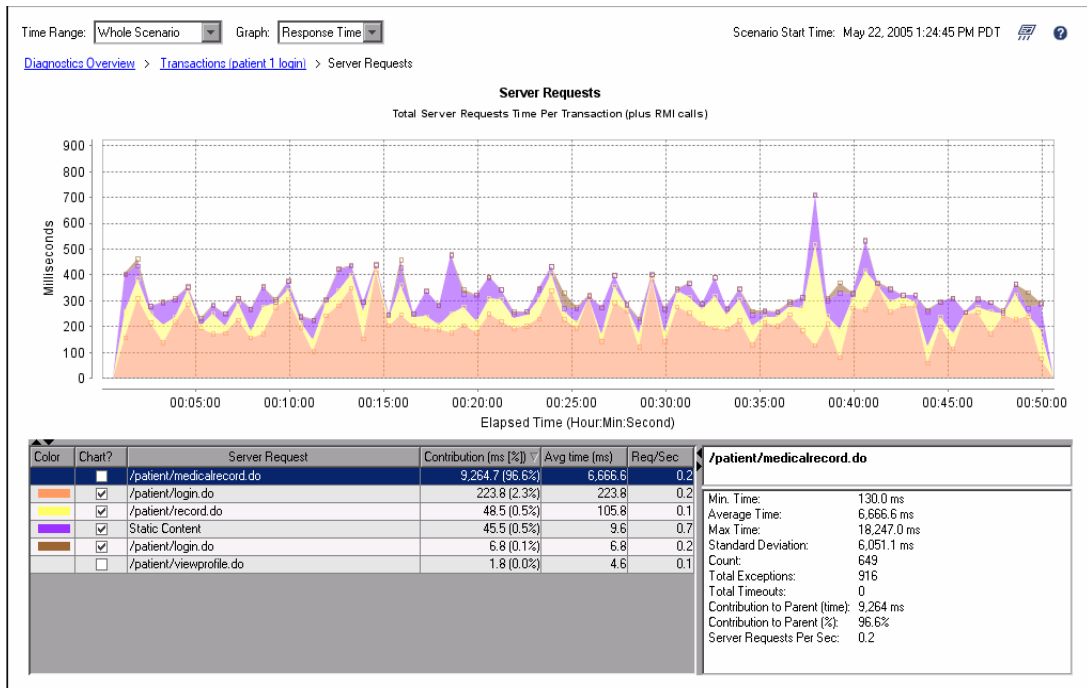
## Description of Server Requests Screen

The Server Requests screen displays the performance metrics for the aggregation of all of the instances of a server request during the scenario. The metrics are graphed using a trended line graph as shown in the following example screen image.



## Description of Server Request Breakdown Screen

The Server Request Breakdown screen displays the performance metrics for a single instance of a transaction or virtual machine. The graph displays the response times for the server requests that were part of the processing for a transaction or virtual machine using a stacked area graph as shown in the following example. The server requests will be displayed with the server request with the highest contribution appearing at the bottom. Layers with smaller contributions will be positioned in descending contribution order on top of each other.



### Server Breakdown Screen Types

► Total Server Request Time for Virtual Machine

This screen is accessed by drilling down on a Virtual Machine entry from the Virtual Machine screen. See “Drilling Down Into the Virtual Machine Metrics” on page 265.

► Total Server Request Time Per Transaction (Plus RMI calls)

This screen is accessed by drilling down on a Transaction from the Transactions screen. See “Drilling Down Into the Transaction Performance Metrics” on page 247.

---

**Note:** When you drill down into the performance metrics for a transaction that uses RMI, the breakdowns of the transaction processing that are displayed in the stacked area graphs will not sum to the total processing time for the transaction because the RMI server requests are double counted. They are counted in the "Remote to:" server requests and are also included in the callee server requests as well. On the Server Request Breakdown screen for the transaction, if you deselect the Chart checkbox for the "Remote to" server requests, the stacked area chart of the server request processing will correctly sum up to equal the transaction processing time.

---

### Screen Context

On the Server Requests screen, the only bread crumb displayed is the **Diagnostics Overview** bread crumb.

---

**Note:** When you return to the Diagnostics Overview screen the drop-down list selections are retained but the changes that you made to the sort order for the table or the rows to display in the graph will be lost.

---

On the Server Request Breakdown screen for a transaction you will see a bread crumb for the transaction or virtual machine in addition to the **Diagnostics Overview**. For more information on using the bread crumbs to navigate and to determine the context for the information displayed on the screen see “Screen Context - Bread Crumb Trail” on page 231.

### **Server Requests: Response Time Graph**

When **Response Time** has been selected from the **Graph** drop-down, the graph on the Server Request screens shows the server request response times for the transactions with the highest overall response time for the run.

The x-axis of the graph shows the elapsed time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows the response time in milliseconds.

### **Server Requests: Count Graph**

When the **Count** selection is made from the **Graph** drop-down, the graph on the Server Request screens shows the number of calls that are made to a service request per second.

The x-axis of the graph shows the actual chronological time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows the count of server requests per second.

### **Server Requests Table**

The Server Requests table lists all of the server requests for the context displayed in the bread crumbs. The metrics reported in the table are filtered based upon the time period specified in the **Time Range** drop-down. See “Definitions for the Columns in the Diagnostics Table” on page 233 for a description of the columns in this table.

### **Details Panel**

The Details Panel lists more information for the selected row of the Server Requests table. See “Definitions for the Rows in the Details Panel” on page 236 for a description of the rows in this table.

## Refining the View In the Server Requests Screen

You may use the standard Diagnostics screen controls to adjust the amount of data and the type of data that is displayed on the Server Requests Screen. For more information about the ways that you can control the how information is presented on this screen see “Using the Diagnostics Navigation and Display Controls” on page 230.

## Drilling Down Into the Server Request Performance Metrics

From the Server Request screens you may drill down into the Layers, and Service Requests that are behind the transactions.

You may drill down into the server requests using one of the following methods:

- ▶ Double click on a server request listed in the Server Request Table at the bottom of the screen. The virtual machine metrics for the selected server request are displayed on the Virtual Machines screen. See “Description of the Virtual Machines Screen” on page 262 for more information on this screen.
- ▶ Right-click on a server request listed in the Server Request Table at the bottom of the screen. A pop-up navigation menu for the selected server request is displayed with the following options:

### View Virtual Machines

To view the processing metrics for the virtual machine on which the server request was processed, select the **View Virtual Machines** menu item from the pop-up menu. The virtual machine metrics for the selected server request are displayed on the Virtual Machines screen. See “Description of the Virtual Machines Screen” on page 262 for more information on this screen.

### View Aggregated Profile for...

To view the aggregated call profile for a server request select the **View Aggregate Profile for...** menu option. The Aggregate Profile screen is displayed with the aggregated profile for the selected server request. See “Description of Aggregate Profile Screen” on page 268 for more information on this screen.

**Note:** This is a shortcut that allows you to navigate directly to the aggregated profile for a server request on a particular virtual machine without having to go to the Virtual Machines screen first. The bread crumb will include a entry for the level that was skipped.

---

### **View Layers for...**

To view the layer breakdown for server request processing on a virtual machine select the **View Layers for...** menu item from the pop-up menu. The Layer Breakdown screen is displayed with the processing metrics for the selected server request. See “Description of the Layers Screen” on page 257 for more information on this screen.

---

**Note:** This is a shortcut that allows you to navigate directly to the Layer Breakdown screen for a server request on a particular virtual machine without having to go to the Virtual Machines screen first. The bread crumb will include a entry for the level that was skipped.

---

## **Analyzing Performance with the Layer Screens**

### **Purpose of Layers Screens**

The Layer Screens display the performance metrics for the layers where processing has taken place in your application. The Layers - Relative Load screen displays the metrics for Layers across all virtual machines and transactions. The Layers Breakdown screen displays the metrics for a selected transaction or virtual machine.

The default layers for which metrics are gathered and reported on the Layers screen include: servlets, JSPs, session and entity beans, JNDI, JDBC, JMS, and Struts for the J2EE Probes and WEB.ASP, DB.ADO, and MSG.MSMQ for the .NET Probe with ASP.NET applications.

Occasionally, due to design decisions, a class that does not directly implement a J2EE component may contain J2EE functionality. Or you may wish to monitor a class that is of special interest to you. For these purposes, you can define a custom layer. To enable Diagnostics for J2EE & .NET to display custom classes or packages, you must configure the J2EE Probe so that it will monitor the classes and packages. For details, see Chapter 8, “Installing the Mercury Diagnostics Probe for J2EE.”

## Accessing the Layers Screens

**To get to the Layers - Relative Load screen from the Diagnostics Overview screen:**

- 1 Double click the Layers - Relative Load graph.
- 2 The Layers screen is displayed. The graph should contain the same Layers that were displayed on the **Diagnostics Overview** Screen but, may differ slightly if new data was received while drilling down.

**To get to the Layers Breakdown screen:**

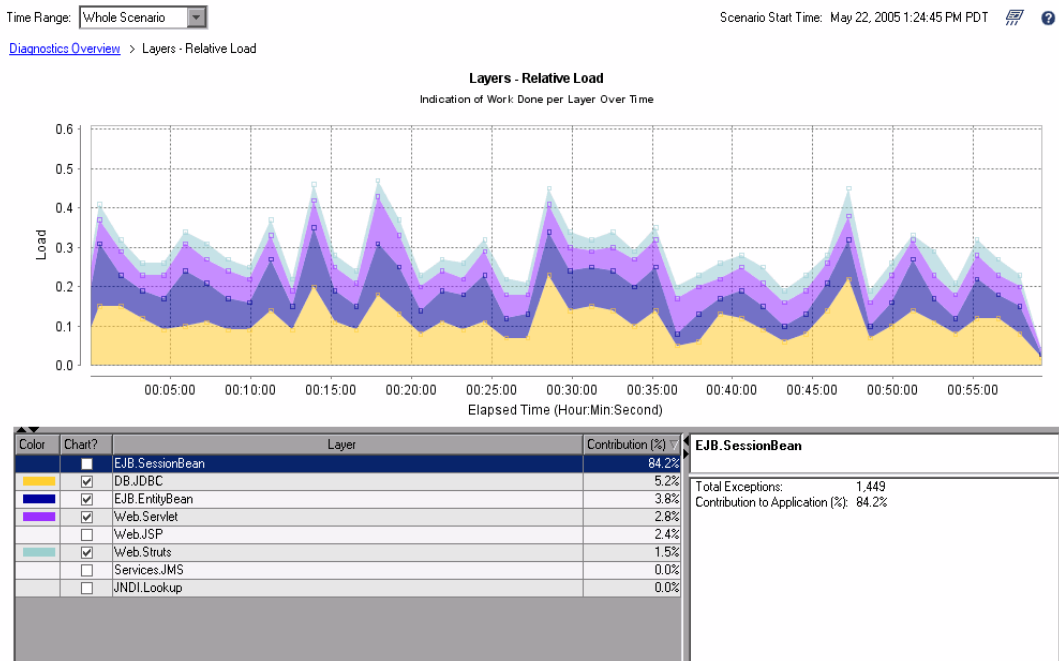
You can access the Server Requests screen by drilling down into the metrics reported on the Transactions screen and the Virtual Machines screen.

- ▶ Instructions for drilling down to the Layers screen for a particular transaction on the Transactions screen can be found at “Drilling Down Into the Transaction Performance Metrics” on page 247.
- ▶ Instructions for drilling down to the Layers screen for a particular Virtual Machine on the Virtual Machines screen can be found at “Drilling Down Into the Virtual Machine Metrics” on page 265.



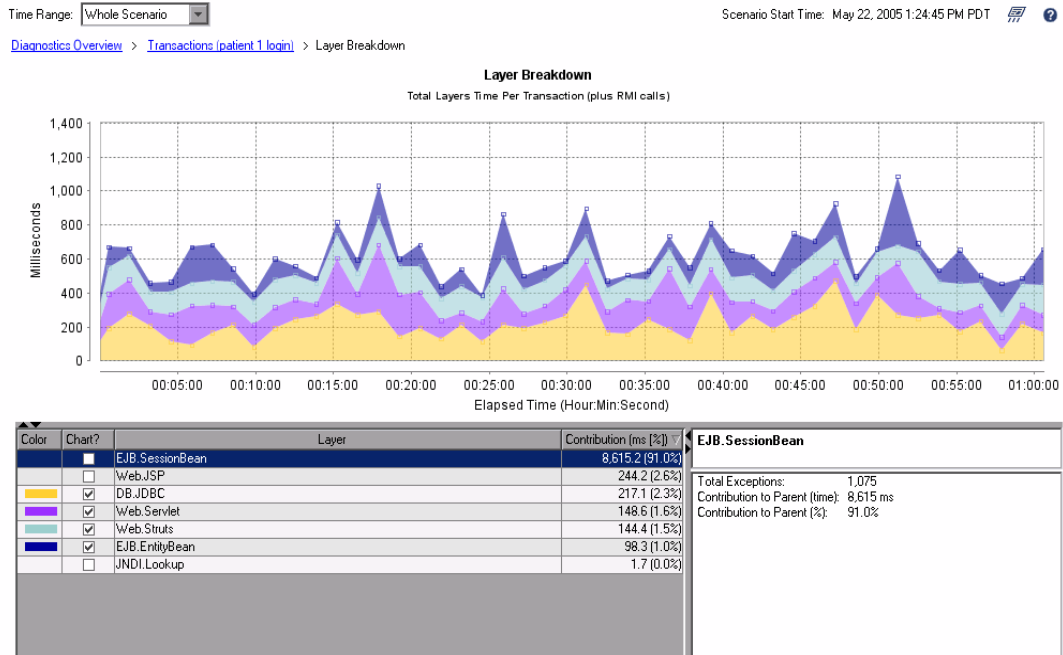
## Description of the Layers Screen

The Layers screen displays the performance metrics for the aggregation of all of the processing in the layers during the scenario. The metrics are graphed using a stacked area graph as shown in the following example. The layers will be displayed with the layer with the highest contribution appearing at the bottom. Layers with smaller contributions will be positioned in descending contribution order on top of each other.



## Description of the Layer Breakdown Screen

The Layer Breakdown screen displays the performance metrics for the aggregation of all of the processing in the layers for the context shown in the bread crumbs. The metrics are graphed using a using a stacked area graph as shown in the following example:



### Types of Layer Breakdown Screens

- ▶ Total Layers Time Per Transaction (Plus RMI Calls)

This screen is accessed by drilling down on a Transaction entry from the Transactions screen. See “Drilling Down Into the Transaction Performance Metrics” on page 247.

- ▶ Total Layers Time Per Transaction for Server Request for Virtual Machine

This screen is accessed by drilling down from a Transaction on the Transactions screen, through a Server Request on the Server Request Breakdown Screen, through a Virtual Machine on the Virtual Machine Breakdown screen.

► Total Layers Time Per Server Request for Virtual Machine

This screen is accessed by drilling down from a Server Request on the Server Requests screen, through a Virtual Machine on the Virtual Machine Breakdown screen.

► Total Layer Time for Virtual Machine

This screen is accessed by drilling down from a Virtual Machine on the Virtual Machine screen.

► Total Layers Time Per Virtual Machine for Server Request

This screen is accessed by drilling down from a Virtual Machine on the Virtual Machine Breakdown screen, through a Server Request on the Server Request Breakdown screen.

**Screen Context - Bread Crumb Trail**

When you are looking at the aggregated results across all virtual machines and server requests, the **Diagnostics Overview** bread crumb is the only bread crumb displayed on this screen.

---

**Note:** When you return to the Diagnostics Overview screen the drop-down list selections are retained but the changes that you made to the sort order for the table or the rows to display in the graph will be lost.

---

When you are viewing the Layer Breakdown for a server request or virtual machine you will see a bread crumb for the appropriate item in addition to the **Diagnostics Overview**. For more information on using the bread crumbs to navigate and to determine the context for the information displayed on the screen see “Screen Context - Bread Crumb Trail” on page 231.

**Layers: Response Time Graph**

When **Response Time** has been selected from the **Graph** drop-down, the graph on the Layer screens shows the layer response times for the layers with the highest overall response time for the run.

The x-axis of the graph shows the actual chronological time for the run in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the total Response Time in milliseconds.

### **Layers: Count Graph**

The **Graph** drop-down does not appear on this screen and there is no count graph for the Layers screens.

### **Layers Table**

The Layers table lists all of the layers that pertain to the context shown in the bread crumbs listed at the top of the screen. The metrics reported in the table are filtered based upon the time period specified in the **Time Range** drop-down. See “Definitions for the Columns in the Diagnostics Table” on page 233 for a description of the columns in this table.

### **Details Panel**

The Details Panel lists more information for the selected row of the Layers table. See “Definitions for the Rows in the Details Panel” on page 236 for a description of the rows in this panel.

### **Refining the View In the Layers Screen**

You may use the standard Diagnostics screen controls to adjust the amount of data and the type of data that is displayed on the Layers Screen. For more information about the ways that you can control the how information is presented on this screen see “Using the Diagnostics Navigation and Display Controls” on page 230.

### **Drilling Down Into the Layer Metrics**

There is no drill down provided from the Layer screens.

## Analyzing Performance with the Virtual Machines Screen

In Diagnostics a virtual machine corresponds to a probe implementation. The Virtual Machines screen displays the average response times for the virtual machines that have been captured. The Virtual Machines screen displays the metrics for the processing for a particular server request on a particular Virtual Machine.

---

**Note:** When RMI instrumentation is used, a "CrossVM" layer will appear in the data denoting the time spent in remote method calls. (even though the user didn't specify that layer in the auto\_detect.points file).

---

### Accessing the Virtual Machines Screen

**To get to the Virtual Machines screen from the Diagnostics Overview screen:**

- 1 Double click the Virtual Machines graph.
- 2 The Virtual Machines screen is displayed. The graph should contain the same Virtual Machines that were displayed on the **Diagnostics Overview** screen but, may differ slightly if new data was received while drilling down.

**To get to the Virtual Machines screen from the other Diagnostics screens:**

When you have navigated to another screen by drilling down from the Virtual Machine screen, you can return to the Virtual Machine screen by clicking on the Virtual Machines bread crumb at the top of the screen.

**To get to the Layers screen from the Diagnostics screens:**

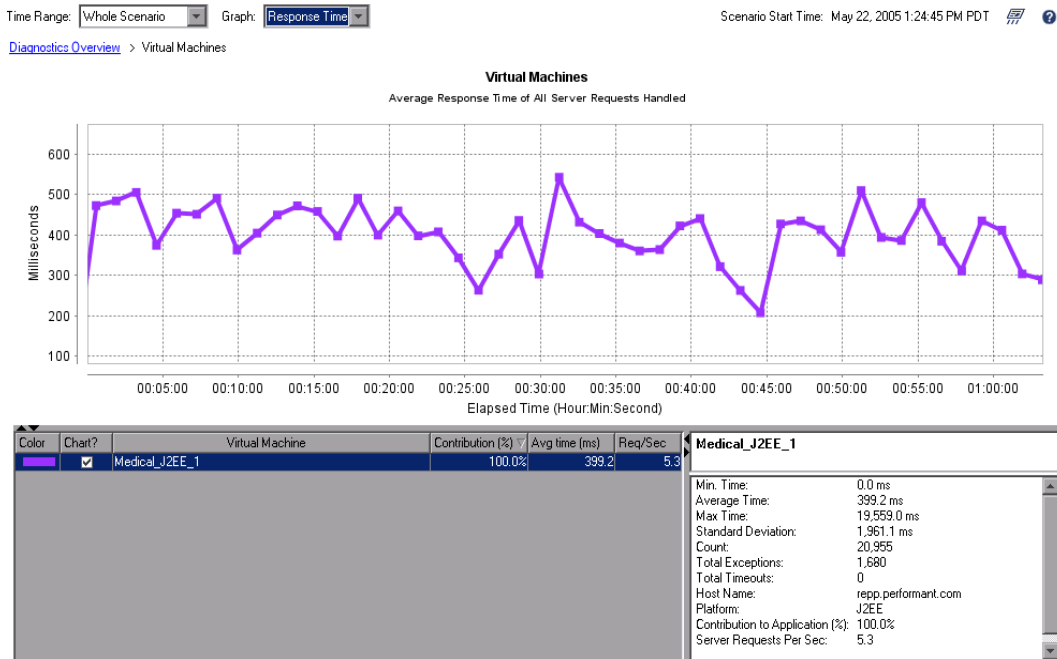
You can access the Virtual Machines screen by drilling down into the metrics reported on the Transactions screen and the Server Request screen.

- Instructions for drilling down to the Virtual Machines screen for a particular transaction on the Transactions screen can be found at "Drilling Down Into the Transaction Performance Metrics" on page 247.

- Instructions for drilling down to the Virtual Machines screen for a particular server request can be found at “Drilling Down Into the Server Request Performance Metrics” on page 254.

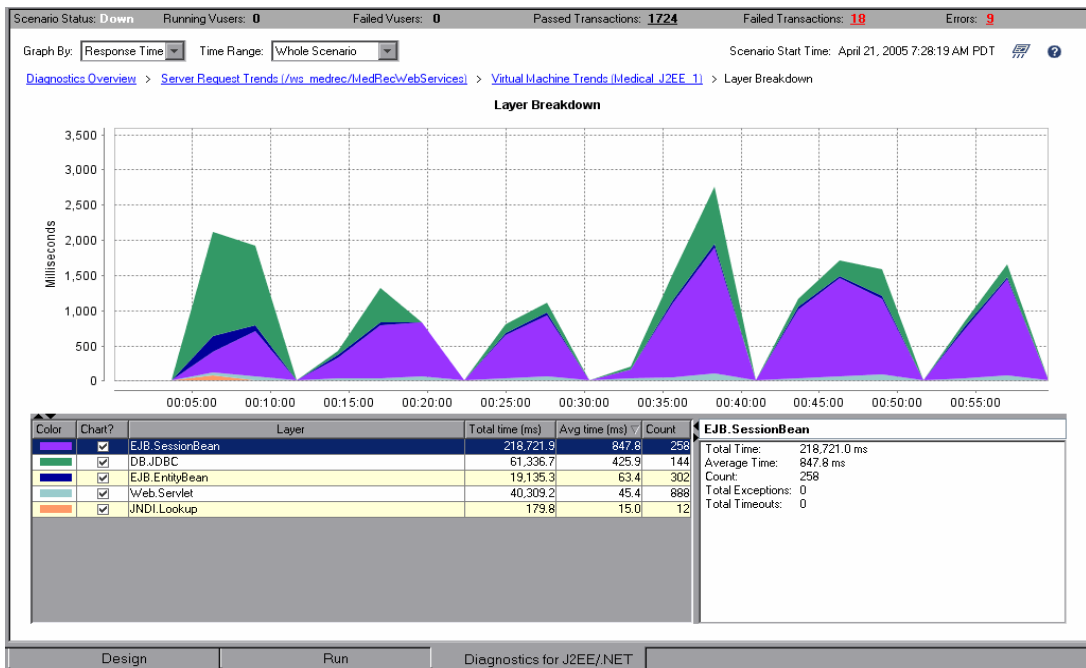
## Description of the Virtual Machines Screen

The Virtual Machines screen displays the performance metrics for the aggregation of all of the processing for the virtual machine during the scenario. The metrics are graphed using a trended line graph as shown in the following example screen image.



## Description of the Virtual Machine Breakdown Screen

The Virtual Machines Breakdown screen displays the performance metrics for the aggregation of all of the processing for the virtual machine during the scenario. The metrics are graphed using a stacked area chart shown in the following example. The virtual machines will be displayed with the VM with the highest contribution appearing at the bottom. VM with smaller contributions will be positioned in descending contribution order on top of each other.



## Types of Virtual Machine Breakdown Screens

- ▶ Total Virtual Machine Time Per Transaction for Server Request

This screen is accessed by drilling down from a Transaction entry from the Transactions screen, through the Server Request Breakdown screen.

- ▶ Total Virtual Machine Time for Server Request

This screen is accessed by drilling down from a Transaction entry from the Transactions screen, through the Server Request Breakdown screen.

### **Screen Context - Bread Crumb Trail**

When you are looking at the aggregated results across all virtual machines and server requests, the **Diagnostics Overview** bread crumb is the only bread crumb displayed on this screen.

---

**Note:** When you return to the Diagnostics Overview screen the drop-down list selections are retained but the changes that you made to the sort order for the table or the rows to display in the graph will be lost.

---

When you are viewing the Virtual Machine Breakdown for a transaction or server request you will see a bread crumb for the appropriate item in addition to the **Diagnostics Overview**. For more information on using the bread crumbs to navigate and to determine the context for the information displayed on the screen see “Screen Context - Bread Crumb Trail” on page 231.

### **Virtual Machines: Response Time Graph**

When **Response Time** has been selected from the **Graph** drop-down, the graph on the Virtual Machines screen shows the VM response times for the VMs with the highest overall response time for the run.

The x-axis of the graph shows the actual chronological time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows the total response time in milliseconds.

### **Virtual Machines: Count Graph**

When the **Count** selection is made from the **Graph** drop-down, the graph on the Virtual Machines screen shows the number of server request calls that are made to the VM in each second.

The x-axis of the graph shows the actual chronological time for the run in hours, minutes, and seconds (hh:mm:ss).

The y-axis of the graph shows count of calls to the VM per second.



### **Virtual Machine Table**

The Virtual Machines table lists all of the VMs that pertain to the context shown in the bread crumbs listed at the top of the screen. The metrics reported in the table are filtered based upon the time period specified in the **Time Range** drop-down. See “Definitions for the Columns in the Diagnostics Table” on page 233 for a description of the columns in this table.

### **Details Panel**

The Details Panel lists more information for the selected row of the Virtual Machines table. See “Definitions for the Rows in the Details Panel” on page 236 for a description of the rows in this panel.

### **Refining the View In the Virtual Machines Screen**

You may use the standard Diagnostics screen controls to adjust the amount of data and the type of data that is displayed on the Virtual Machines Screen. For more information about the ways that you can control the how information is presented on this screen see “Using the Diagnostics Navigation and Display Controls” on page 230.

### **Drilling Down Into the Virtual Machine Metrics**

From the Virtual Machines screen you may drill down into the Server Requests, Layers, and the Aggregated Profile that are behind the VM processing.

You may drill down into the server requests using one of the following methods:

- ▶ Double click on a virtual machine listed in the Virtual Machine table at the bottom of the screen. The server request metrics for the selected virtual machine are displayed on the Server Request Breakdown screen. See “Description of Server Request Breakdown Screen” on page 251 for more information on this screen.
- ▶ Right-click on a virtual machine listed in the Virtual Machine table at the bottom of the screen. A pop-up navigation menu for the selected server request is displayed with the following options:

### **View Server Requests**

To view the processing metrics for the server requests that were processed on the virtual machine, select the **View Server Requests** menu item from the pop-up menu. The server requests for the metrics for the selected virtual machine are displayed on the Server Request Breakdown screen. See “Description of Server Request Breakdown Screen” on page 251 for more information on this screen.

### **View Layers**

To view a breakdown of the processing for the virtual machine based upon the layers where the processing is taking place, select the **View Layers** menu item from the pop-up menu. The Layer Breakdown screen is displayed. See “Description of the Layer Breakdown Screen” on page 258 for more information on this screen.

## Analyzing Performance with the Aggregate Profile Screen

### Purpose of Aggregate Profile Screen

The Aggregate Profile Screen provides a way for you to drill down into server requests and virtual machines metrics to examine the performance of the method calls.

An aggregate profile presents graphical views of the sum of all call instances for a type of service request selected from the Server Requests screen or virtual machine selected from the Virtual Machines screen. The time depicted for a call in the aggregate profile is the average time spent in the call for all of the aggregated calls.

---

**Note:** An aggregate profile shows the complete set of calls for all of the invocations of the items being profiled. This means that some of the calls that appear in an aggregate profile may not have applied to every instance.

---

### Accessing the Aggregate Profile Screen

**To get to the Aggregate Profile Screen from the Virtual Machines screen:**

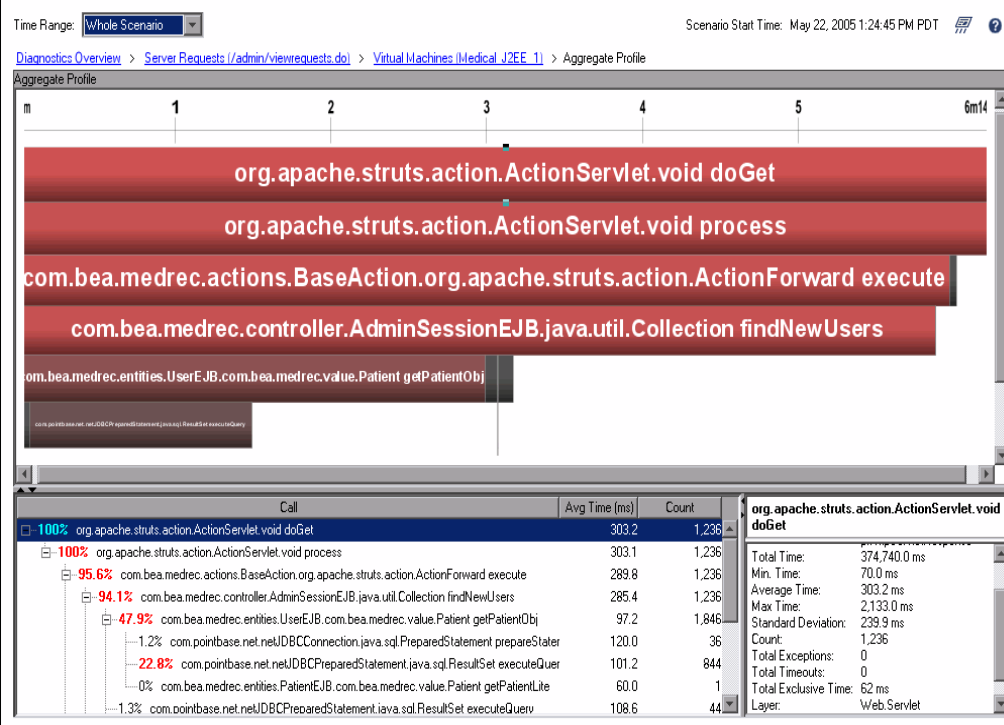
You can get to the Aggregate Profile Screen for an item from the Virtual Machines screen. See “Drilling Down Into the Diagnostics Metrics” on page 228 for navigation paths to the Aggregate Profile Screen.

**To view the Aggregated Profile for a selected virtual machine:**

- 1** Right-click on an item in the Virtual Machines table and select the **View Aggregate Profile** menu option.
- 2** The Aggregate Profile Screen is displayed with the call profile for the item indicated on the Virtual Machines screen.

## Description of Aggregate Profile Screen

The Aggregate Profile screen is divided into three parts. The top part of the screen contains the Call Profile Graph and the bottom part of the screen contains the Call Tree Table and the Details Pane.



## Description of Call Profile Graph

The horizontal axis of the call profile graph represents elapsed time where time progresses from left to right. The calls are distributed across the horizontal axis based upon the time when they take place and sequence of the calls relative to each other. The legend across the top of the Profile Graph denotes the amount of time in seconds.

The vertical axis on the call profile represents the call stack or nesting level. The calls made at the higher levels of the call stack are shown at the top of the profile and those made at deeper levels of the call stack shown at the lower levels of the profile.

Each box in the profile graph represents a call where the left edge of the box is the start of the call and the right edge is the return from the call. The length of the box indicates the duration of the call execution. The call boxes that appear directly beneath a parent call box are the child calls that are invoked by the parent call.

The gaps between the call boxes on a layer of the profile indicate one of the following processing conditions:

- ▶ The processing during the gap is taking place in local code to a particular call and not in child calls in a lower layer.
- ▶ The call is waiting to acquire a lock or mutex.
- ▶ The processing during the gap is taking place in a child call that was not instrumented or included in a capture plan for the run.

You can view the details for a particular call by mousing over the box that represents the call. The call details that are included in the mouse-over pop-up box are:

- ▶ The number of calls made.
- ▶ The average time spent in each call.
- ▶ The total amount of time spent in the call.
- ▶ The threads on which the calls occurred.

The calls that are part of a path through the profile that has the highest response time are colored red. Call path components that are not part of a critical high-latency path are colored yellow.

### **Analyzing Performance with a Call Profile Graph**

The Call Profile Graph allows you to do the following analysis:

- ▶ Determine whether an observed latency has one cause at a certain point in the code or many causes distributed throughout the code.
- ▶ In a multi-tier correlated diagram, determine which tier contributes the highest percentage of the total latency.
- ▶ Explore and inspect the dynamic behavior of a complex system.

## Description of the Call Tree Table

The Call Tree Table appears directly below the Call Profile Graph. This table presents the same data represented in the Call Profile Graph in a tabular format.

The first row in the table contains the “root” of the call stack which is the server request that you drilled down on when you requested the call profile. The children rows in the tree are the method calls that were made as a result of the server call.

The table contains the following columns:

---

**Note:** For an aggregate profile the counts and times in the following table represent the total for all of the calls that were included in the aggregation.

---

- ▶ **Call** - The server request call or method call whose performance metrics are reported in the row.
- ▶ **Avg Time** - The average latency for the call. The time is reported in milliseconds.
- ▶ **Count** - A count of the number of times that the call was executed.

## Details Panel

The Details Panel lists more information for the selected row of the Call Tree table. See “Definitions for the Rows in the Details Panel” on page 236 for more information about this metrics in this panel.

## Drilling Down Into the Graph or Call Tree

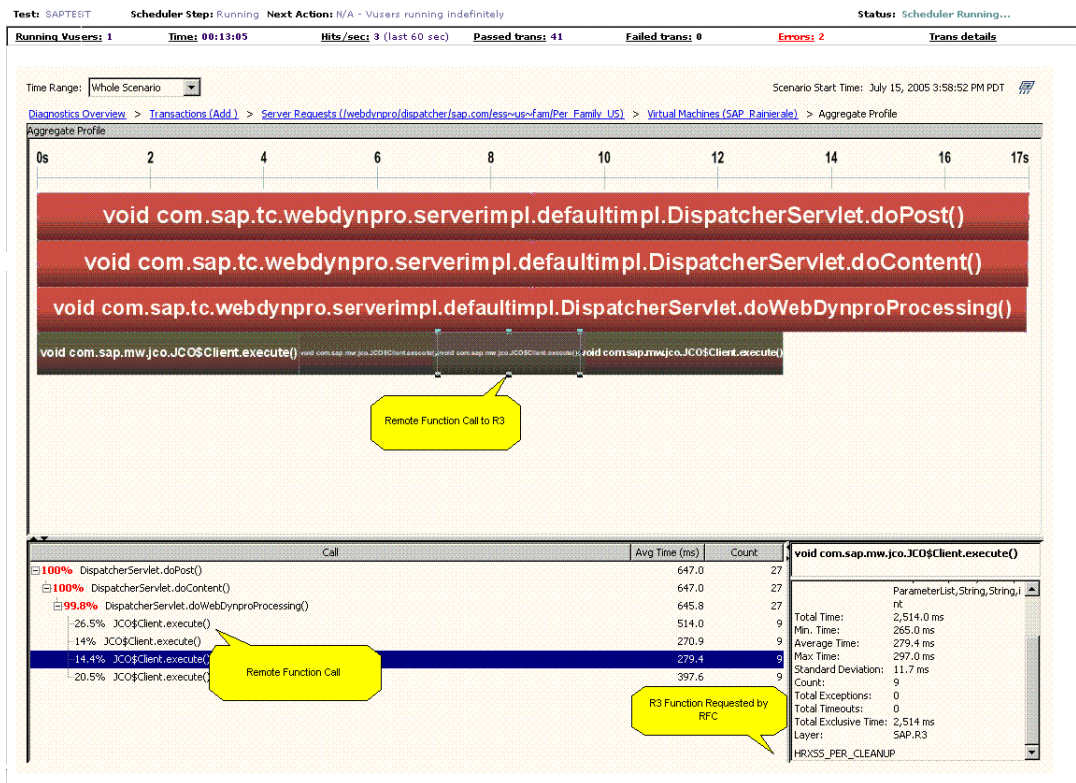
To find out more about a call on the profile graph, you can mouse-over a call in the graph to get a pop-up with the details.

To find out more about the calls in the call tree, you can expand and collapse the levels of the tree so that the lower level information on a call is displayed.

## Analyzing J2EE to SAP R3 Remote Function Calls

The Remote Function Call (RFC) protocol in SAP allows communication to take place between SAP J2EE and SAP R3 environments. The remote calls that are made between SAP J2EE and SAP R3 environments are displayed in the Aggregate Profile as shown in the following screen image.

When a JAVA method executes an RFC, it specifies the R3 function as a string in the RFC. Diagnostics displays the RFC as a child box under the service request in the call profile graph and lists it as a child of the service request in the Call Tree Table. When the RFC is selected from the Call Tree Table, the details for the call are displayed in the Details Panel. The last line in the Details Panel for an RFC to the R3 layer contains the R3 function that was being executed.



### **Analyzing Remote Method Invocation (RMI)**

When analyzing RMI on the Aggregate Profile screen you must be aware that the latency displayed for the remote caller in the call profile graph and the call tree table includes the processing time for the remote callee. The remote callee is also displayed separately in the call profile graph and call tree with just the latency for its processing.

### **Analyzing Life-cycle Methods for Portlets**

Life-cycle methods for Portlets are identified by the name of the method (beginRender, endRender, etc.), and the Portlet on which the method was invoked. The Portlet name is a combination of its title and label.

When you see a life-cycle method in the call profile graph, you can identify the life-cycle method of a Portlet using the method name that is displayed in the call tree table and the Portlet name this is displayed in the details pane.



# 16

---

## J2EE & .NET Diagnostics Analysis Graphs

After a scenario or session step run, you can use LoadRunner Analysis to display the J2EE & .NET Diagnostics graphs to analyze server performance.

For instructions for using LoadRunner Analysis, refer to the *Mercury LoadRunner Analysis User's Guide*.

This chapter describes the following topics:

- ▶ About J2EE & .NET Diagnostics Graphs
- ▶ Viewing J2EE & .NET Diagnostics in the Summary Report
- ▶ Viewing J2EE & .NET Diagnostics Data
- ▶ J2EE & .NET Diagnostics Graphs
- ▶ J2EE & .NET Server Diagnostics Graphs

### About J2EE & .NET Diagnostics Graphs

The J2EE & .NET Diagnostics graphs in LoadRunner Analysis enable you to trace, time, and troubleshoot individual transactions and server requests through J2EE & .NET Web, application, and database servers. You can also quickly pinpoint problem servlets and JDBC calls to maximize business process performance, scalability, and efficiency.

The J2EE & .NET Diagnostics graphs are comprised of two groups:

- ▶ **J2EE & .NET Diagnostics Graphs:** These graphs show you the performance of requests and methods generated by virtual user transactions. They show you the transaction that generated each request.

- ▶ **J2EE & .NET Server Diagnostics Graphs:** These graphs show you the performance of all the requests and methods in the application you are monitoring, without connection to any transactions. These include requests generated by virtual user transactions and by real users.

To obtain data for these graphs, you need to activate the Mercury Diagnostics Server before running the scenario or session step. When you set up the Mercury Diagnostics Server online monitors, you specify the sampling percentage of diagnostics data to include in the diagnostics graphs.

## Viewing J2EE & .NET Diagnostics in the Summary Report

The J2EE & .NET Diagnostics Usage section of the Summary report in Analysis provides general information about scenario execution and a usage chart for the J2EE & .NET Diagnostics and server request layers. This report is available from the tree view or as a tab in the Analysis window.






---

**Note:** If you do not see diagnostics data on the Summary Report, check if you are using a user-defined template. To view relevant data, choose a different template from the list of templates, or create and apply a new template. For more information about using templates, see “Using Templates” on page 20.






---

The J2EE /.NET Diagnostic Usage section breaks the individual transactions and server requests into Web server activity (Servlets and JSPs data), application server activity (JNDIs), and back-end activity of database requests (JDBC methods and SQL queries), and provides the total usage time for each transaction and request.

#### J2EE/.NET Diagnostics Usage

Top Transactions	J2EE/.NET Diagnostics Layers	Total time (sec)
<a href="#">RunChain</a>		2,499,545
<a href="#">myPage</a>		138,252
<a href="#">enterSamplePortal</a>		80,869
<a href="#">ContentManagement</a>		45,482
<a href="#">JumpToAdminPortal</a>		33,997



Top Requests	J2EE/.NET Diagnostics Layers	Total time (sec)
<a href="#">/CallChainWebApp/CallChain</a>		2,503,043
<a href="#">/sampleportal/sample.portal</a>		275,255
<a href="#">/portalAppAdmin/portal.portal</a>		48,724
<a href="#">com.mercury.ga.callchain.eib.CSessionBean - StringBuffer callMethods(String,String,int,boolean)</a>		45,847
<a href="#">Static_Content</a>		2,545



#### To view server side transaction and server request diagnostics data from the Summary Report:

In the J2EE/.NET Diagnostics Usage section of the Summary Report, click the transaction or J2EE /.NET layer on which you want to perform breakdown. The J2EE/.NET - Transaction Time Spent in Element graph or J2EE/.NET - Server Request Time Spent in Element graph opens.

Clicking a transaction or server request displays the breakdown to layers over time of the selected transaction or server request.

Clicking a layer displays the specific layer breakdown in the transaction or server request.

For more information on J2EE/.NET Diagnostics graphs, see “Viewing J2EE & .NET Diagnostics Data” on page 276.

## Viewing J2EE & .NET Diagnostics Data

The J2EE & .NET Diagnostics graphs provide an overview of the entire chain of activity on the server side of the system. At the same time, you can break down J2EE/.NET layers into classes and methods to enable you to pinpoint the exact location where time is consumed. In addition, you can view custom classes or packages that you set the J2EE/.NET probe to monitor. You can also view the transaction chain of calls and call stack statistics to track the percentage of time spent on each part of the transaction.

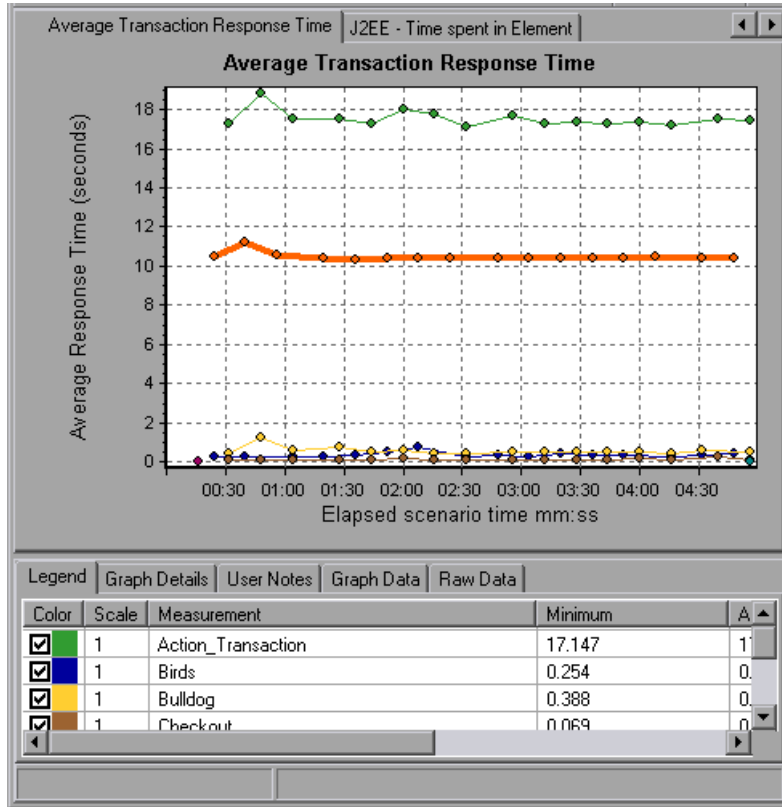
You can correlate the end user response time with the Web server activity (Servlets and JSPs data), application server activity (JNDIs), and back-end activity of database requests (JDBC methods and SQL queries).

### Example Transaction Breakdown

The following graphs illustrate the breakdown of a transaction to its layers, classes, and methods.

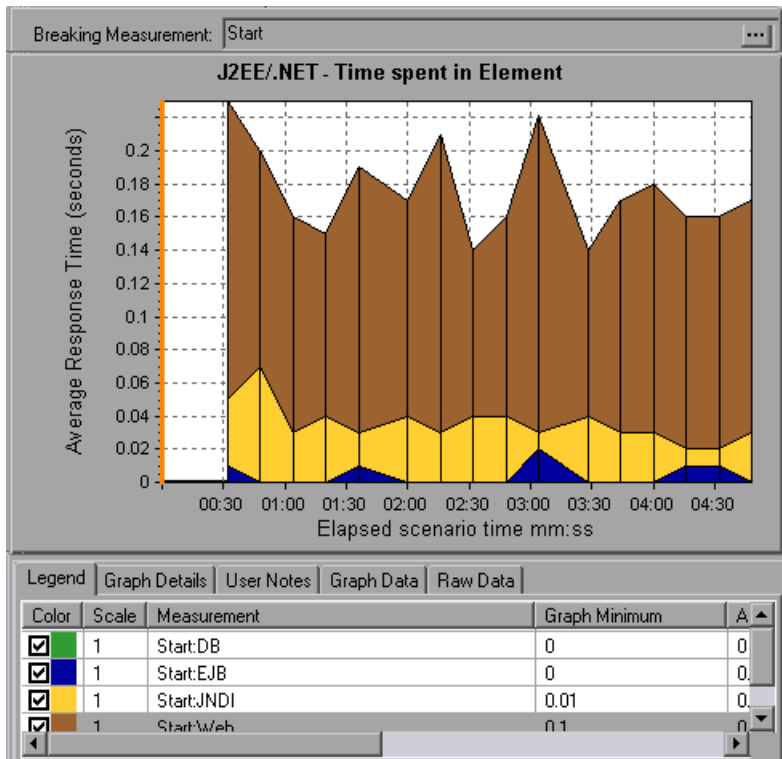
### Transaction Level

The following figure displays the top level Average Transaction Response Time graph. The graph displays several transactions: **Birds**, **Bulldog**, **Checkout**, **Start**, etc.



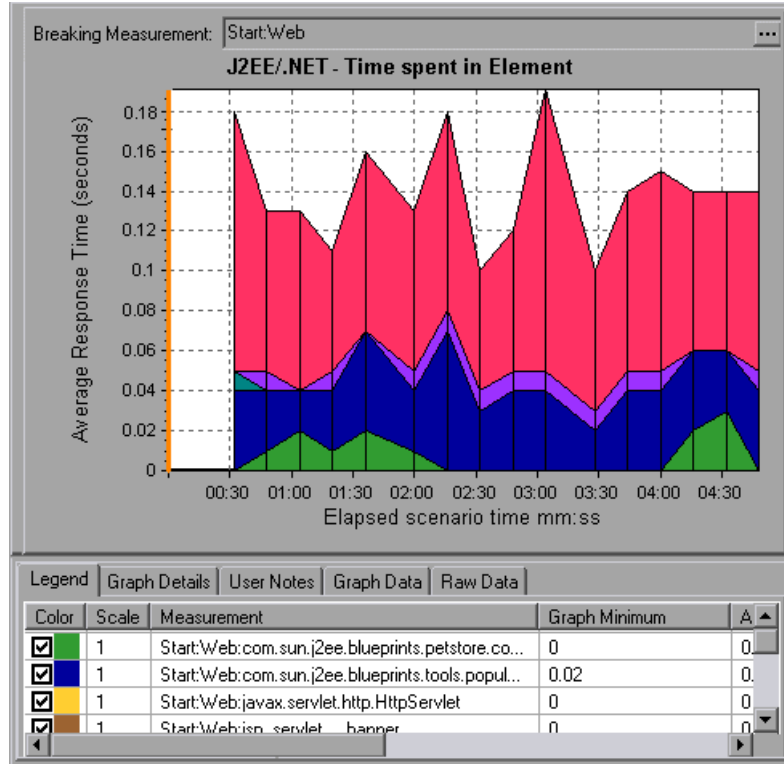
### Layer Level

In the following figure, the **Start** transaction has been broken down to its layers (DB, EJB, JNDI, and Web). In J2EE/.NET transactions, the Web layer is generally the largest.



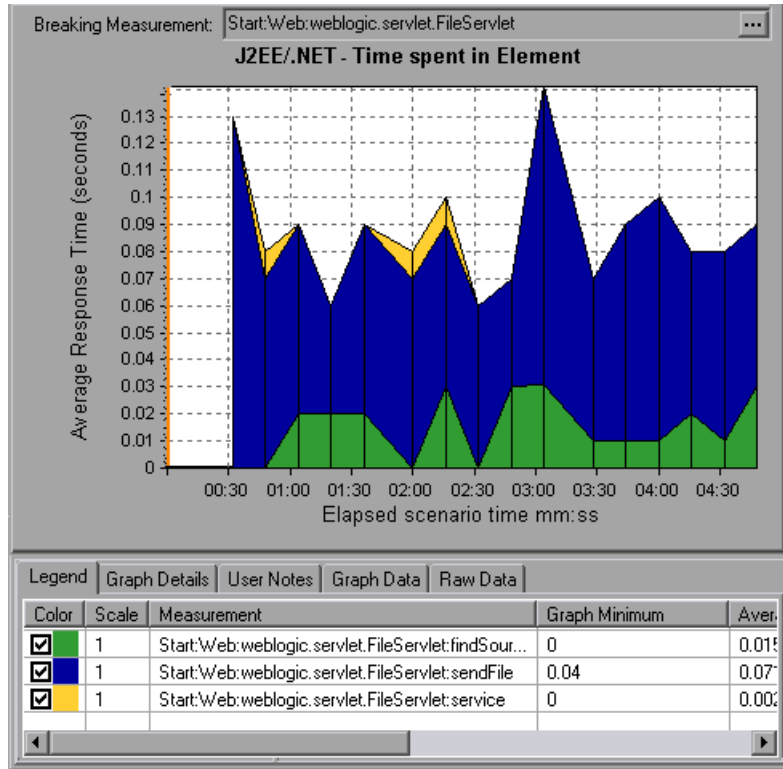
## Class Level

In the following figure, the Web layer of the **Start** transaction has been broken down to its classes.



### Method/Query Level

In the following figure, the **weblogic.servlet.FileServlet** component of the **Web** layer of the **Start** transaction has been broken down to its methods.




---

**Note:** Some JDBC methods can invoke SQLs which can be broken down further. In this case there is another level of breakdown, that is SQL Statements. For the methods that can not be further broken down into SQL statements when reaching this level of breakdown, you see **NoSql**.

---



## Cross VM Analysis

When a server request makes a remote method invocation, the J2EE & .NET Diagnostics graphs display certain measurements relating to the classes and methods involved in these requests. These measurements are displayed at a layer, class and method level. The VM making the call is referred to as the *caller VM*, and the VM that executes the remote call is the *callee VM*.

**Cross VM Layer** is a measurement that represents a dummy layer that integrates the data from the remote classes and methods in server requests that take place across two or more virtual machines.

**Remote-Class** is a measurement that represents a dummy class that integrates the data from the remote methods in server requests that take place across two or more virtual machines.

**Remote-Class: Remote Method** is a measurement that represents a dummy method. Remote-Class: Remote Method measures the total time, call count, exclusive latency, minimum and maximum values, standard deviation, and so on of the methods that are executed remotely, relative to the caller virtual machine.

---

**Note:** Since this data is measured on the caller virtual machine the exclusive latency will include all of the time required for making the remote method invocation such as network latency.

---

## Using the J2EE & .NET Breakdown Options

You can activate the J2EE & .NET breakdown options in any one of the following ways:

- ▶ from the View menu
- ▶ by right-clicking on a transaction or server request and choosing the option from the short-cut menu
- ▶ by clicking the button in the toolbar above the graph

---

**Note:** The breakdown menu options and buttons are not displayed until an element (transaction, server request, layer) is selected.

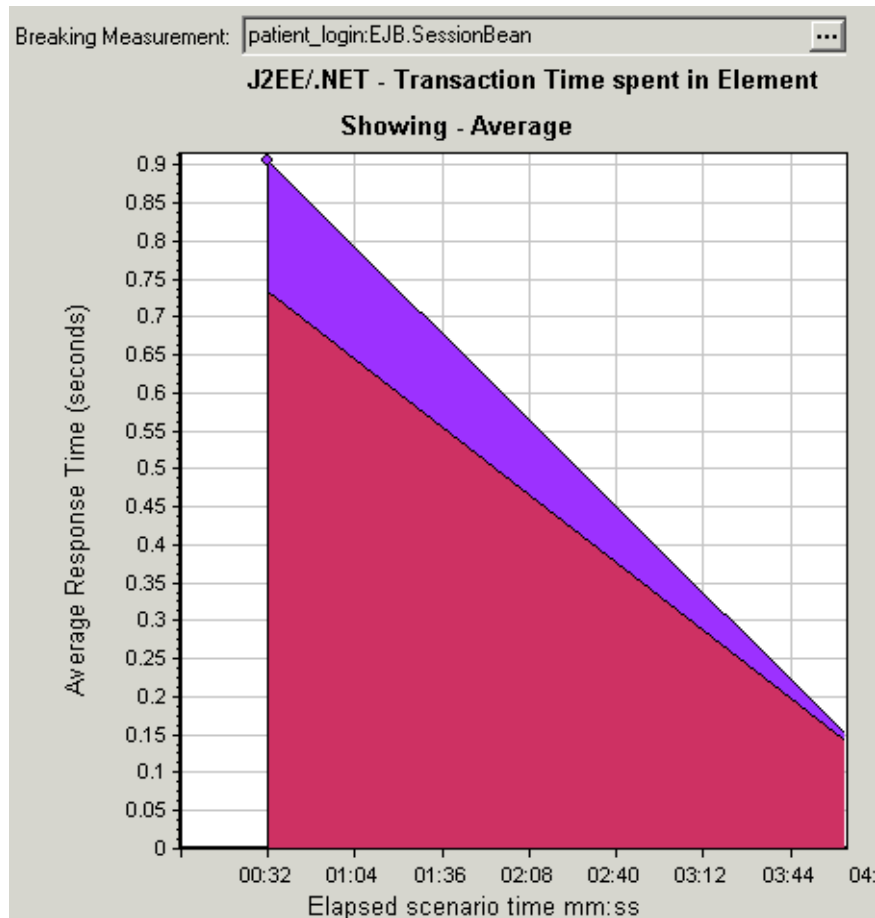
---

### To view server side diagnostics data:

- 1 From the Average Response Time graph, right-click a transaction line and choose **J2EE/.NET Diagnostics > Show Server Requests**, or choose **View > J2EE/.NET Diagnostics > Show Server Requests**. Alternatively, click the **Show Server Requests** button in the toolbar above the graph.



A new graph opens showing the breakdown of the selected transaction. The name of the transaction is displayed in the Breaking Measurement box.



You can view the full SQL statement for a selected SQL element by choosing **Show measurement description** from the Legend tab right-click menu. The Measurement Description dialog box opens displaying the name of the selected measurement and the full SQL statement.

---

**Note:** If there is no URI in the SQL, **URI-None** appears in front of the full measurement description in the Measurement Description dialog box.

---



To view transaction properties for the breakdown measurement, click the **Breaking Measurement** button. To disable this feature, choose **View > Display Options**, and clear the **Show Breaking Measurement** check box.

**2** At this point, you can select a displayed element and use the J2EE/.NET Diagnostics menu or buttons to do the following:

► Break the data down to a lower level by doing one of the following:



- Select **View > J2EE/.NET Diagnostics > Break down the server request to layers**, or click the measurement breakdown button in the toolbar above the graph.

---

**Note:** The option in the J2EE/.NET Diagnostics menu, and the tool tip of the measurement breakdown button, vary according to the element that you want to break down. For example, when you select a server request, the menu option and tool tip are **Break down server request to layers**.

---



- Select **View > J2EE/.NET Diagnostics > Show VM**, or click the **Show VM** button in the toolbar above the graph. This breaks the data down to the application host name (VM).

- Return to a previous level by doing one of the following:



- Select **View > J2EE/.NET Diagnostics > Undo Break down the server request to layers**, or click the **Undo** *<Measurement Breakdown>* button in the toolbar above the graph.

---

**Note:** The option in the J2EE/.NET Diagnostics menu, and the tool tip of the measurement breakdown button, vary according to the element whose breakdown you want to undo. For example, when you select a layer, the menu option and tool tip are **Undo break down server request to layers**.

---



- Select **View > J2EE/.NET Diagnostics > Hide VM**, or click the **Hide VM** button in the toolbar above the graph.
- Display the chain of call or call stack statistics in the measurements tree window: Drag the orange time line on to the graph to the time specifying the end of the period for which you want to view data, and choose



**View > J2EE/.NET Breakdown > Show Chain of Calls**, or click the **Show Chain of Calls** button in the toolbar above the graph.

---

**Note:** A measurement that is broken down in the Average Method Response Time in Transactions graph will be different from the same measurement broken down in the J2EE/.NET - Transaction Time Spent in Element graph. This is because the J2EE/.NET - Average Method Response Time in Transactions graph displays the average transaction time, whereas the J2EE/.NET - Transaction Time Spent in Element graph displays the average time per transaction event (sum of method execution time).

---

## Viewing Chain of Calls and Call Stack Statistics

You can view the chain of calls for transactions and methods. The chain of calls answers the question “Whom did I call?”

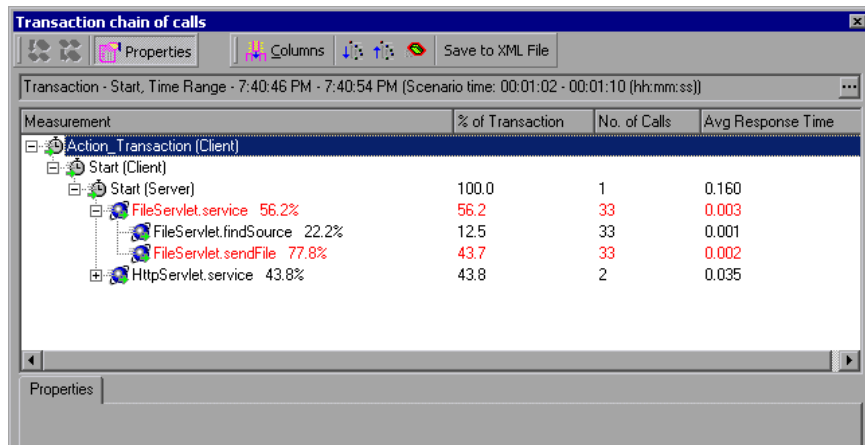
You can also view the call stack statistics for methods. Call stack statistics answer the question “Who called me?”

Chain of call and call stack statistics data are shown in the measurements tree window. The title of the window changes depending on which kind of data you are viewing.

- ▶ To set the point to which the measurements tree window relates, you must drag the orange time line to the desired spot.
- ▶ To view transaction call chains, right-click a component and choose **J2EE/.NET Breakdown > Show Chain of Calls**. The Chain of Calls window opens displaying the chain of calls from the parent transaction downwards.
- ▶ To view method statistics, in the Chain of Calls window right-click a method and choose **Show Method Chain of Calls** or **Show Method Call Stack Statistics**.

### The Chain of Calls Windows

You use the Chain of Calls window to view the components that the selected transaction or method called. In the following figure, all the calls in the critical path of the Start server-side transaction are displayed.



---

**Note:** Each red node signifies the most time consuming child of its parent.

---

You use the Call Stack Statistics window to view which components called the selected component. In the following figure, the **FileServlet.service** was called by Start (Server), which was called by Start (Client), and so on, down to the transaction at the bottom of the chain.

The screenshot shows the 'Method call stack statistics' window. The title bar reads 'Method call stack statistics'. Below the title bar are icons for 'Properties', 'Columns', and 'Save to XML File'. The main area displays a table with the following data:

Measurement	% of Root Method	No. of Calls	Avg Time Spent in Root
FileServlet.service 0.0%	100.0	33	0.003
Start (Server)	100.0	33	0.003
Start (Client)			
Action_Transaction (Client)			

Below the table is a 'Properties' section with the following details:

- Method name: service
- Class name: FileServlet
- Package name: weblogic.servlet.FileServlet

Summary statistics at the bottom:

- Percent of root method time: 100.0%
- Percent of called method time: 0.0%
- Average method response time: 0.003 seconds
- Total time spent: 0.090 seconds
- Number of calls: 33

A 'Close' button is located at the bottom right of the window.

## Understanding the Chain of Calls Window



**Switch to Method Chain of Calls:** When the call stack statistics data is displayed, displays the method chain of calls data (only if the root is a method).



**Switch to Method Call Stack Statistics:** When the method chain of calls data is displayed, displays the method call stack statistics data (only if the root is a method).



**Show Method Chain of Calls:** Displays the Chain of Calls window.



**Show Method Call Stack Statistics:** Displays the Call Stack Statistics window.



**Properties:** Hides or displays the properties area (lower pane).



**Columns:** Enables you to select the columns shown in the Calls window. To display additional fields, drag them to the desired location in the Calls window. To remove fields, drag them from the Calls window back to the Columns chooser.

The following columns are available in the Chain of Calls window:

Column	Description
Measurement	Name of the method, displayed as <b>ComponentName:MethodName</b> . In the case of a database call, query information is also displayed. The percent shown indicates the percentage of calls to this component from its parent.
% of Root Method	Percentage of the total time of the method from the total time of the root tree item.
No of Calls	Displays the amount of times this transaction or method was executed.
Avg Response Time	Response time is the time from the beginning of execution until the end. Average response time is the total response time divided by the number of divided by the number of instances of the method.
STD Response Time	The standard deviation response time.



Column	Description
<b>Min Response Time</b>	The minimum response time.
<b>Max Response Time</b>	The maximum response time.
<b>% of Caller</b>	Displays the percentage of method time in relation the parent method time.
<b>Total time</b>	Displays the total method execution time, including the child execution time.

The following columns are available in the Call Stack Statistics window:

Column	Description
<b>Measurement</b>	Name of the method, displayed as <b>ComponentName.MethodName</b> . In the case of a database call, query information is also displayed. The percent shown indicates the percentage of calls to this component from its child.
<b>% of Root Method</b>	Percentage of the total time of the transaction (or method) from the total time of the root tree item.
<b>No. of Calls to Root</b>	Displays the amount of times this transaction or method was executed.
<b>Avg Time Spent in Root</b>	Time spent in root is the time that the sub-area spent in the root sub-area/area/transaction. Average Time Spent in Root time is the total time spent in the root divided by the number of instances of the method.
<b>STD Time Spent in Root</b>	The standard deviation time spent in the root.
<b>Min Time Spent in Root</b>	The minimum time spent in the root.
<b>Max Time Spent in Root</b>	The maximum time spent in the root.
<b>% of Called</b>	Displays the percentage of method time in relation the child method time.
<b>Total Time Spent in Root</b>	Displays the total method execution time, including the child execution time.



**Expand All:** Expands the entire tree.



**Collapse All:** Collapses the entire tree.



**Expand Worst Path:** Expands only the parts of the path on the critical path.

**Save to XML File:** Saves the tree data to an XML file.

**Method Properties Area:** Displays the full properties of the selected method.

**SQL Query:** Displays the SQL query for the selected method. (For Database only.)

## Viewing J2EE to SAP R3 Remote Calls

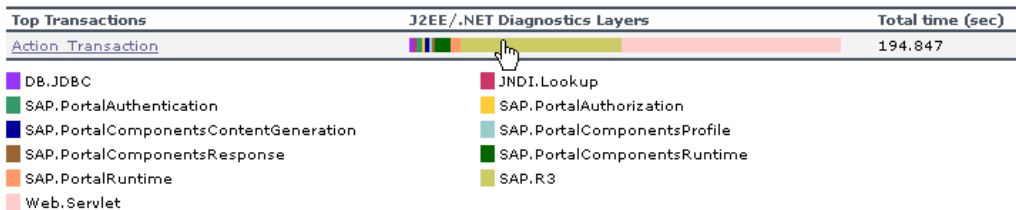
The *Remote Function Call (RFC)* protocol in SAP allows communication to take place between SAP J2EE and SAP R3 environments. When remote calls take place between SAP J2EE and SAP R3 environments, Analysis displays information about the RFC functions, including the name of each function.

You view information about RFC functions by breaking down the SAP R3 layer. You can view the RFC function information in a graph display or in the Chain Of Calls window.

**To view RFC function information in a graph display:**

- 1 Go to the **J2EE/.Net Diagnostics Usage** section of the Summary Report. Next to the relevant transaction, click the color representing the **SAP.R3** layer.

### J2EE/.NET Diagnostics Usage

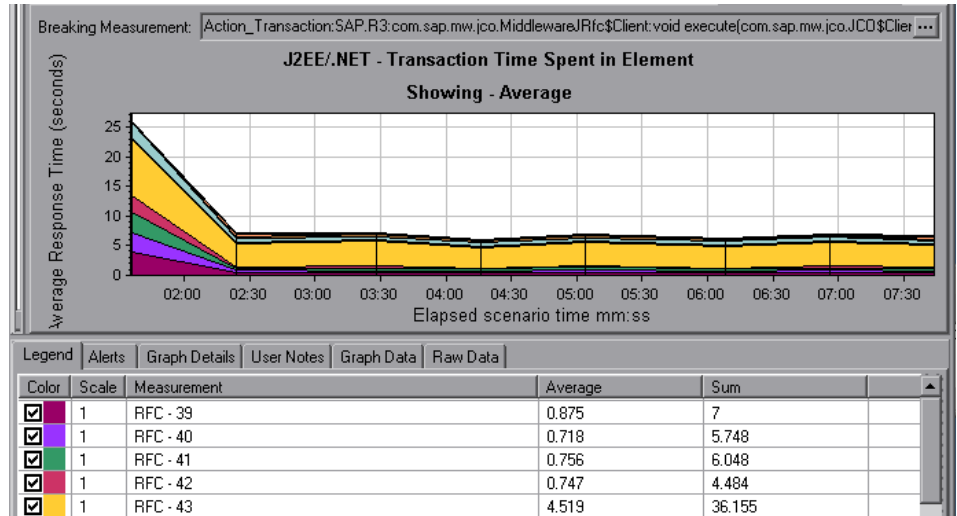


The J2EE/.NET - Transaction Time Spent in Element graph opens, representing the SAP.R3 layer.

- 2 Right click the graph and choose **J2EE/.NET Diagnostics > Break down the class to methods.**

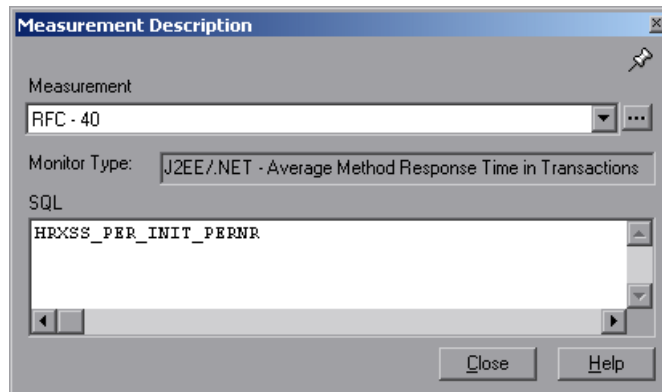
- Break down the graph further by right clicking the graph and choosing **J2EE/.NET Diagnostics > Break down the method to SQLs**.

The graph is broken down into the different RFC functions.



- To view the name of each RFC function, right click an RFC measurement in the **Measurement** column in the graph legend and choose **Show measurement description**.

The Measurement Description dialog box opens. The name of the RFC function is displayed in the **SQL** box.



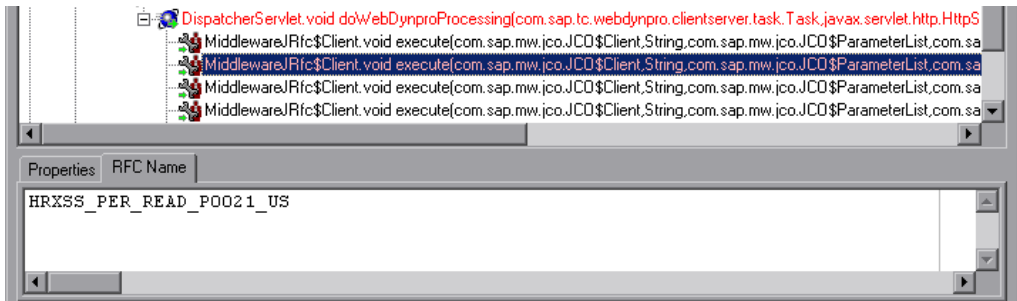
To view RFC function information in the Chain Of Calls window:

- 1 Go to the **J2EE/.Net Diagnostics Usage** section of the Summary Report. Next to the relevant transaction, click the color representing the **SAP.R3** layer.

The J2EE/.NET - Transaction Time Spent in Element graph opens, representing the SAP.R3 layer.

- 2 Right click the graph and choose **J2EE/.NET Diagnostics > Show chain of calls**.

The Transaction chain of calls window opens. When you click any of the RFC functions, in the **Measurement** column, the name of the function is displayed in the lower pane in the **RFC Name** tab.



## J2EE & .NET Diagnostics Graphs

The following J2EE & .NET Diagnostics graphs are available:

- J2EE/.NET - Transaction Response Time Server Side Graph
- J2EE/.NET - Average Method Response Time in Transactions Graph
- J2EE/.NET - Transaction Time Spent in Element Graph
- J2EE/.NET - Transactions per Second Graph
- J2EE/.NET - Method Calls per Second in Transactions Graph
- J2EE/.NET - Average Number of Exceptions in Transactions Graph
- J2EE/.NET - Average Number of Timeouts in Transactions Graph

---

**Note:** To obtain data for these graphs, you must enable the Mercury Diagnostics Server (from the Controller or Console) before running the scenario or session step.

---

## Setting Graph Filter Properties

You can filter the J2EE & .NET Diagnostics graphs so that the displayed data is more suitable to your needs. You can filter using the following methods:

- ▶ Before opening a graph, enter filter criteria in the **Graph Properties** box of the **Open Graph** dialog box. For more information, refer to the section “Opening Analysis Graphs” in the *Mercury LoadRunner Analysis User’s Guide*.
- ▶ From an open graph, enter filter criteria in the **Filter condition** fields in a filter dialog box. For more information, refer to the section “Filtering and Sorting Data” in the *Mercury LoadRunner Analysis User’s Guide*.

You can filter the J2EE & .NET Diagnostics graphs by the following fields:

**Scenario Elapsed Time:** Shows data for transactions that ended during the specified time.

**Transaction Name - J2EE/.NET:** Shows data for a specified transaction.

**Layer Name:** Shows data for specified layers.

**Class Name:** Shows data for specified classes.

**SQL Logical Name:** Shows data for specified SQL logical names. Due to the length of some SQL names, after you choose an SQL statement it is assigned a "logical name." This logical name is used in the filter dialog, legend, grouping, and other places in place of the full SQL statement. You can view the full SQL statement in the Measurement Description dialog box (**View > Show Measurement Description**).

Some JDBC methods have the ability to invoke SQL's (each method can invoke several different SQL's) so there is another level of breakdown which is the SQL statements.

---

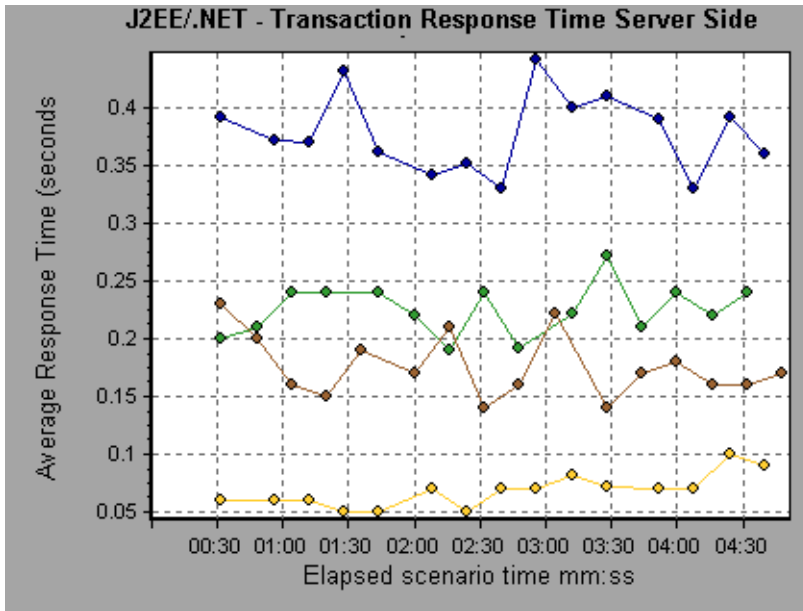
**Note:** For the methods that do not have SQL statement when reaching this level of breakdown you see **NoSql**.

---

### J2EE/.NET - Transaction Response Time Server Side Graph

The J2EE/.NET - Transaction Response Time Server Side graph displays the transaction server response time of transactions that include steps that cause activity on the J2EE/.NET backend. The reported times, measured from the point when the transaction reached the Web server to the point it left the Web server, include only the time that was spent in the J2EE/.NET backend.

The x-axis represents elapsed time. The y-axis represents the average response time (in seconds) of each transaction.

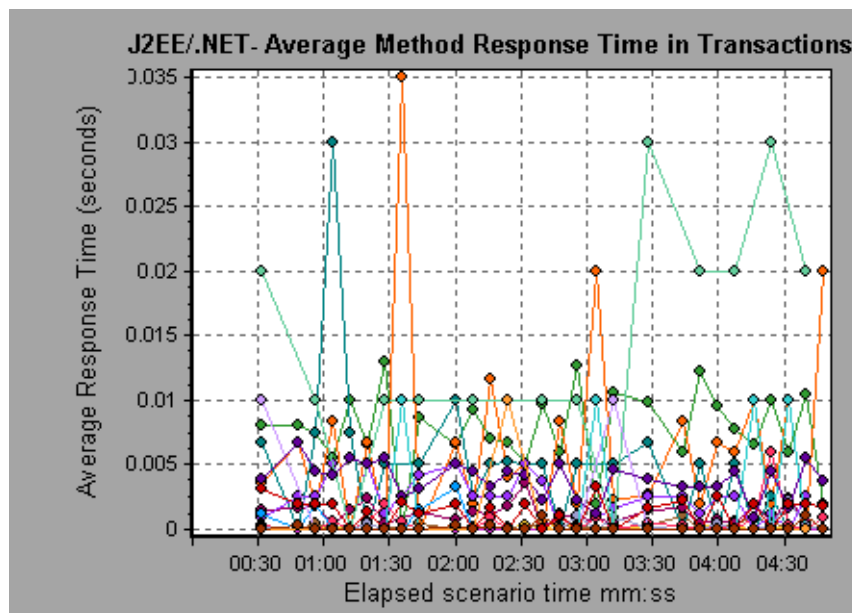


To break the displayed elements down further, see “Viewing J2EE & .NET Diagnostics Data” on page 276.

## J2EE/.NET - Average Method Response Time in Transactions Graph

The J2EE/.NET - Average Method Response Time in Transactions graph displays the average response time for the server side methods, computed as Total Method Response Time/Number of Method calls. For example, if a method was executed twice by an instance of transaction A and once by another instance of the same transaction, and it took three seconds for each execution, the average response time is  $9/3$ , or 3 seconds. The method time does not include calls made from the method to other methods.

The x-axis represents elapsed time. The y-axis represents the average response time (in seconds) per method.



To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Transaction Time Spent in Element Graph

The J2EE/.NET - Transaction Time Spent in Element graph displays the server response time for the selected element (layer, class, or method) within each transaction.

The display of graph data is determined by the graph properties selected when the graph was opened, as described in the following table:

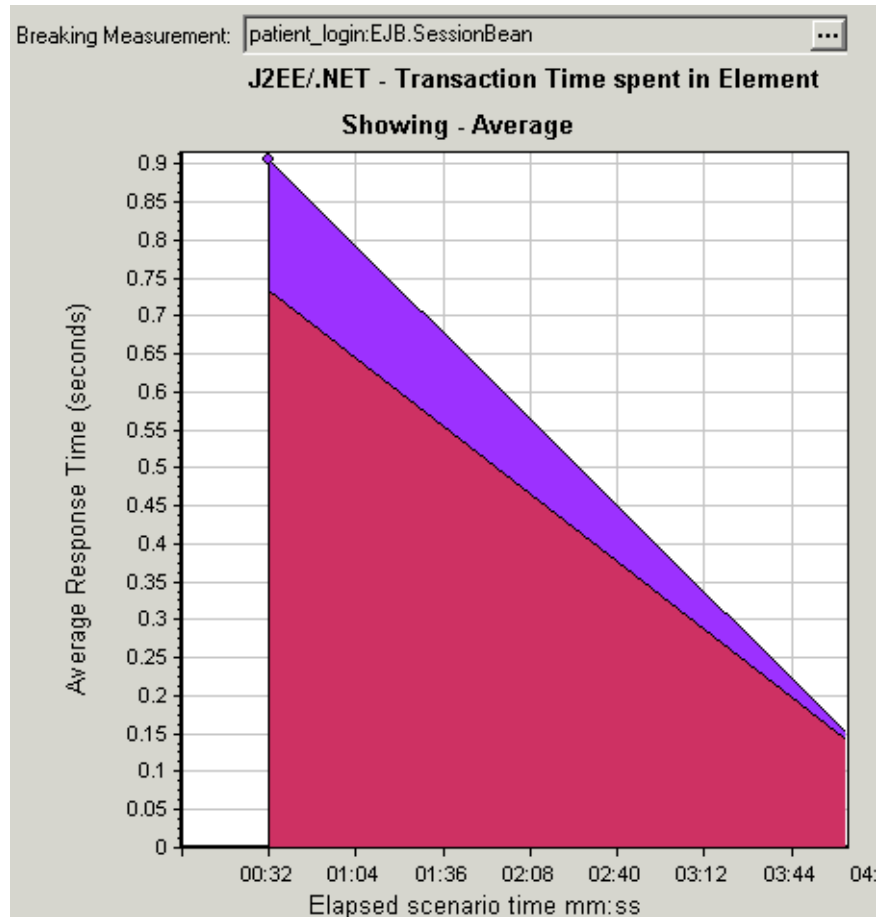
If you filter by these properties...	The graph data is displayed like this
None	Time spent in each transaction.
Transaction	Filtered by transaction. Grouped by layer.
Transaction and layer	Filtered by transaction and layer. Grouped by class.
Transaction, layer, and class	Filtered by transaction, layer, and class. Grouped by method.

For more information on filtering by graph properties, see “Setting Graph Filter Properties” on page 293.

The time is computed as Total Response Time/Total Number of Transactions. For example, if a method was executed twice by an instance of transaction A and once by another instance of the same transaction, and it took three seconds for each execution, the average response time is  $9/2$ , or 4.5 seconds. The transaction time does not include the nested calls from within each transaction.



The x-axis represents elapsed time. The y-axis represents the average response time (in seconds) per element within the transaction.



To obtain data for this graph, you must enable the J2EE & .NET Diagnostics module (from the Controller or Console) before running the scenario or test. For more information refer to Chapter 12, “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1” and Chapter 13, “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1.”

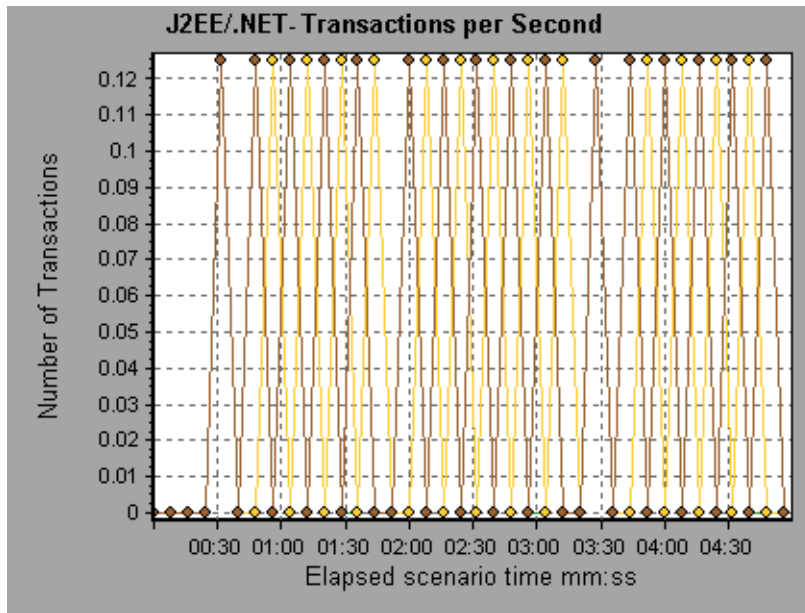
You can break down the displayed elements. For more information, see “Viewing J2EE & .NET Diagnostics Data” on page 276.

## J2EE/.NET - Transactions per Second Graph

The J2EE/.NET - Transactions per Second graph displays the number of completed sampled transactions during each second of a scenario run.

The number of transactions included in the sample is determined by the sampling percentage set in the Diagnostics Distribution dialog box in the Controller (**Diagnostics > Configuration**).

The x-axis represents elapsed time. The y-axis represents the number of completed sampled transactions per second.



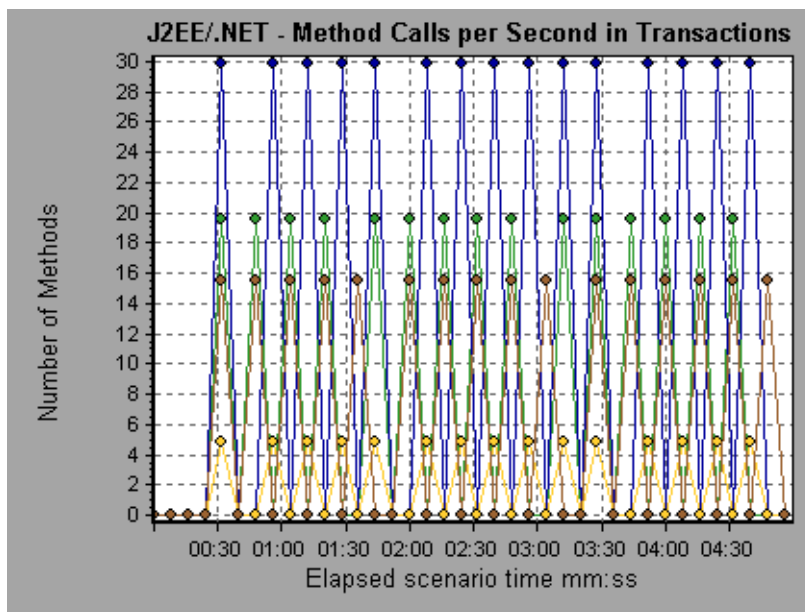
To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Method Calls per Second in Transactions Graph

The J2EE/.NET – Method Calls per Second in Transactions graph displays the number of completed sampled methods during each second of a scenario run.

The number of methods included in the sample is determined by the sampling percentage set in the Diagnostics Distribution dialog box in the Controller (**Diagnostics > Configuration**).

The x-axis represents elapsed time. The y-axis represents the number of completed sampled methods per second.

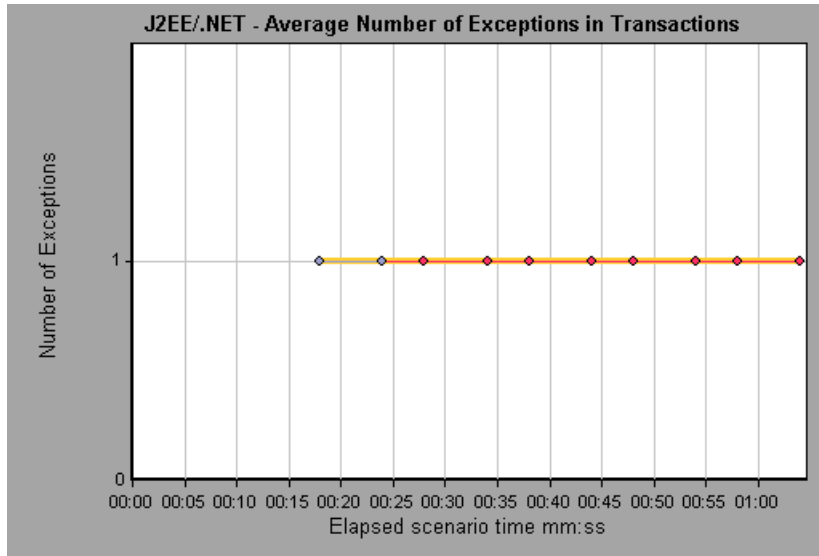


To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Average Number of Exceptions in Transactions Graph

The J2EE/.NET – Average Number of Exceptions in Transactions graph displays the average number of code exceptions per method, transaction, or request that were monitored during the selected time range.

The x-axis represents elapsed time. The y-axis represents the number of events.

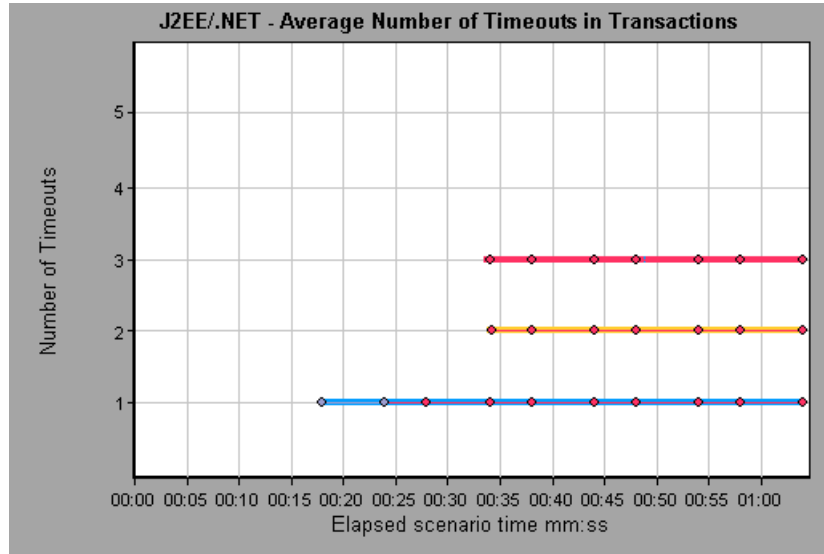


To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Average Number of Timeouts in Transactions Graph

The J2EE/.NET – Average Number of Timeouts in Transactions graph displays the average number of timeouts per method, transaction, or request that were monitored during the selected time range.

The x-axis represents elapsed time. The y-axis represents the number of events.



To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE & .NET Server Diagnostics Graphs

The following J2EE/.NET Server Diagnostics graphs are available:

- ▶ J2EE/.NET - Server Request Response Time Graph
- ▶ J2EE/.NET - Average Server Method Response Time Graph
- ▶ J2EE/.NET - Server Request Time Spent in Element Graph
- ▶ J2EE/.NET - Server Requests per Second Graph
- ▶ J2EE/.NET - Server Methods Calls per Second Graph
- ▶ J2EE/.NET - Average Number of Exceptions on Server Graph
- ▶ J2EE/.NET - Average Number of Timeouts on Server Graph

---

**Note:** To obtain data for these graphs, you must enable the Mercury Diagnostics Server (from the Controller or Console) before running the scenario or session step.

---

### Setting Graph Filter Properties

You can filter the J2EE/.NET Server Diagnostics graphs so that the displayed data is more suitable to your needs. You can filter using the following methods:

- ▶ Before opening a graph, enter filter criteria in the **Graph Properties** box of the **Open Graph** dialog box. For more information, refer to the section “Opening Analysis Graphs” in the *Mercury LoadRunner Analysis User’s Guide*.
- ▶ From an open graph, enter filter criteria in the **Filter condition** fields in a filter dialog box. For more information, refer to the section “Filtering and Sorting Data” in the *Mercury LoadRunner Analysis User’s Guide*.

You can filter the J2EE/.NET Server Diagnostics graphs by the following fields:

**Scenario Elapsed Time:** Shows data for requests that ended during the specified time.

**Server Request:** Shows data for a specified request.

**Layer Name:** Shows data for specified layers.

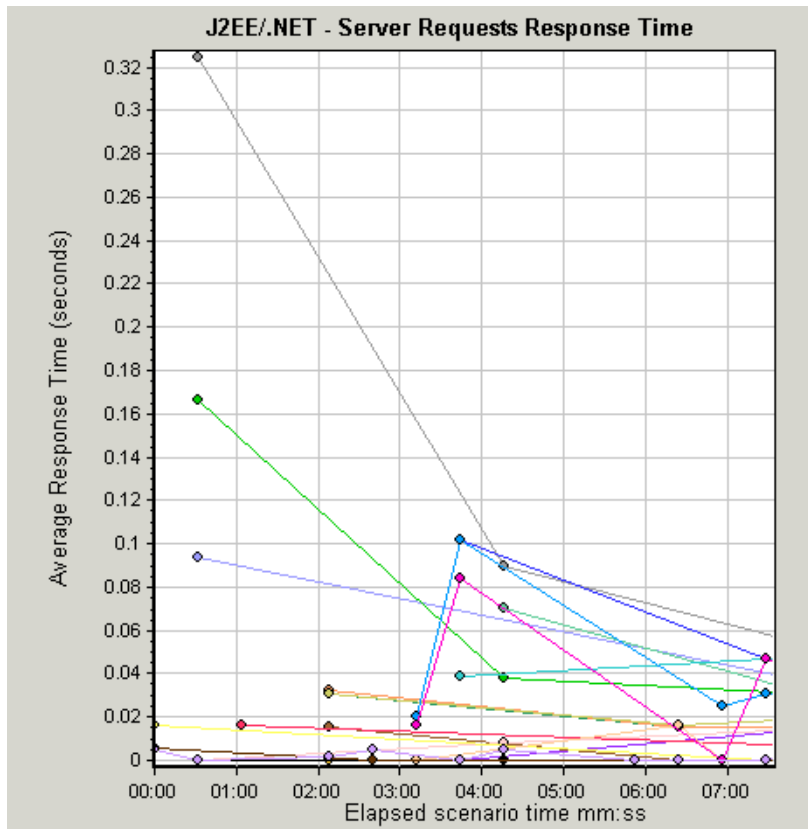
**Class Name:** Shows data for specified classes.

**SQL Logical Name:** Shows data for specified SQL logical names. Due to the length of some SQL names, after you choose an SQL statement it is assigned a “logical name.” This logical name is used in the filter dialog, legend, grouping, and other places in place of the full SQL statement. You can view the full SQL statement in the Measurement Description dialog box (**View > Show Measurement Description**).

### J2EE/.NET - Server Request Response Time Graph

The J2EE/.NET - Server Request Response Time graph displays the server response time of requests that include steps that cause activity on the J2EE/.NET backend. The reported times, measured from the point when the request reached the Web server to the point it left the Web server, include only the time that was spent in the J2EE/.NET backend.

The x-axis represents elapsed time. The y-axis represents the average time (in seconds) taken to perform each request.



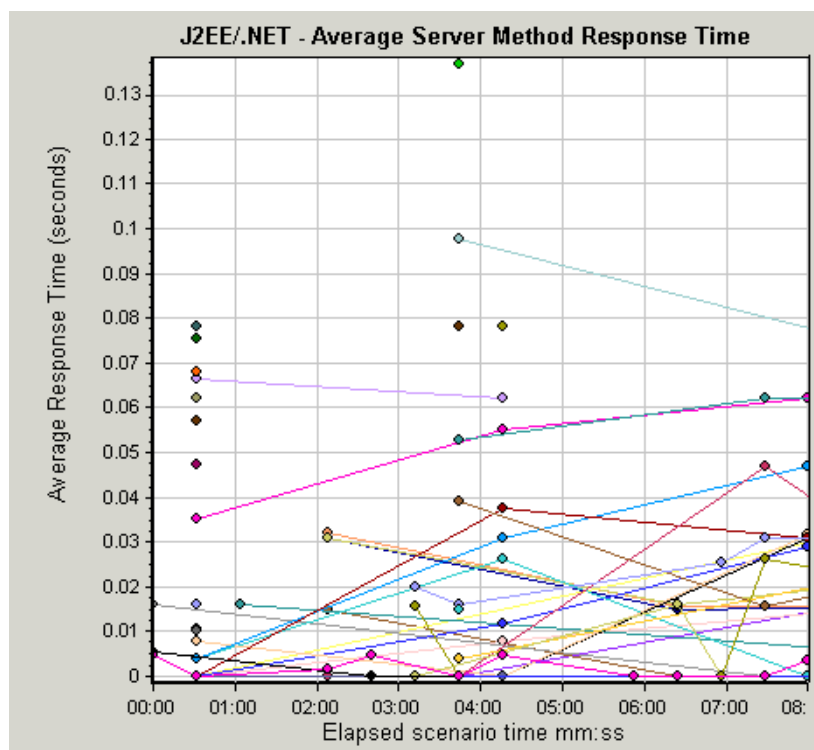
To break the displayed elements down further, see “Viewing J2EE & .NET Diagnostics Data” on page 276.



## J2EE/.NET - Average Server Method Response Time Graph

The J2EE/.NET - Average Server Method Response Time graph displays the average response time for the server side methods, computed as Total Method Response Time/Number of Method calls. For example, if a method was executed twice by an instance of transaction A and once by another instance of the same transaction, and it took three seconds for each execution, the average response time is  $9/3$ , or 3 seconds. The method time does not include calls made from the method to other methods.

The x-axis represents elapsed time. The y-axis represents the average response time (in seconds) per method.



To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Server Request Time Spent in Element Graph

The J2EE/.NET - Server Request Time Spent in Element graph displays the server response time for the selected element (layer, class, or method) within each server request.

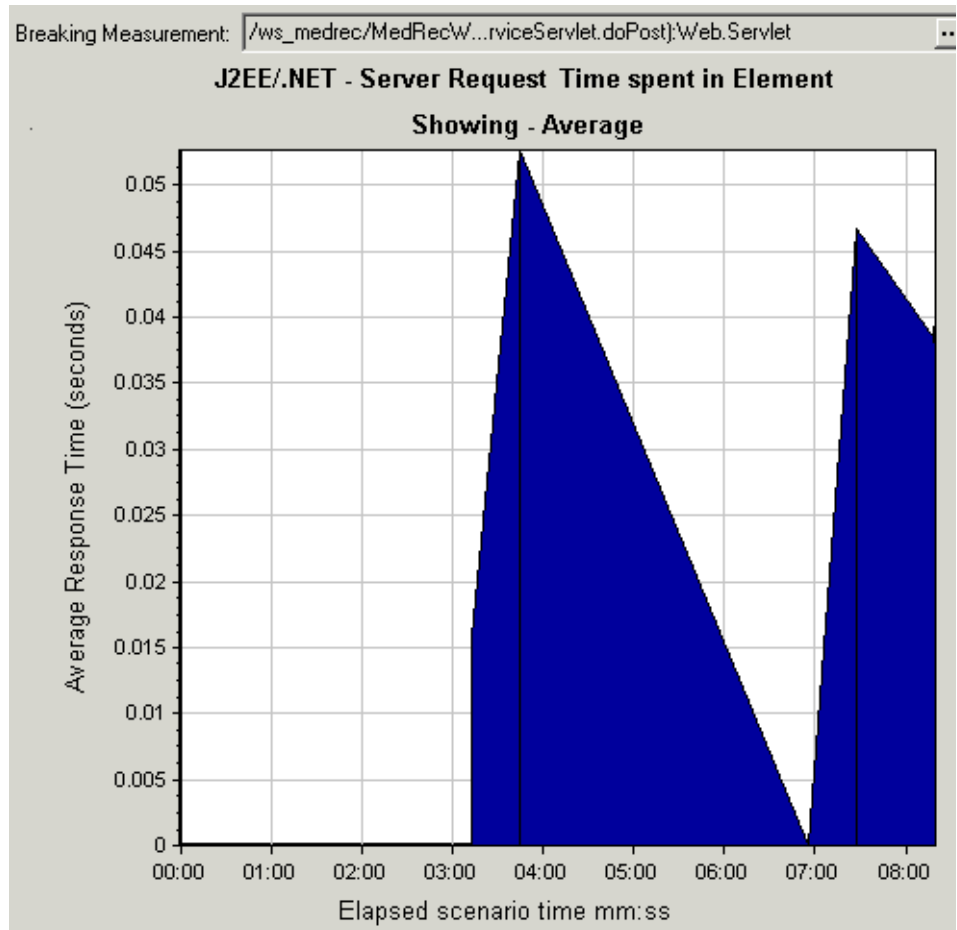
The display of graph data is determined by the Graph Properties selected when the graph was opened, as described in the following table:

If you filter by these properties	The graph data is displayed like this
None	Time spent in each server request.
Server request.	Filtered by server request. Grouped by layer.
Server request and layer	Filtered by server request and layer. Grouped by class.
Server request, layer, and class	Filtered by server request, layer, and class. Grouped by method.

For more information on filtering by graph properties, see “Setting Graph Filter Properties” on page 293.

The time is computed as Total Response Time/Total Number of Server Requests. For example, if a method was executed twice by an instance of server request A and once by another instance of the same server request, and it took three seconds for each execution, the average response time is  $9/2$ , or 4.5 seconds. The server request time does not include the nested calls from within each server request.

The x-axis represents elapsed time. The y-axis represents the average response time (in seconds) per element within the server request.



To obtain data for this graph, you must enable the J2EE & .NET Diagnostics module (from the Controller or Console) before running the scenario or session step.

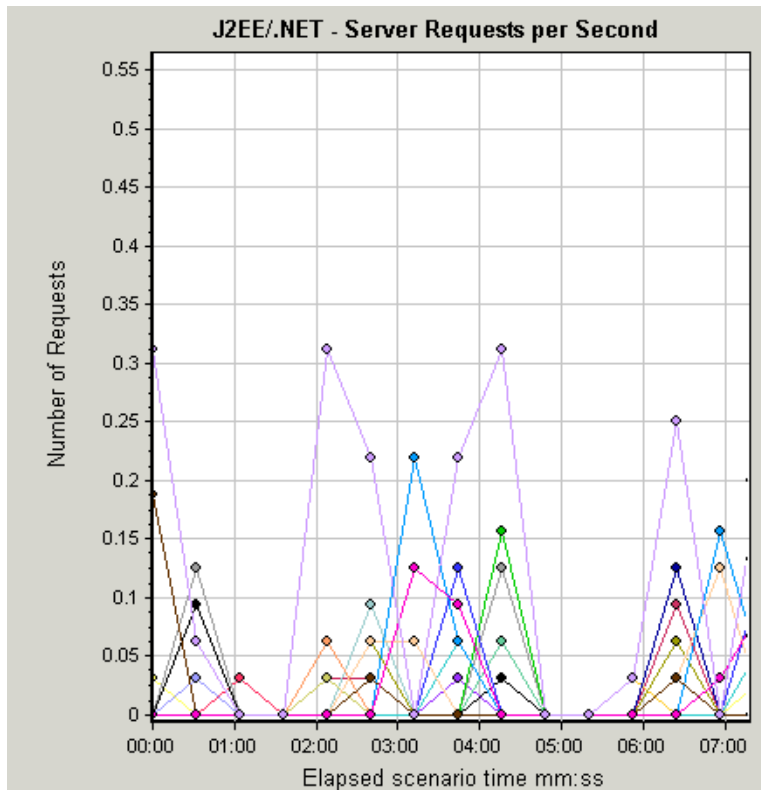
You can break down the displayed elements. For more information, see “Viewing J2EE & .NET Diagnostics Data” on page 276.

### J2EE/.NET - Server Requests per Second Graph

The J2EE/.NET - Server Requests per Second graph displays the number of completed sampled requests during each second of a scenario run.

The number of requests included in the sample is determined by the sampling percentage set in the Diagnostics Distribution dialog box in the Controller (**Diagnostics > Configuration**).

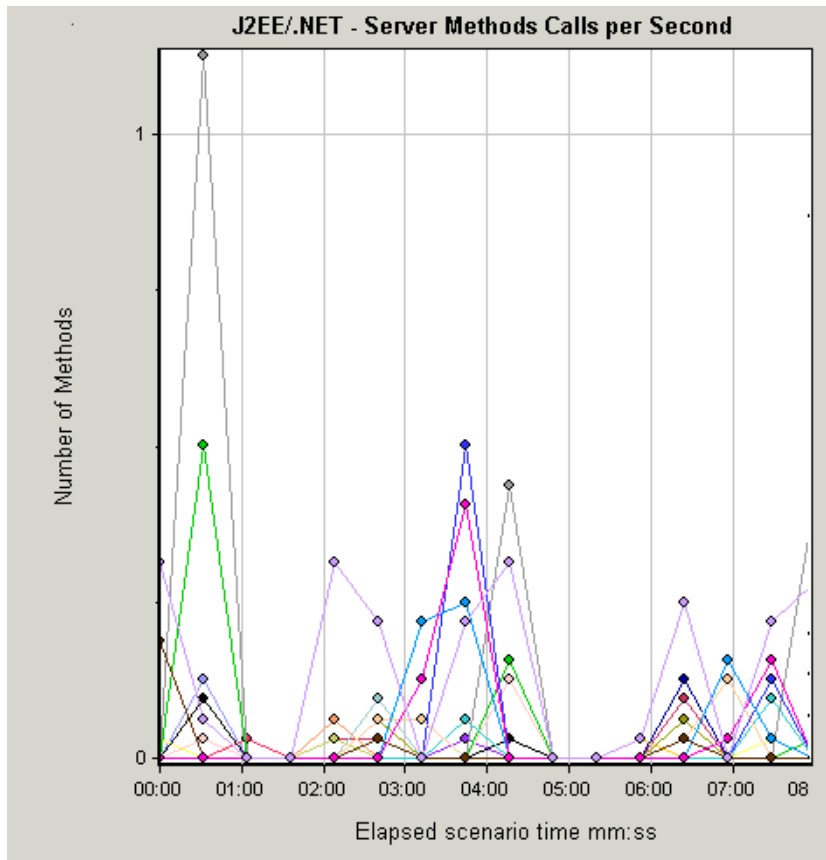
The x-axis represents elapsed time. The y-axis represents the number of completed sampled requests per second.



To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

## J2EE/.NET - Server Methods Calls per Second Graph

The J2EE/.NET – Server Methods Calls per Second graph displays the number of completed sampled methods during each second of a scenario run. The number of methods included in the sample is determined by the sampling percentage set in the Diagnostics Distribution dialog box in the Controller (**Diagnostics > Configuration**). The x-axis represents elapsed time. The y-axis represents the number of completed sampled methods per second.

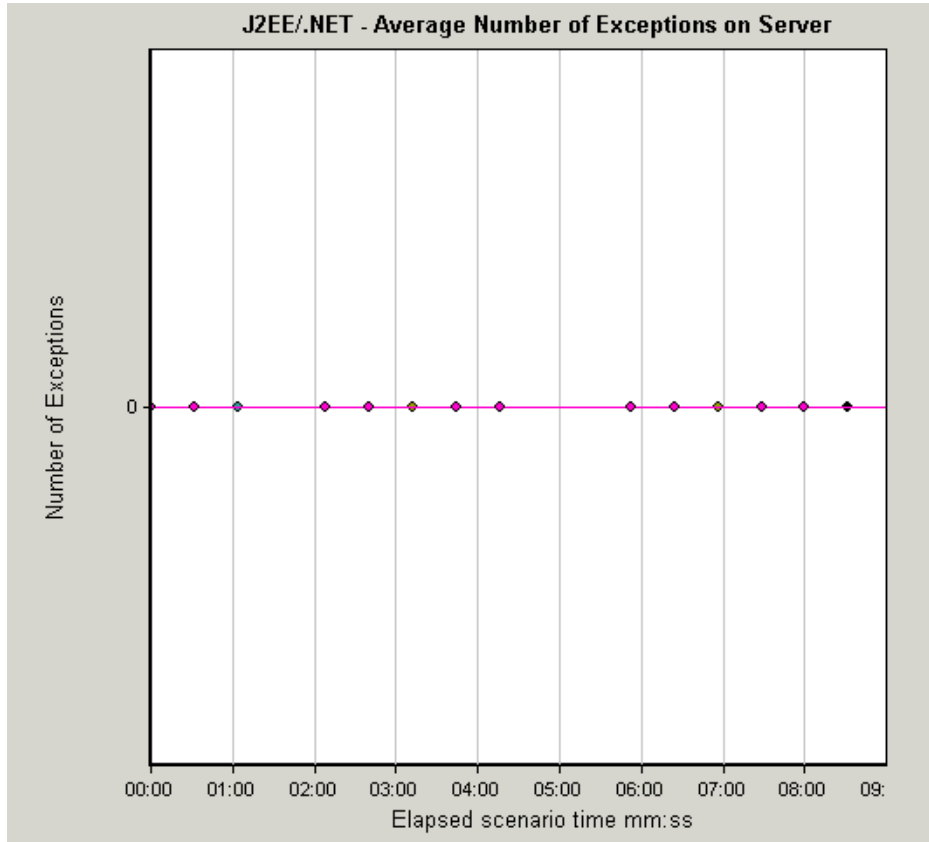


To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

### J2EE/.NET - Average Number of Exceptions on Server Graph

The J2EE/.NET – Average Number of Exceptions on Server graph displays the average number of code exceptions per method that were monitored during the selected time range.

The x-axis represents elapsed time. The y-axis represents the number of events.

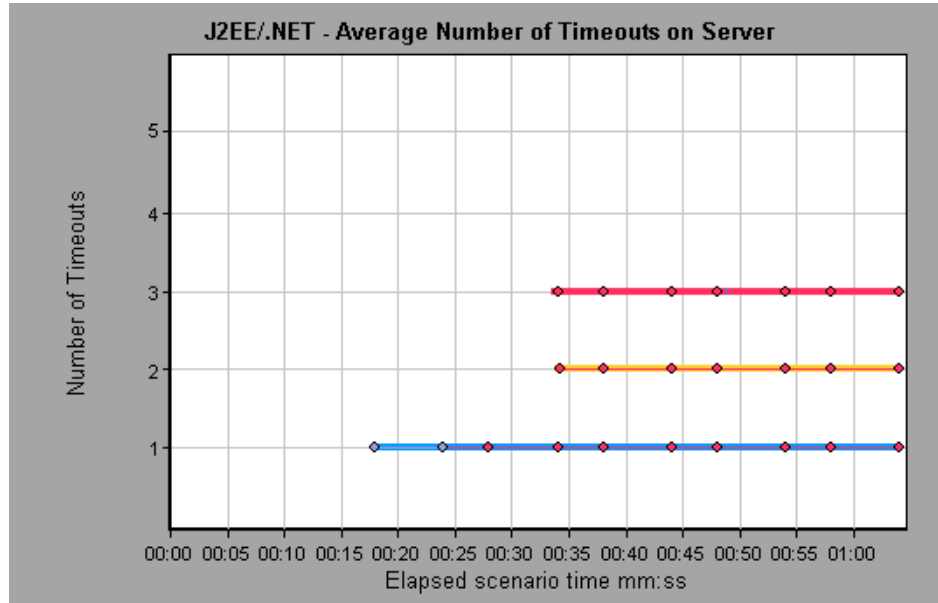


To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.

### J2EE/.NET - Average Number of Timeouts on Server Graph

The J2EE/.NET – Average Number of Timeouts on Server graph displays the average number of timeouts per method that were monitored during the selected time range.

The x-axis represents elapsed time. The y-axis represents the number of events.



To break the displayed elements down further, see “Using the J2EE & .NET Breakdown Options” on page 282.





# 17

---

## Web Service Support

This chapter describes the Web Service support in Mercury Diagnostics for J2EE & .NET.

This chapter includes the following sections:

- ▶ About Web Service Support
- ▶ Writing VuGen Scripts for Web Service Support

### About Web Service Support

The term *Web Services* describes self-contained applications that can run universally across the Internet. Using Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP), the Web Services serve as building blocks for the rapid development and deployment of new applications.

Since all communication is in XML, Web Services are not limited to a specific operating system or programming language. Web Services, therefore, allow applications from various sources to communicate with each other without extra coding and without intimate knowledge of each other's IT systems behind the firewall.

Mercury Diagnostics for J2EE & .NET Web Service support allows you to view J2EE and .NET breakdowns of virtual transactions recorded in VuGen scripts using the Web Service protocol. By looking at all the virtual transaction views in Diagnostics for J2EE & .NET, you can tell which Web service is the slowest, and you can analyze its performance problems.

Using a Web Services VuGen script with Diagnostics requires installing the Web Services patch on the Load Generator machine. The patch is located in the Diagnostics CD Patches directory.

---

**Note:** Differentiation between Web services can be performed only by defining relevant virtual transactions.

---

## **Writing VuGen Scripts for Web Service Support**

To enable breakdown of Web Service transactions, in VuGen create a script with a separate transaction for each Web Service that you want to monitor.

# 18

---

## Troubleshooting Diagnostics for J2EE & .NET

This chapter describes how to solve problems that may occur when using Diagnostics for J2EE & .NET. It includes:

- ▶ Component Installation Interrupted on a Solaris Machine
- ▶ Version Mismatch Between Diagnostics Commander and LoadRunner Add-In
- ▶ J2EE Probe Fails to Operate Properly

## Component Installation Interrupted on a Solaris Machine

If a component installer on a Solaris machine is interrupted before it has finished the installation, 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`.

## Version Mismatch Between Diagnostics Commander and LoadRunner Add-In

The version number of the Diagnostics Commander and the LoadRunner Add-In must be the same. If they are not the same, you will see the following error message when you view the Diagnostics Screens in the LoadRunner Controller:

This version of the Mercury J2EE/.NET Diagnostics user interface, <version\_nbr>, may not be compatible with the Diagnostics Commander version <version\_nbr>. Please use the version of the user interface which came with the Diagnostics Commander.

The versions may not match when the Diagnostics Commander and LR are running on different host machines and you have upgraded the Commander but have not upgraded the LoadRunner installation with the new Diagnostics Add-In.

If you see this message, you must install the correct version of the LoadRunner Add-in. See “Installing LoadRunner 8.1 and the LoadRunner Diagnostics Add-in” on page 49.

## **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 “Running the JRE Instrumenter” on page 141.

# **Part V**

---

## **Appendixes**





# A

---

## Using the System Health Monitor

This appendix describes how to use the System Health Monitor to review the configuration of the Diagnostics for J2EE & .NET components and verify that they are working properly.

This appendix contains the following sections:

- ▶ Introducing the System Health Monitor
- ▶ Accessing the System Health Monitor
- ▶ Using the System Health Monitor
- ▶ Drilling Down Into The System Health Monitor Map
- ▶ Customizing the System Health Monitor Display
- ▶ Creating and Using System Health Monitor Snapshots
- ▶ Interpreting Odd Situations Displayed on the System Health Monitor

## Introducing the System Health Monitor

The System Health Monitor provides you with a map of all of the components of your Mercury Diagnostics for J2EE & .NET deployment and gives you real-time status and health information 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 into the System Health Monitor map, you can reveal the details about the configuration, performance metrics, and processing logs for the components. You can also find trouble shooting tips for handling many of the issues that are revealed in the System Health Monitor map. In many cases, the System Health Monitor will be your first and your only stop when you need to know information about the components in your Diagnostics Deployment and the machines that host them.

You can 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, or from the LoadRunner Controller.

### **To Access the System Health Monitor from your Web browser:**

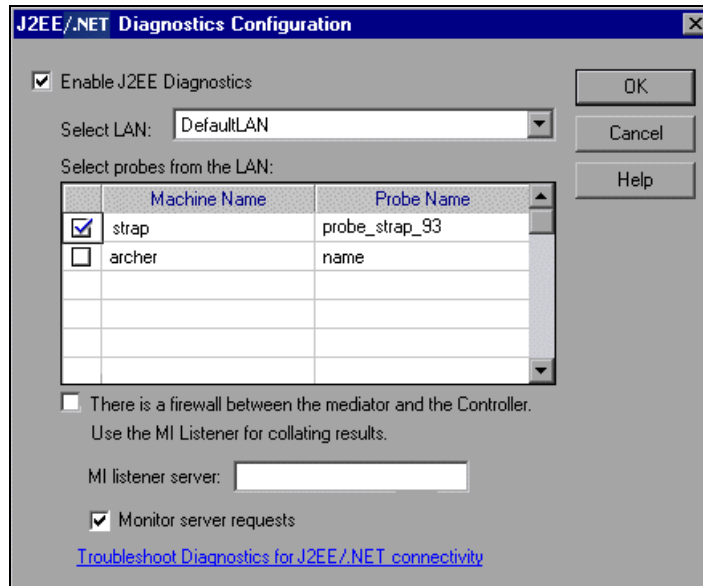
Enter the following URL into your browser:

`http://<commander host>:<commander port>/registrar/health`

The System Health Monitor opens in your browser.

### To Access the System Health Monitor from LoadRunner Controller:

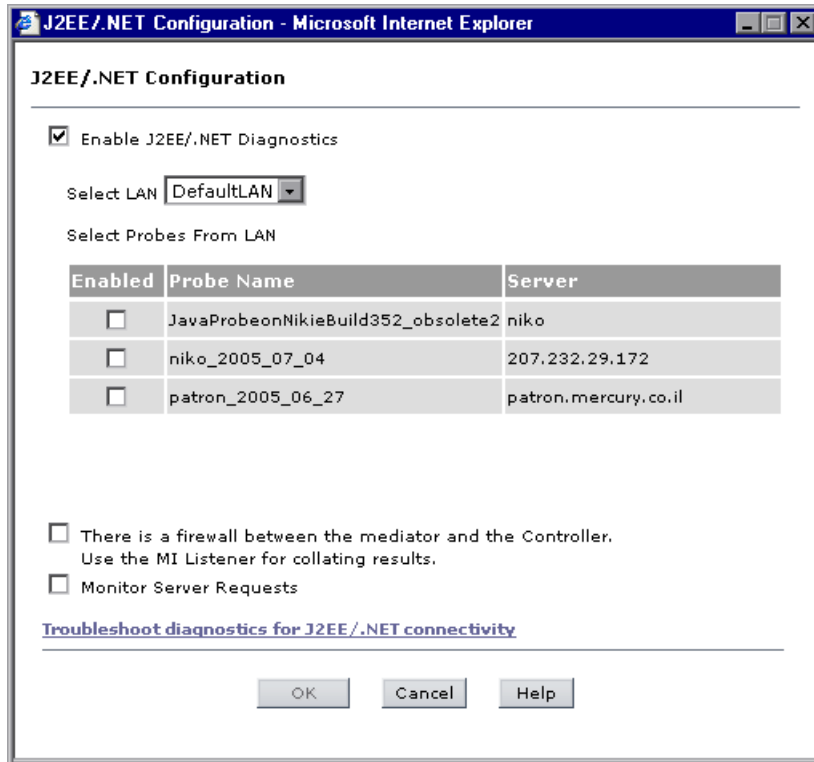
- 1 Choose **Diagnostics > Configuration** from the menu bar in the Controller Window. The Diagnostics Distribution dialog box opens.
- 2 In the Online and Offline Diagnostics section of the Diagnostics Distribution dialog, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.



- 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 Diagnostics 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 Problems Accessing the System Health Monitor**

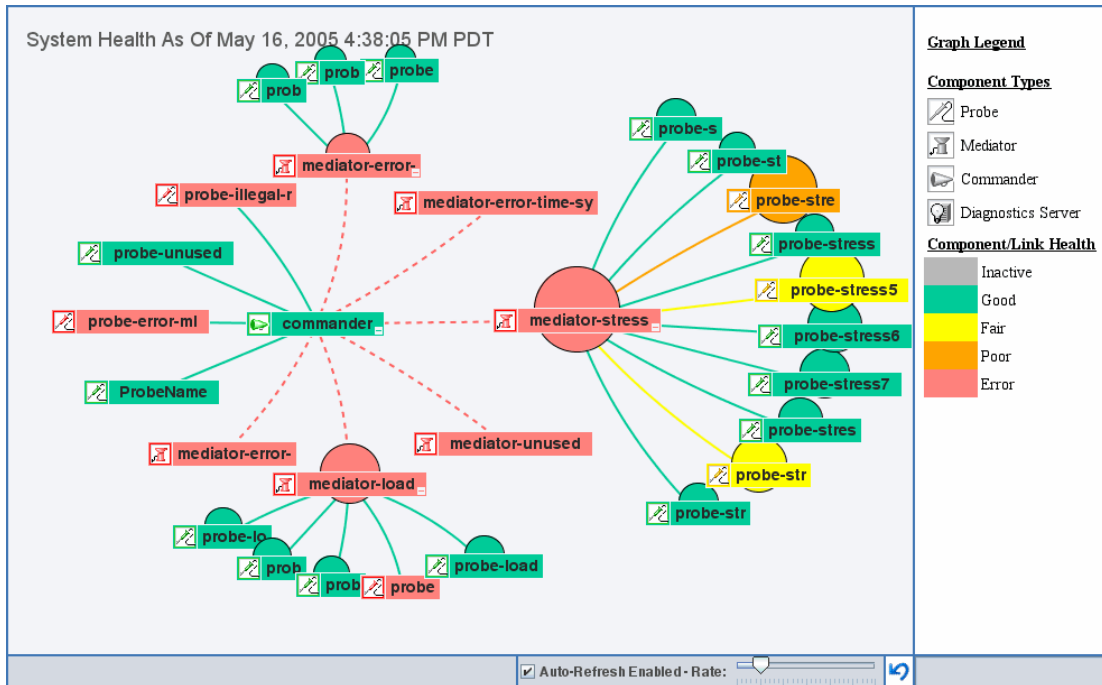
If you are unable to access the System Health Monitor verify that:

- You specified the correct machine name for the Commander host.
- You specified the correct port number for Commander communications. The default port is 2006.
- The Commander was successfully started and is running on the host machine.

## Using the System Health Monitor

### Interpreting the System Health Monitor Component Map

When the System Health Monitor opens you see a map of your deployment of the Diagnostics for J2EE & .NET system components.

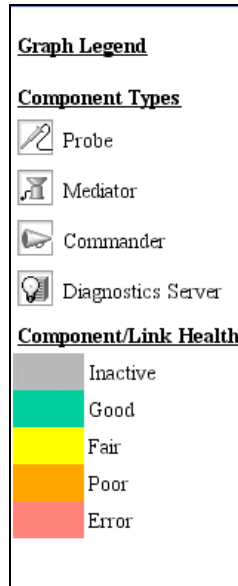


The System Health Monitor component map is made up of the following:

- ▶ 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.
- ▶ A “wafer” behind the component icon that indicates the amount of load that the component is processing.

- Links between the component icons that indicate the state of the communication links between components. The color of the link indicates the amount of traffic between the components.

The Graph Legend to the right of the component map helps you to interpret the information displayed in the component map:



Idle Probes query the Commander for available Mediators on their LAN every 90 seconds by default, and then try a handshake with each of the Mediators. If one or more of the handshakes is not successful, the probe turns red.

Similarly, the Commander turns red if it cannot reach any idle probe. The default time period is 90 seconds.

## Controlling the Refresh of the System Health Monitor

By default, the System Health Monitor is configured to automatically refresh the information displayed. You may change the frequency of the auto-refresh or completely disable the auto-refresh feature. You may also request a manual refresh at any time.

The controls on the System Health Monitor that are used to control the refresh of the information displayed are shown below.



### To disable the automatic refresh feature:

Toggle the **Auto-Refresh Enabled** check box so that it is not selected.

### To enable the automatic refresh feature:

Toggle the **Auto-Refresh Enabled** check box so that it is selected.

### 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.



## Drilling Down Into The System Health Monitor Map

The System Health Monitor deployment map gives you high level information about your Diagnostics deployment and how well the components are performing. It also provides several ways for you to drill down into the components on the map to discover the details of the configuration of the components and the performance of the components as they process the loads.

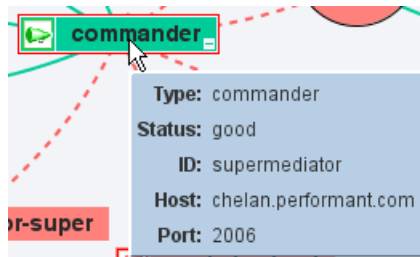
- ▶ Mouse-over tool tips for each component provide you with basic status and configuration information for the component.
- ▶ The Component Monitor Detail table list the processing metrics, component configuration, and log messages for the selected component. The Component Monitor Detail table can be accessed by double clicking on a component icon or by navigating through the right-click menu for the component.
- ▶ Troubleshooting Tips for each component are available through the right-click menu for the component.

### Viewing Component Status and Host Configuration Tool Tips

The status and the basic host configuration for a component can be revealed by activating the tool tip for the component. To activate the tool tip for a component, hold the mouse over the component icon until a tool tip is displayed.

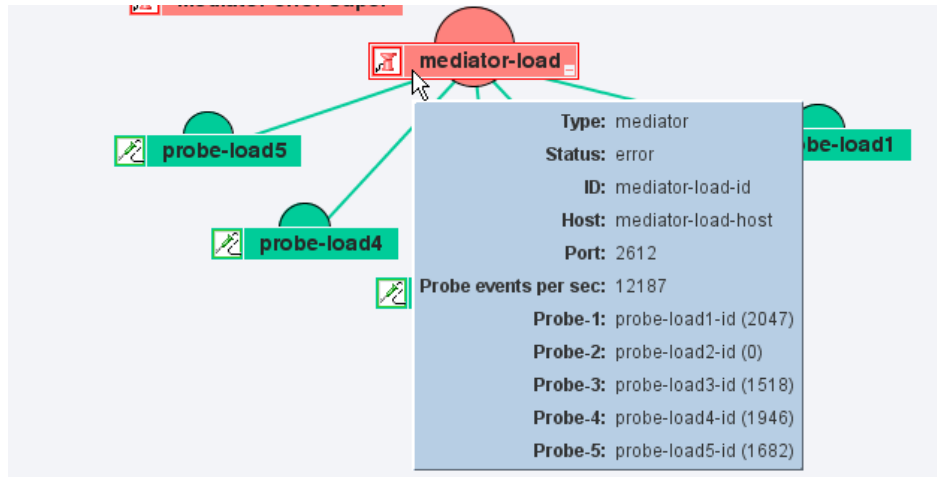
The following images are examples of the tool tips that are displayed for each of the types of components.

### Commander Properties



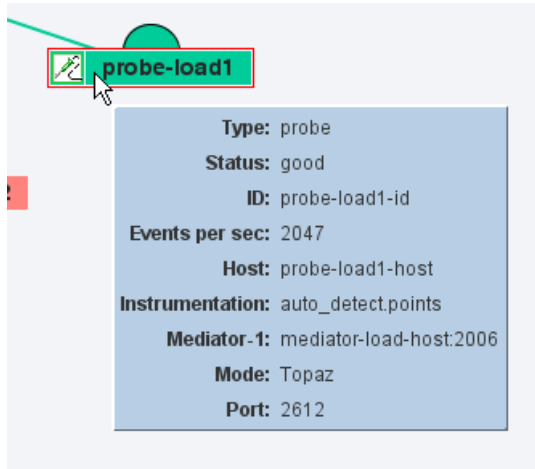
### Mediator Properties

The information in the tool tip for a mediator component includes a list of the Probes that have been configured to work with the component.



## Probe Properties

The information in the tool tip for a probe component includes the Mediator that the Probe is working with, the mode that the probe has been configured to work in.



## Introducing the Component Monitor Detail Table

If you need more information about a component than what is provided in the mouse-over tool tip, you may drill down further using the Component Monitor Detail table. The Component Monitor Detail is divided into three sections: Metrics, Configuration, and Log as described below.

### Component Monitor - Metrics

The following is an example of the Metrics portion of the Component Monitor detail table for a Mediator component.

Metric	
BPM transactions	0
BPM transactions since launch	0
Diagnostics samples dropped	0
Diagnostics samples dropped since launch	0
Diagnostics samples published offline	0
Diagnostics samples published offline since launch	3023
Diagnostics samples published online	0
Diagnostics samples published online since launch	0
ELT count since launch	2851
Last update (commander time)	Mon Apr 25 12:22:13 PDT 2005
Load Runner transactions	0
Load Runner transactions since launch	3034
Probe events per sec	0
Probe events since launch	387210
Time skew	-14 sec (behind commander)
Up time	2 day(s) 13 hour(s) 1 minute(s) 55 second(s)

## Component Monitor - Configuration

The following is an example of the Metrics portion of the Component Monitor detail table for a Mediator component.

⊗ Configuration	
Customer name	Default Client
Diagnostics Server	<unknown>
Diagnostics Time Synch	true
Host	repp
IP Address	10.241.3.72
Install dir	C:\Program Files\Mercury Interactive\Diagnostics\Mediator
LAN ID	DefaultLAN
Log file-1	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator.log
Log file-2	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator-full.log
Log file-3	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator-metrics.log
Log file-4	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator-system-properties.log
Log file-5	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator-out
Log file-6	C:\Program Files\Mercury Interactive\Diagnostics\Mediator\log\mediator-err
Port	2612
Supermediator	http://issaquah.performant.com:2006/supermediator
System	x86 Windows 2000 5.0 Service Pack 4 (en Cp1252)
URL	http://repp:8081
Version	3.5.12.352

## Component Monitor - Log

The following is an example of the Log portion of the Component Monitor detail table for a Mediator component.

⊗ Log	
Mon Apr 25 10:53:37 PDT 2005	SEVERE: Mediator closed bucket before VU data arrived.
Mon Apr 25 10:53:37 PDT 2005	WARNING: The Super Mediator did not send Mediator it's VU information for time bucket (Diagnos
Mon Apr 25 10:08:34 PDT 2005	WARNING: Probe Medical_J2EE_1 is dropping events. Details are logged at DEBUG level

## Accessing the Component Monitor Detail Table

There are two ways to access the Component Monitor Detail for a component: through the right-click menu for the component and by double clicking the component.

**To view access the component monitor details from a component's right-click menu:**

- 1 Right-click on a component icon to cause the popup menu for the component to be displayed as shown in the following example:



- 2 Select menu item for the area of the Component Monitor detail that you would like to see first. When the Component Monitor detail is displayed, only the section of the table that corresponds to your menu selection will be displayed.

**To access the component monitor details:**

Double click icon for the component to cause the Component Monitor Detail table is displayed in a separate browser page. All three sections of Component Monitor Detail are displayed.

## Viewing Troubleshooting Tips

**To view tips for troubleshooting problems with Diagnostics components:**

In the right-click menu, choose **View Troubleshooting Tips**. Troubleshooting information for the selected component is displayed.

Scan through 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:

- 1 Right-click the System Health screen (anywhere except a component). The following menu is displayed:



- 2 Choose **View Log History**. The log for all component activity is displayed.

TIME	NODE ID	LOG MESSAGE
Fri Apr 22 22:04:58 PDT 2005	mediator-repp	WARNING: Probe Medical_J2EE_1 is dropping events. Details are logged at DEBUG level
Fri Apr 22 21:59:31 PDT 2005	mediator-issaquah	WARNING: 'Trust All' security policy configured. No Certificates will be validated.

Component monitor will auto-refresh with health display...

Java Applet Window

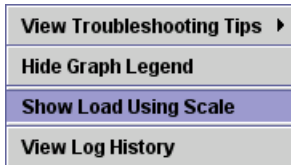
## Customizing the System Health Monitor Display

You can customize the way that information is displayed on the System Health Monitor. The two configuration options are made available on the right-click menu for the background of the System Health Monitor.

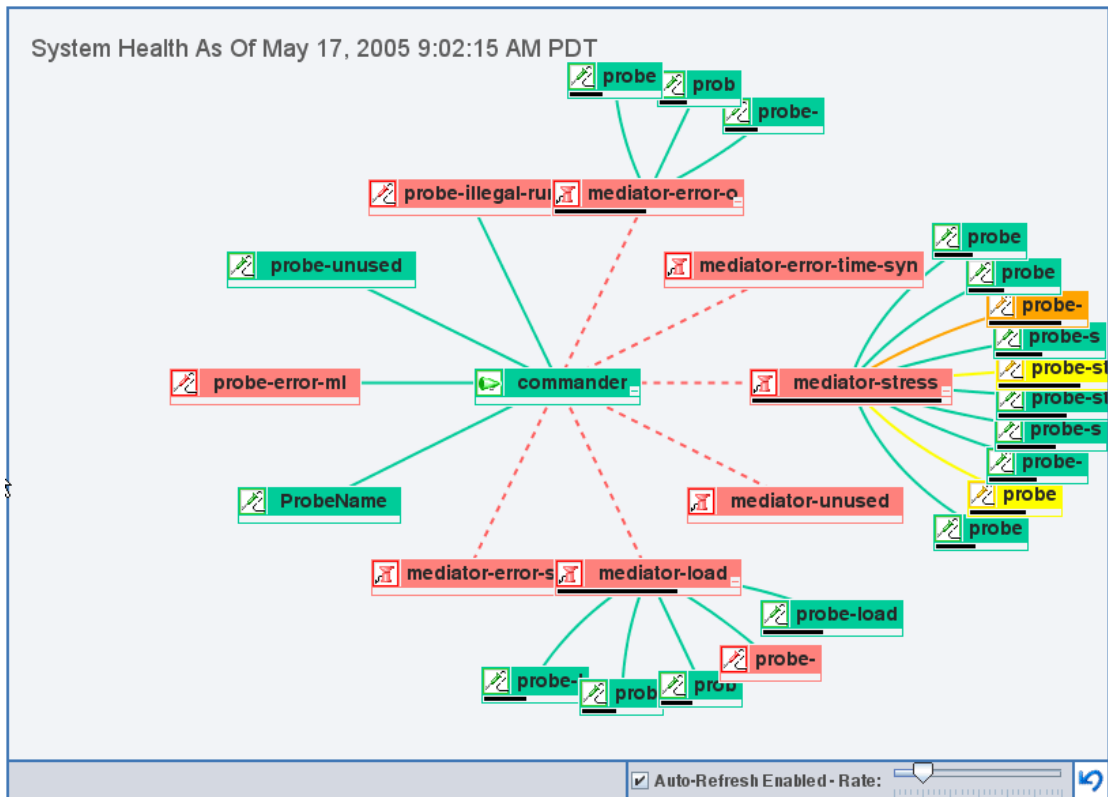
- ▶ Toggle the Graph Legend so that it is either displayed or not. The menu option toggles between **Hide Graph Legend** and **Show Graph Legend**.
- ▶ Toggle the load indicator from the wafer that appears behind the component icon to a bar scale that appears below the component icon. The menu option toggles between **Show Load Using Scale** and **Show Load Using Circles**.

To display the component's load on a bar scale instead of a wafer:

- 1 Right-click the System Health screen at any spot except where a component icon is displayed. The following menu is displayed:



- 2 Choose **Show Load Using Scale**. The wafers are replaced by load bars underneath the components. The length of the bar indicates the amount of load on the component.



The length of the bar indicates the amount of load on the component.



## Creating and Using System Health Monitor Snapshots

### Exporting a Snapshot of the System Health Monitor

You may 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 the XML format or can be imported into any working copy of the System Health Monitor.

---

**Note:** Remember that people who have access to the Commander 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 If the System Health Monitor is not already displayed in a browser window, enter the following URL in your Web browser:

`http://<commander host>:<commander port>/registrar/xml`

where <commander host> is the host of the Commander from which the System Health snapshot is to be saved and <commander port> is the port that the Commander is using to communicate. The default port is 2006.

- 2 Using the Save menu option in your Web browser, save the Web page to a file. Be sure to use a name that will help you remember why you saved the snapshot. For example, to remember that you took a snapshot a System Health Monitor with a poor performing mediator 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 Commander host machine.
- 2** Enter the following URL in your Web browser:

`http://<commander host>:<commander port>/registrar/health?xml=<XML_file>`

where <XML\_file> is the path to the XML file on the Commander host machine.

For example, if you copied `sys_health_mediator_yyyymmdd.xml` to the C: drive on the Commander host and Commander host machine name is `jake`, the URL might look like this:

`http://jake:2006/registrar/health?xml=c:\sys_health_mediator_yyyymmdd.xml`

The <commander port> by default should be 2006.

When the System Health Monitor is displayed in your Web browser, the information will be from the imported XML file.

## Troubleshooting Using the System Health Monitor

Troubleshooting tips can be accessed directly from the System Health Monitor as described in “Viewing Troubleshooting Tips” on page 334. The information in these tips will help you to interpret and respond to the information that is displayed on the System Health Monitor graph.

The following situations are some odd behaviors that are not explained in the troubleshooting tips that you should be aware of.

### Interpreting Odd Situations Displayed on the System Health Monitor

#### Mediator

- Do not assign the same port number for the Mediator host machine to listen for the Probe and to listen for the web server. When this is done, the Mediator will fail to start the web server. Despite this failure, the Mediator will be displayed on the System Health Monitor as healthy.



# B

---

## Advanced Diagnostics Commander Configuration

The following configuration instructions are intended for experienced users with in-depth knowledge of this product. Please use caution when modifying any properties for the Diagnostics components.

### Adjusting the Heap Size for the Commander's VM

The size of the heap can impact the performance of the Commander. You should adjust the size of the heap based upon the number of probes that will be sending data through the Commander. The default heap size is 384MB which is adequate for up to 20 probes. The following table lists the recommended heap settings for various numbers of probes:

Number of Probes	Recommended Heap Size (MB)
0 to 20 Probes	384 (default setting)
21 to 50 Probes	768
51 to 100 Probes	1400

The heap size is set in the <comander\_install\_dir>\dat\nanny mediator.nanny file using the VM arguments:

```
-Xms384m -Xmx384m
```

**To adjust the Commander heap size:**

- 1 Open the commander.nanny file that is to be edited. This file is located at:  
`<commander_install_dir>\launch_service\dat\nanny\commander.nanny`
- 2 Replace the heap size specified in the `-Xmx???m -Xms???ms` option on the `start_<os>` property that is appropriate for your system OS with the recommend Heap Size.

```
start_nt="ProductDir\jre\bin\javaw.exe" -Xmx???m -Xms???m ...
```

---

**Note:** Both the `-Xmx???m` and the `-Xms???m` options must have the same value for “???”.

---

For example if you were updating the heap size from the default settings to the recommend heap size for 21 - 50 probes; before you modify this line in the commander.nanny file it will look like this:

```
start_nt="ProductDir\jre\bin\javaw.exe" -Xms384m -Xmx384m "-  
Dserver.dir=ProductDir" -Dsun.net.client.defaultReadTimeout=30000 -  
Dsun.net.client.defaultConnectTimeout=30000 -Dcom.sun.management.jmxre-  
mote -Xbootclasspath/a:"ProductDir\lib\loading.jar;ProductDir\etc" com.mer-  
cury.opal.common.loader.ModuleLoader
```

After you modify this line in the commander.nanny file it will look like this:

```
start_nt="ProductDir\jre\bin\javaw.exe" -Xms768m -Xmx768m "-  
Dserver.dir=ProductDir" -Dsun.net.client.defaultReadTimeout=30000 -  
Dsun.net.client.defaultConnectTimeout=30000 -Dcom.sun.management.jmxre-  
mote -Xbootclasspath/a:"ProductDir\lib\loading.jar;ProductDir\etc" com.mer-  
cury.opal.common.loader.ModuleLoader
```

## Configuring the Commander for Multi-Homed Environments

The machines that host the Commander can be configured with more than one Network Interface Card (NIC), and the Commander process listens on all interfaces on its host machine. Some customer environments do not allow applications to listen on all network interfaces on a machine. If this is the case in your environment, follow these instructions to configure the Commander to listen on specific network interfaces.

### Modifying jetty.xml

The **jetty.xml** file has a section which defines the interfaces on which the Commander is permitted to listen. By default, the **jetty.xml** file included with the Commander has no listeners defined and the Commander listens on all of the interfaces.

**To configure the Commander to listen on specific network interfaces on a machine:**

- 1 Open the `<commander_install_dir>/etc/jetty.xml` file 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>
```

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 Commander 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 Commander can resolve the host names of the Commander 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 Commander host.

---



## Sample jetty.xml File

The following example shows the **jetty.xml** file for the Commander where the Commander 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>

```

```
</New>  
  </Arg>  
</Call>  
.....  
</Configure>
```

# C

---

## Advanced Diagnostics Mediator Configuration

The following configuration instructions are intended for experienced users with in-depth knowledge of this product. Please use caution when modifying any of the Diagnostics component's properties.

This appendix contains the following sections:

- Configuring the Mediator for Large Installations
- Tuning the Mediator Garbage Collection
- Tuning the Mediator for a Smaller Installation
- Overriding the Default Mediator Host Machine Name
- Configuring the Mediator for Multi-Homed Environments
- Reducing Mediator Memory Usage
- LoadRunner / Performance Center Diagnostics Mediator Assignments
- Using the Mediator Configuration Web Pages
- Configuring the Mediator for the LoadRunner Offline file

## Configuring the Mediator for Large Installations

If you will be using a Mediator with more than twenty probes, it is recommended that you make modifications to the default configuration of the Mediator. The following sections provide instructions for making these Mediator configuration modifications.

### Adjusting the Probe Event Buffer Size

Each Mediator has a probe buffer that is used to reduce the effect that temporary processing slowdowns on the Mediator have on the probe's processing. Without this buffer, the probes would react to the temporary processing slowdowns on the Mediator by throttling more frequently. By default, the Mediator's probe buffer size is set to be 2MB per probe. The per-probe buffer size for a Mediator is determined based upon the value of the static Mediator property, `probe.event.buffer.size` which is located in the `<mediator_install_dir>\etc\mediator.properties` file.

To determine the amount of memory that will be allocated for a Mediator's probe buffer, multiply the number of probes that are assigned to the Mediator by the value of the buffer size property.

Total Probe Buffer Size for a Mediator = # of probes assigned to the mediator \* `probe.event.buffer.size`

To determine a reasonable probe buffer size:

- 1 Determine the probe's data rate (bytes/sec) from the Mediator metrics.
- 2 Multiply the probe data rate by the number of seconds of buffering that you need.
- 3 Set the `probe.event.buffer.size` property, located in `<mediator_install_dir>\etc\mediator.properties`, to the probe buffer size that you calculated.

---

**Note:** This buffer size does not guarantee that the probe will not be throttled for this period of time since the VM may be blocked and unable to buffer data when it becomes immediately available. However, the probe buffers should be able to handle most of the minor delays.

---

### **Adjusting the Direct Buffer Allocation Limit**

The Mediator attempts to allocate the probe buffers outside of the java heap, using “direct” ByteBuffers. Ideally, there will be enough direct allocation space available so that all of the probe buffers will be direct. However, each VM has a limit to the amount of direct ByteBuffers that can be allocated.

If the Mediator is unable to allocate a direct buffer for the probe, it will fall back on a heap-based “indirect” buffer.

The default limit for direct buffer allocations for the Sun 1.4.2 VM, which is currently used by the Mediator, is 64MB. You should increase the direct buffer allocations limit when the Total Probe Buffer Size will exceed 64MB.

#### **To increase the direct buffer allocation limit:**

- 1** Open the mediator.nanny file to be edited. This file is located at:

```
<mediator_install_dir>\dat\nanny\mediator.nanny
```

- 2** Add the following argument to the VM arguments on the start\_nt line:

```
-XX:MaxDirectMemorySize=???M
```

Replace “???” with the number of megabytes that you want to allocate for direct memory.

Before you modify this line in the mediator.nanny file it will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Media-
tor\jre\bin\javaw.exe^-server -Xmx128m -Xms128m -XX:+UseAdaptiveSizePol-
icy -XX:+UseParallelGC -Dsun.net.client.defaultReadTimeout=30000 -
Dsun.net.client.defaultConnectTimeout=30000 -Xbootclasspath/a:"C:\Program
Files\Mercury Interactive\Diagnostics\Mediator\lib\loading.jar;C:\Program
Files\Mercury Interactive\Diagnostics\Mediator\etc" com.mercury.diagnos-
tics.common.loader.ModuleLoader
```

After you modify this line in the mediator.nanny file it will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Media-
tor\jre\bin\javaw.exe^-server -Xmx128m -Xms128m -XX:+UseAdaptiveSizePol-
icy -XX:+UseParallelGC -XX:MaxDirectMemorySize=128M -
Dsun.net.client.defaultReadTimeout=30000 -Dsun.net.client.defaultConnect-
Timeout=30000 -Xbootclasspath/a:"C:\Program Files\Mercury Interactive\Diag-
nostics\Mediator\lib\loading.jar;C:\Program Files\Mercury
Interactive\Diagnostics\Mediator\etc" com.mercury.diagnostics.com-
mon.loader.ModuleLoader
```

---

**Note:** 32-bit VMs have a 4GB address space limit, so setting the MaxDirectMemorySize value too high may cause the heap to be too small. In this case, you can either use a 64-bit VM, if one exists for your platform, or decrease the probe buffer size.

---

## Adjusting the Heap Size

The size of the heap can impact the performance of the Mediator. If the heap is too small you may experience problems with the Mediator “hanging” for periods of time. If the heap is too large it is possible that the Mediator will experience long garbage collection delays.

The default value for the heap size is 256MB. The heap size is set in the <mediator\_install\_dir>\launch\_service\dat\nanny mediator.nanny file using the VM arguments:

`-Xmx256m -Xms256m -XX:+UseAdaptiveSizePolicy.`

If you encounter problems with the mediator "hanging", you can increase the heap size specified in the `-Xmx256m -Xms256m` options.

**To adjust the Mediators heap size:**

- 1 Calculate the amount of heap that the Mediator will need based upon the configuration of the machine that is the Mediator host.

The heap size must be large enough to provide the room that the Mediator needs to do its processing plus the amount of space that the indirect probe buffers need.

- The amount of heap that the Mediator needs is about 384M.
- The amount of space that the indirect probe buffers need can be calculated using the values calculated in the previous section as follows:

$$\text{Indirect Probe Buffer Space} = \text{Total Probe Buffer Space} - \text{MaxDirectMemorySize}$$

For example, if the Total Probe Buffer Space needed was 128M and the MaxDirectMemory Size was at the default of 64MB then the Indirect Probe Buffer Space would be 64MB.

- The optimal heap size can be calculated as:

$$\text{Optimal Heap Size} = \text{Amount of Heap the Mediator Needs} + \text{Indirect Probe Buffer Space}$$

For example, continuing the previous example, Optimal Heap Size =  
 $384\text{M} + 64\text{M} = 428\text{M}$

- 2 Open the mediator.nanny file that is to be edited. This file is located at:  
`<mediator_install_dir>\dat\nanny\mediator.nanny`

- 3 Replace the heap size specified in the `-Xmx???m -Xms???ms` option on the `start_nt` line with the Optimal Heap Size that you calculated:

**`-Xmx???m -Xms???m -XX:+UseAdaptiveSizePolicy -XX:+UseParallelGC`**

Continuing the previous example; the current heap size, represented by "???" is replaced with 428 Mb.

**`-Xmx428m -Xms428m -XX:+UseAdaptiveSizePolicy -XX:+UseParallelGC`**

---

**Note:** Both the `-Xmx???m` and the `-Xms???m` options must have the same value for "???".

---

Before you modify this line in the `mediator.nanny` file it will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Mediator\jre\bin\javaw.exe^-server -Xmx256m -Xms256m -XX:+UseAdaptiveSizePolicy -XX:+UseParallelGC -Dsun.net.client.defaultReadTimeout=30000 -Dsun.net.client.defaultConnectTimeout=30000 -Xbootclasspath/a:"C:\Program Files\Mercury Interactive\Diagnostics\Mediator\lib\loading.jar;C:\Program Files\Mercury Interactive\Diagnostics\Mediator\etc" com.mercury.diagnostics.common.loader.ModuleLoader
```

After you modify this line in the `mediator.nanny` file it will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Mediator\jre\bin\javaw.exe^-server -Xmx428m -Xms428m -XX:+UseAdaptiveSizePolicy -XX:+UseParallelGC -Dsun.net.client.defaultReadTimeout=30000 -Dsun.net.client.defaultConnectTimeout=30000 -Xbootclasspath/a:"C:\Program Files\Mercury Interactive\Diagnostics\Mediator\lib\loading.jar;C:\Program Files\Mercury Interactive\Diagnostics\Mediator\etc" com.mercury.diagnostics.common.loader.ModuleLoader
```



## Tuning the Mediator Garbage Collection

Under rare circumstances, the default Garbage Collector, ParallelGC, may cause the Mediator to have long periods where it is non-responsive or does not process Probe data. When this happens, it may trigger the Probe to throttle the capture or to drop events. This problem most commonly occurs when you are running LoadRunner / Performance Center with high Virtual User loads.

There are several symptoms that will help you to identify this problem. The most common situation occurs when the Mediator host is a parallel-processor machine (either SMP or SMT). The investigation sequence should be as follows:

- ▶ Notice that the Probe is throttling or dropping events.
- ▶ Check the health for the Commander and the Mediator on the System Health Monitor and find that they are healthy.
- ▶ Check the CPU utilization for the Mediator host and notice that one of the processors is near 100% while the other processors are almost at 0%.

If you encounter problems with the Garbage Collector interfering with the performance of the Mediator you can adjust the VM arguments in the mediator.nanny file that is located in the directory at **<mediator\_install\_dir>\dat\nanny\mediator.nanny**.

### To Tune the Mediator Garbage Collection:

- 1** Adjust the size of the Java Heap by replacing by replacing the heap size specified in the `-Xmx???m -Xms???m` option as described in “Adjusting the Heap Size” on page 350 to make sure that you have an adequate heap size.
- 2** Replace the `-XX:+UseParallelGC` option added in the previous step with `-XX:+UseConcMarkSweepGC`. This option will decrease the throughput (events/sec) of the Mediator and will result in a constant throughput.

After you have replaced the Aggressive Heap option in the mediator.nanny file as instructed in the previous step, the mediator.nanny file will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Media-
tor\jre\bin\javaw.exe^-server -Xmx428m -Xms428m -XX:+UseAdaptive-
SizePolicy -XX:+UseParallelGC -XX:MaxDirectMemorySize=128M -
Dsun.net.client.defaultReadTimeout=30000 -Dsun.net.client.defaultConnect-
Timeout=30000 -Xbootclasspath/a:"C:\Program Files\Mercury Interac-
tive\Diagnostics\Mediator\lib\loading.jar;C:\Program Files\Mercury
Interactive\Diagnostics\Mediator\etc" com.mercury.diagnostics.com-
mon.loader.ModuleLoader
```

After you replace the UseParallelGC option the mediator.nanny file it will look like this:

```
start_nt=C:\Program Files\Mercury Interactive\Diagnostics\Media-
tor\jre\bin\javaw.exe^-server -Xmx428m -Xms428m -XX:+UseAdaptive-
SizePolicy -XX:+UseParallelGC -XX:+UseConcMarkSweepGC -
XX:MaxDirectMemorySize=128M -Dsun.net.client.defaultReadTime-
out=30000 -Dsun.net.client.defaultConnectTimeout=30000 -Xbootclass-
path/a:"C:\Program Files\Mercury
Interactive\Diagnostics\Mediator\lib\loading.jar;C:\Program Files\Mercury
Interactive\Diagnostics\Mediator\etc" com.mercury.diagnostics.com-
mon.loader.ModuleLoader
```

## Tuning the Mediator for a Smaller Installation

The default Mediator configuration that is established when the Mediator is installed assumes that the Mediator will be the only Mercury Diagnostics Component running on the host machine and so will allocate roughly all of the available memory on the machine to the Java heap. The Java argument `-Xmx???m -Xms???m -XX:+UseAdaptiveSizePolicy` are the options that trigger this memory allocation for the heap.

It is possible to run the Mediator on a machine that is also host to other components for implementations with very light loads such as a for a POC. If you are running the Mediator on the same machine as any of the other Mercury Diagnostics components, you should adjust the size of the Java Heap by replacing the `-Xmx???m -Xms???m -XX:+UseAdaptiveSizePolicy` option as described in “Adjusting the Heap Size” on page 350. The difference is that in the earlier instructions, the goal was to allocate a large enough heap for optimal performance with a large installation. In this section, you want to allocate a small enough heap to allow the other components enough memory to function.

- ▶ Adjust the size of the Java Heap by replacing the heap size specified in the `-Xmx???m -Xms???m -XX:+UseAdaptiveSizePolicy` option making sure to make the `-Xmx???m` parameter small enough so that the other components on the machine have enough memory to function.
- ▶ Since this configuration is for an implementation that will have a very light load, we recommend that you also decrease the probe buffer size to 512k (524288).

## Overriding the Default Mediator Host Machine Name

In situations where a firewall or NAT is in place or where your Mediator host machine has been configured as a multi-homed device, it may not be possible for the Commander to communicate with the Mediator using the host name that was assigned when the Mediator was installed. The **registered\_hostname** property allows you to override the default host machine name that the Mediator uses to register itself with the Commander.

To override the default host machine name for a Mediator set the `registered_hostname` property located in `<mediator_install_dir>/etc/mediator.properties` to an alternate machine name or IP Address that will let the Commander communicate with the Mediator.

This is the host name that is passed to the Probes that are involved in a LoadRunner / Performance Center scenario that allows them to communicate with the Mediator.

## Configuring the Mediator for Multi-Homed Environments

The machines that host the Mediator can be configured with more than one Network Interface Card (NIC), and the Mediator process listens on all interfaces on its host machine. Some customer environments do not allow applications to listen on all network interfaces on a machine. If this is the case in your environment, follow these instructions to configure the Mediator to listen on specific network interfaces.

### Setting the Event Hostname

You should set the **event.hostname** property if the Mediator host machine has multiple network interfaces and you want to specify the hostname that the Mediator will listen on.

This property can be found at:

```
<mediator_install_dir>/etc/mediator.properties
```

Uncomment the property, **event.hostname** and specify the hostname value.

By default, the **event.hostname** property is not set. This means that the Mediator will listen on all hostnames.

### Modifying jetty.xml

The **jetty.xml** file has a section which defines the interfaces on which the Mediator is permitted to listen. By default, the **jetty.xml** file included with the Mediator has no listeners defined and the Mediator listens on all of the interfaces.

**To configure the Mediator to listen on specific network interfaces on a machine:**

- 1 Open the `<mediator_install_dir>/etc/jetty.xml` file 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 Mediator 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 Mediator can resolve the host names of the Mediator 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 Mediator host. See “Overriding the Default Mediator Host Machine Name” on page 356 for more information.

---

## Sample jetty.xml File

The following example shows the jetty.xml file for the Mediator where the Mediator 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 Mediator Memory Usage

The Transaction Timeout Period is a safety mechanism that is used to prevent the mediator from using excessive amounts of memory because it is holding on to old data for too long. The mediator holds on to all of the information that it receives for a transaction until it receives the ELT for the transaction, which tells the mediator the transaction is complete. The timeout period for a transaction is reset each time that the mediator receives data for the transaction. If the machine that the commander 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 and the commander, the mediator may not receive the ELT that lets it know when a transaction has ended. If the ELT is not received by the time that the transaction timeout period expires, the mediator will assume that the ELT is not coming and go ahead and process the data for the transaction and free up the memory that the transaction data was using.

The **correlation.txn.timeout** property sets the duration of the transaction timeout period. If you are experiencing out of memory conditions in the mediator you may want to reduce the transaction timeout period so that the mediator will give up waiting for the end of a transaction sooner. Use caution when adjusting the value of this property because multiple probes may be sending data to the mediator, and a active transaction may be idle in one mediator, so setting this too low could cause problems. It is recommended that if you need to reduce the value of this property that you set it to 30s more than the longest transaction in your load test.



## LoadRunner / Performance Center Diagnostics Mediator Assignments

### Default Mediator Assignment

When a LoadRunner / Performance Center run is started, the Commander selects the Mediators to be used by the Probes during the run. Each Probe receives one Mediator assignment. The Commander assigns a Mediator to a Probe based on the Logical LAN ID and a least-used basis. For each Probe, the Commander determines which Mediators are active and have the same Logical LAN ID. The Commander then selects the Mediator that has been assigned to the fewest probes.

This least-used Mediator is started and its information is passed to the Probe. If the Mediator fails to start, the Commander checks the next least-used Mediator. The Probe receives the assigned Mediator information and connects to the Mediator for the run.

### Advanced Mediator Assignment

There are two different methods for changing the default Mediator assignments for Probes.

### Direct Assignment

When you use the Direct Assignment method, you create a mapping file that allows you to override the Mediator assignment for a Probe. This is useful when you have a Probe that produces much more data than the other probes in a run and you want to make sure that a Mediator that has been installed on a more powerful machine processes the events from that Probe.

To change the Mediator Assignment for a Probe using the Direct Assignment method create a file called `mediator_assignment.properties` in the `<commander_install_dir>\etc` directory on the Commander host machine.

The format of the `mediator_assignment.properties` file is:

```
<id> = <mediator.id>
```

- ▶ Replace `<id>` with the ID of the probe.
- ▶ Replace `<mediator.id>` with the ID of the mediator.

The mediator\_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 Commander.

### **Multiple Assignment**

The Multiple Assignment method should only be used in extreme cases where a single Mediator is not able to handle the output from a Probe. The capabilities of the Mediator make it extremely unlikely that this situation will occur.

#### **To change the default Mediator assignment for a probe using the Multiple Assignment method:**

Change the value of the multiple\_mediator\_per\_probe property in the <commander\_install\_dir>\etc\webserver.properties file to “true”. When this property is set to “true” the Commander is instructed to assign all active Mediators with the same Logical LAN ID to the Probe. The Probe will connect to each of the Mediators and send data distributed evenly to those Mediators.

---

**Note:** Changes to this property value will not take effect until you restart the Commander.

---

## **Using the Mediator Configuration Web Pages**

The Mediator Configuration Web pages provide a way for you to set the property values that control how the Mediator communicates with the other diagnostic 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 Web pages to update the Mediator properties instead of editing the property files directly.

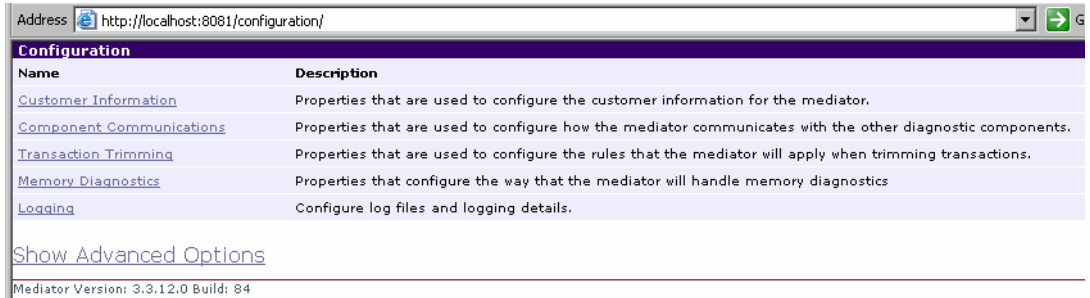
---

**To access the Mediator configuration Web pages:**

Access the Mediator Configuration Web pages on the Mediator host machine using the following URL:

[http://<mediator\\_host\\_name>:8081/configuration/](http://<mediator_host_name>:8081/configuration/)

The Mediator Configuration Main Menu is displayed as shown below.



Each menu option is a link to a page where you may update a group of related properties:

- Customer Information
- Component Communications
- Transaction Trimming
- Memory Diagnostics
- Logger

## Viewing Advanced Mediator Configuration Options

At the bottom of most of the Mediator Configuration pages is a link that will cause additional configuration options to be displayed on the page. These options are Advance Options that you should only update with guidance from your Mercury Customer Support representative.

---

**Note:** Do not manipulate the Advance Options without the guidance of your Mercury Customer Support representative.

---

### To display the Advanced Mediator Configuration Options:

- 1 Click the **Show Advanced Options** link at the bottom of the page.
- 2 The list of options on the page will be updated to include the Advanced Configuration options and the link with toggle to **Hide Advanced Options**.

### To hide the Advanced Mediator Configuration Options:

- 1 Click the **Hide Advanced Options** link at the bottom of the page.
- 2 The list of options on the page will be updated so that the Advanced Configuration options are no longer available and the link will toggle to **Show Advanced Options**.

## Modifying Mediator Properties

### To modify the Mediator properties:

- 1 Select menu item for the properties that you wish to update.
- 2 Review the properties that are displayed and revise any values that should be updated.
- 3 Click **Submit** to save your changes.

---

**Note:** To cancel any pending changes click **Reset All** at any time prior to clicking **Submit**.

---

- 4 A message appears at the top of the page to indicate that your changes were saved.

For the Component Communications and the Transaction Trimming properties, a message reminding you to restart the Mediator is also displayed, along with a convenient **Restart Mediator** button. The property changes will not take effect until you restart the Mediator.

---

**Note:** If you have other changes to make to the Mediator properties, you should finish making all of your changes before restarting the Mediator.

---



---

**Note:** Do not restart the Mediator while a run is in progress.

---

For the Logging properties, it is not necessary to restart the Mediator; however it may take up to a minute for your changes to be applied.

The screenshot shows a web browser window with the address bar containing `http://localhost:8081/configuration/Component+Communications`. The main content area displays a green message: "Changes were successfully saved." Below this, a red circle highlights a button labeled "Restart Mediator" next to the text "You must restart for your configuration changes to take effect." Below the message is a table titled "Component Communications" with the following data:

Name	Value	Description
Commander URL	<input type="text" value="http://metis:2006/rece"/>	The web address for the Diagnostics Commander.
LAN Identifier	<input type="text" value="ThisLan"/>	The logical LAN ID for the mediator.
Probe Data Port	<input type="text" value="2612"/>	The mediator port where it receives event data from the probe.
Mediator Webserver Port	<input type="text" value="8081"/>	The port where the mediator listens for HTTP requests from the Diagnostics Commander.

At the bottom of the table are "Submit" and "Reset All" buttons. Below the table is a link for "Show Advanced Options" and a footer indicating "Mediator Version: 3.3.12.0 Build: 84".

## Configuring the Mediator for the LoadRunner Offline file

For each Load Runner scenario or Performance Center test that is run, the Mediator 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 Mediator 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 Mediator when Load Runner scenario is started.

Locate the offline file on the Mediator host machine in `<mediator_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 Mediator 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 Mediator's offline aggregation periods. 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 Mediator 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 Mediator offline aggregation period properties `bucket.lr.offline.duration` and `bucket.lr.offline.sr.duration` in `<mediator_install_dir>/etc/mediator.properties`.

### To reduce the size of the offline files by increasing the Mediator offline aggregation periods:

- 1** Ensure that the Mediator is not participating in any active LoadRunner / Performance Center runs. This is necessary because the Mediator 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://<mediator\\_hostname>:8081/configuration/Aggregation?level=60](http://<mediator_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 Mediator 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 Mediator. 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 “Modifying Mediator Properties” on page 364.

- 6 To cause the configuration changes to take effect restart the Mediator by clicking **Restart Mediator**.



# D

---

## Advanced J2EE Probe and Application Server Configuration

This appendix provides instructions for configuring the application server and the J2EE Probe using the properties and settings that have been provided for situations that require more advanced configurations.

It contains the following sections:

- Configuring the J2EE Probe for Various Mercury Products
- Setting the J2EE Probe Product Mode Properties
- Configuring the J2EE Probe and Application Server for Deep Diagnostics
- Unconfiguring the Probe for a Product
- Configuring the Application Server for Allocation Capture
- Specifying Layers to Instrument
- Controlling Automatic Method Trimming
- Controlling J2EE Probe Throttling
- Configuring a Probe With a Proxy Server
- Specifying Probe Properties as Java System Properties

## Configuring the J2EE Probe for Various Mercury Products

### Overview

The J2EE Probe is a lifecycle probe that can be used to monitor your applications from development through implementation and production. The same Probe can be used with multiple Mercury products to enable you to understand and improve the performance of your applications. The product mode property is used to indicate the product for which the Probe has been configured.

The Product Mode is set at the time that you install the J2EE Probe when you selected the application that the Probe was going to work with. (See Chapter 8, “Installing the Mercury Diagnostics Probe for J2EE.”) To enable the J2EE Probe to capture data with different Mercury products you must configure the product mode of the Probe.

### AC Product Mode– Mercury Diagnostics Profiler

For Mercury Diagnostics Profiler the Product Mode is **AC**.

---

**Note:** If you would like to use the J2EE Probe with other Mercury Products in addition to the Mercury Diagnostics Profiler, contact Mercury Customer Support.

---

### AM Mode– Application Management

When configured in AM mode, the J2EE Probe will work with Mercury Business Availability Center 5.0 (BAC). A Probe can run in stand-alone AM mode or it can be configured so that it will also be able to gather the additional metrics for Mercury Deep Diagnostics.

### AD Mode– Application Diagnostics

When configured in AD mode, the J2EE Probe will work with LoadRunner 8.1 and Performance Center 8.1. A Probe can run in stand-alone AD mode or it can be configured so that it will also be able to gather the additional metrics for Mercury Deep Diagnostics.

## **DD Mode - Deep Diagnostics**

Mercury Deep Diagnostics for J2EE does not include a probe in its installation; instead, it uses the J2EE Probe installed with Mercury's LoadRunner 8.1, Performance Center 8.1 or BAC 5.0. The J2EE Probe must be configured to work with Mercury Deep Diagnostics for J2EE. A Probe can run in stand-alone DD mode or it can be configured to so that it will also be able to gather additional metrics for the AM mode or the AD mode.

---

**Note:** AM and AD modes are mutually exclusive. The probe can only capture data for one of these modes at any given time. Both AM and AD can be configured to include the Deep Diagnostics mode.

---

## **Setting the J2EE Probe Product Mode Properties**

---

**Note:** If you installed a Probe for use with the Mercury Diagnostics Profiler and would like to use the J2EE Probe with other Mercury Products in addition to the Mercury Diagnostics Profiler, contact Mercury Customer Support. The instructions that follow will not work for a Probe installed for the Mercury Diagnostics Profiler without intervention from Mercury Customer Support.

---

**Note:** The following discussion provides the instructions for updating the Probe properties by editing the property files. It is strongly recommended that you update the properties using the Mercury Configuration Utility. See "Reconfiguring a Probe" on page 158 for information on setting the Product Mode, Commander Host Name, and Commander Port Number using the Configuration Utility.

---

The Product Mode for the J2EE is configured using the **active.products** property which can be found in the property file:

`<probe_install_dir>\etc\probe.properties`

The following is an image of the probe.properties file showing the active.products property. The comment above the property explains the possible combinations of values that can be set. In this example, the **active.products** property is set to “AD,DeepDiagnostics” which means that the Probe is configured to work with LoadRunner / Performance Center 8.1 and Deep Diagnostics.

```
#
# The products that this probe should report to.
#
# Possible values are:
#
#   DeepDiagnostics      (for DD standalone)
#   AM                   (for AM/Topaz)
#   AD                   (for AD/LoadRunner/Performance Center)
#   AM,DeepDiagnostics  (when using DD and AM simultaneously)
#   AD,DeepDiagnostics  (when using DD and AD simultaneously)
#
# (That is, the probe can only report to AM or AD, not both)
#
active.products=AD,DeepDiagnostics
```

For each product mode there are additional properties and settings that must be configured in addition to the **active.products** property. The following sections provide instructions for doing the configuration for each product mode.

## **AM – Application Management Probe Settings**

---

**Note:** Remember, that it is safer to make these changes using the Configuration Utility. See the note on page 371

---

## 1 Configure the Probe to register with the Commander.

One of the functions of the Commander is to keep track of the Diagnostics components so that it can facilitate communications between them and keep you informed about the status and health of the components.

To configure the Probe to register with the Commander, set host name and the port using the **registrar.url** property which can be found in the property file:

```
<probe_install_dir>\etc\dispatcher.properties
```

The following is an excerpt from the dispatcher.properties file showing the **registrar.url** property:

```
## the URL of the registrar
registrar.url=http://host01.company.com:2006/registrar/
```

## 2 Configure the Probe to connect to a Mediator.

The Probe must be able to transmit the processing metrics that it gathers to the Mediator in order for the AD and AM Mercury products to be able to receive, process, and display the reports. In AD, the Probe is assigned to a Mediator by Diagnostics. In AM, you must tell the Probe which Mediator to work with.

To configure the Probe to communicate with the Mediator, set host name and the port using the **mediator.host.name** property and the **mediator.port.number** property which can be found in the property file:

```
<probe_install_dir>\etc\dynamic.properties
```

The default Mediator port is 2612. If you set the Mediator port to a value other than the default when you installed the Mediator, make sure to use that same port number when you set the **mediator.port.number** property.

The following is an excerpt from the dispatcher.properties file showing the 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
```

### 3 Install the Diagnostics Server and Configure the Mediator to be able to communicate with it.

The Diagnostics Server is a Diagnostics component that is required for AM mode. See the Mercury Diagnostics for J2EE & .NET Business Availability Center Edition Installation and User's Guide for more information on installing the Diagnostics Server.

## AD – Application Delivery Probe Settings

---

**Note:** Remember that it is safer to make these changes using the Configuration Utility. See the note on page 371.

---

### 4 Configure the Probe to register with the Commander.

See the instructions above for AM – Application Management Probe Settings to set the **registrar.url** property.

### 5 Configure the probe's Logical LAN ID.

The network traffic between the Mediator and the Probes is high volume. For this reason, the Mediator and the Probes that communicate with it must be located on the same LAN. The Logical LAN ID is not a physical LAN ID. The value that you enter for the Mediator and each of the Probes that you expect to be able to work with the Mediator must be exactly the same.

To configure the Logical LAN ID for the Probe, set the **probe.lanid** property which can be found in the property file:

```
<probe_install_dir>\etc\dispatcher.properties
```

The following is an excerpt from the dispatcher.properties file showing the **probe.lanid** property:

```
### Product Specific Properties ###  
## Aruba  
  
probe.lanid=DefaultLAN
```

## Configuring the J2EE Probe and Application Server for Deep Diagnostics

---

**Note:** If you installed a Probe for use with the Mercury Diagnostics Profiler and would like to use the J2EE Probe with other Mercury Products in addition to the Mercury Diagnostics Profiler, contact Mercury Customer Support. The instructions that follow will not work for a Probe installed for the Mercury Diagnostics Profiler without intervention from Mercury Customer Support.

---

Once you have installed and configured the probe to work in LoadRunner / Performance Center 8.1, you can configure it to work with Deep Diagnostics.

In the following instructions, <probe\_install\_dir> is the directory where the probe is installed.

### To configure the J2EE Probe and Application Server for Deep Diagnostics:

- 1** Ensure that you have installed the Deep Diagnostics (DD) System Under Test agents on the application server machine. If the DD agent is not there, run the Deep Diagnostics setup, and install the DD SUT.

When you install DD you are asked to provide the location of the <probe\_install\_dir>. This information is used to update properties that DD uses to contact the probe and to associate the probe with the application.

- 2** Verify that the Deep Diagnostics Server can see the installed DD agent by looking at the **global** tab, under the **host tree** node in the Deep Diagnostics Console.
- 3** Ensure that you have configured the J2EE Probe and Application Server as described in Chapter 7, “Introducing J2EE Probe Installation and Application Server Configuration.”
- 4** Locate the directory where the J2EE Probe was installed. In the following instruction, the directory where the probe was installed is referenced as <probe\_install\_dir>.

- 5 Set the following properties in <probe\_install\_dir>/etc/capture.properties as indicated:
  - Set the **timeserver.host** property to the host name for the machine where the Deep Diagnostics Server was installed.
  - Set **timer.synchronize.with.server.clock** = false
  - Set **use.native.timestamps** = true
- 6 Set the following properties in <probe\_install\_dir>/etc/probe.properties:
  - Add “DeepDiagnostics” to the value of “**active.products**”. See the instructions provided in the property files for the correct syntax.  
  
To set the Product Mode for both LoadRunner / Performance Center and Deep Diagnostics the property would be as follows:  
  
`active.products=AD/AM,DeepDiagnostics`
  - Set the “**id**” property in probe.properties to identify the J2EE Probe to Deep Diagnostics. This value must exactly match the Application Definition used to specify instrumentation in the Deep Diagnostics.

---

**Note:** If the J2EE Probe is used in Deep Diagnostics mode simultaneously with AD or AM, and a different “id” for the probe is necessary for those modes, you can specify the “**deepdiagnostics.id**” property in probe.properties to indicate which Deep Diagnostics Application Definition this probe is associated with.

---

## Unconfiguring the Probe for a Product

The Product Mode for the J2EE Probe is configured using the **active.products** property which can be found in the property file:

<probe\_install\_dir>\etc\probe.properties

For more information about the **active.products** property see “Setting the J2EE Probe Product Mode Properties” on page 371.



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.

For example if a Probe has been configured for Deep Diagnostics and LoadRunner / Performance Center the **active.products** property would be as follows:

```
active.products=DeepDiagnostics,AD/AM
```

To reconfigure the Probe so that Deep Diagnostics would no longer be included, remove the product from the property value:

```
active.products=AD/AM
```

---

**Note:** You can also update the Product Mode using the Mercury Configuration Utility. For information about this utility, see Chapter 10, “Configuring the Application Server and Probe Using the Configuration Utility.”.

---

## Configuring the Application Server for Allocation Capture

If you are going to use allocation capture, you must scope the capture to reduce the number of events and you also may have to increase the size of the permanent generation (perm gen) in the heap so that the capture will work properly. Capturing everything from allocation tier can cause the VM to crash if the perm gen is not sized properly.

The following example shows the setting the size of the permanent generation to 64m.

```
-XX:MaxPermSize=64m
```

## Specifying Layers to Instrument

The instrumentation for the Probe is specified in the Capture Points file. Instrumentation refers to assigning classes and methods to layers in the Capture Points file. The Probe will gather performance metrics for the layers that are included in the Capture Points file.

Diagnostics has created the default layers for the performance metrics based upon the resources that are used to perform the processing for certain classes and methods. These 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 the layers that have been defined for Diagnostics see “Diagnostics Layers” on page 9.

---

**Note:** The Capture Points file is named `auto_detect.points` and is located in: `<probe_install_dir>\etc\auto_detect.points`

---

You may define custom layers by adding entries to the Capture Points files. You may want to define custom layers for a class that contains J2EE/.NET functionality that does not directly implement a J2EE/.NET component. Or, you may wish to monitor a class that is of special interest to you in a separate layer. To enable Diagnostics for J2EE & .NET to display custom classes or packages, you must include entries for the custom classes in the Capture Points file.

### Modifying the Capture Points file

---

**Note:** These instruction only apply when using the J2EE Probe in the AM or AD product mode. These instructions do not apply to the DD or AC product mode.

---

The Capture Points file is used to configure the instrumentation of your application. The Capture Points file is named `auto_detect.points` and is located in:

**<probe\_install\_dir>\etc\auto\_detect.points**

To specify the layers that are to be displayed in the Capture Points file, enter a key 'layer=...' instrumentation definition for each layer. For example, to specify that the Web.JSP layer should be displayed, you enter the following line:

```
layer = Web.JSP
```

Following is an excerpt from a Capture Points file:

```
[JDBC-SQL]
keyword = jdbcsql

[JSP-_jspService]
;----- implements HttpJspPage Interface -----
class    = javax.servlet.jsp.HttpJspPage
ignore_cl = weblogic.servlet.jsp.JspBase
method   = _jspService
ignore_method = <clinit> ()V
signature = (Ljavax/servlet/http/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)V
deep_mode = soft
layer    = Web.JSP

[Servlet-all]
;----- extends HttpServlet -----
class    = javax.servlet.http.HttpServlet
ignore_cl = javax.servlet.http.HttpServlet, weblogic.servlet.JSPServlet,
com.bea.netuix.servlets.manager.UIServlet, com.bea.netuix.servlets.manager.PortalServlet, com.sap.engine.services.servlets_jsp.server.servlet.InvokerServlet, com.sapportals.portal.prt.dispatcher.Dispatcher
ignore_tree = org.apache.jasper.runtime.HttpJspBase
ignore_method = <clinit> ()V
method      = !.*
signature   = !.*
deep_mode   = hard
layer      = Web.Servlet
```

## Controlling Automatic Method Trimming

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`.

### Latency Trimming

Methods that complete with latency greater than or equal to the value of the “**minimum.method.latency**” property will be captured and those that complete with latency less than this limit will be trimmed to avoid incurring the overhead for methods that are less likely to be of interest.

Because of threading and buffering behavior, partial information about a method that was trimmed may be transmitted to the Mediator. When the Mediator 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.

---

**Note:** The following should be considered 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 upon your platform & whether native timestamps are being used (`use.native.timestamps = false`), it may not be useful to specify this value in increments of less than 10ms.
- ▶ For Am or AD capture, the throttle mechanism will automatically increase this value if the mediator is unable to keep up with the number of events being sent. For Deep Diagnostics, the throttle mechanism will automatically increase this value if the probe is unable to write events out to the workload log file fast enough. For more information on throttling, see “Controlling J2EE Probe Throttling” on page 382.

### Depth Trimming

Methods that are called at a stack depth that is less than or equal to the value of the “**maximum.stack.depth**” property will be captured and 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, and `/login.do` calls `a()` calls `b()` calls `c()`, only `/login.do`, `a`, and `b` will be captured.

---

**Note:** 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 mediator's depth trim (`trimming.type=depth` in `mediator.properties`) which is not enabled by default.

## Controlling J2EE Probe Throttling

The default configuration for the J2EE Probe has been set to minimize the performance impact of collecting diagnostics information. In addition, the J2EE Probe is able to automatically scale back 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 J2EE 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 varies between the configured minimum latency trim value which defaults to 51 milliseconds, and 6 seconds depending on the load.

The probe properties that are used to control the 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

When you are using the J2EE Probe to monitor a production system, allowing the Probe to automatically throttle is usually the preferred behavior. 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 J2EE 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.

---

## Configuring a Probe With a Proxy Server

There are two properties that are used tell the probe the url of the Commander with which the Probe is to work. The property that you set depends upon whether there is a proxy present or not.

- **dispatcher.properties**

The **registrar.url** property in `<probe_install_dir>\etc\dispatcher.properties` is set when you install the Probe. This is the Commander url that is to be used when there is a direct connection.

- **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 Commander.

## Specifying Probe Properties as Java System Properties

All of the J2EE Probe properties, except for those defined in the `dynamic.properties` property file, may be specified as a Java System properties on the startup command-line for the application server. This is very useful when there will be more than one JVM using a single J2EE Probe installation.

To specify a Probe property as a Java System property, pre-pend the letter “D” and the first part of the properties file name to the property name. This is best explained via 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, “**id**”:

```
-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, “**active.products**”:

```
-Dprobe.active.products=AD,DeepDiagnostics
```

- ▶ To set the “**registrar.url**=`http://host01.company.com:2006/registrar/`” 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, “**registrar.url**”:

```
-Ddispatcher.registrar.url=  
http://host01.company.com:2006/registrar
```



# E

---

## Advanced .NET Probe Configuration

This appendix provides instructions for configuring the .NET Probe.

It Contains the following topics:

- Automatically Discovering ASP.NET Applications
- Customizing the Instrumentation for ASP.NET Applications
- Restarting IIS
- Elements Used in the Probe\_config.xml File
- Disabling Logging
- Overriding the Default Probe Host Machine Name

### Automatically Discovering ASP.NET Applications

The .NET Probe installer attempts to detect the ASP.NET applications on the machine where the Probe is installed and configures the Probe to capture basic ASP/ADO/MSMQ workload for each of the ASP.NET applications detected.

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 may have been installed in a manner that prevents them from being detected. One reason that an ASP.NET application will be missed is when an ASP.NET application has been installed as web directories instead of virtual directories.

After the .NET Probe has been installed, you can force the Probe to rescan the IIS configuration if you have fixed ASP.NET application deployments or installed additional ASP.NET applications. To force the Probe to rescan the IIS configuration and update your probe\_config.xml file use the command **Start > Mercury Diagnostics .NET Probe > Rescan ASP.NET Applications**.

For each ASP.NET application that is automatically detected, the installer creates an appdomain reference in the probe\_config.xml file located in the <probe\_install\_dir>/etc directory. Each of the appdomain references in the probe\_config.xml file contains reference to a points file. The referenced points file is the application-specific points file that the installer created for each detected application. The points file controls the workload that the Probe will capture for each application. You can modify the points file to capture application specific custom business logic.

## 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 is to be applied to all ASP.NET processing on the monitored server. You may create custom instrumentation files to capture the business logic that has been implemented through application specific classes.

To let the .NET Probe know that you want to apply the custom instrumentation to an application you must update the probe\_config.xml file so that the customized points file can be associated with the application.

### To associate a customized points file with an application:

- 1 Create a points file with the instrumentation for the application specific classes. To create a points file copy an existing points file in the <probe\_install\_dir>/etc folder.

---

**Note:** If the application was auto detected, a points file will already exist for the application.

---

- 2 Customize the points file so that the Probe will capture custom business logic for your applications.

The following example illustrates how to modify the points file so that the Probe will capture IBuySpy custom code:

```
[IBuySpy Callee]
class = !IBuySpy.*
method = !.*
signature =
scope =
ignoreScope =
layer = Custom.IBuySpy
```

- 3 Add an <appdomain name> tag for the application and its points file to the probe\_config.xml file which is located in the <probe\_install\_dir>/etc directory.

```
<appdomain name="your app name">
  <points file="your app.points"/>
</appdomain>
```

The following example illustrates this step. A custom points file has been created for the MSPetsShop application. The file has been named MSPetShop.points. The <appdomain name> tag for the application and the points file has been added to the probe\_config.xml file.

```
<?xml version="1.0" encoding="utf-8"?>
  <probeconfig>
    <id probeid="" lanid=""/>
    <webserverportrange start="" end=""/>
    <registrar url="http://localhost:2006/registrar"/>
    <instrumentation>
      <logging level="off" threadids="no"/>
    </instrumentation>
    <process name="ASP.NET">
      <logging level="info"/>
      <points file="ASP.NET.points"/>
      <appdomain name="MSPetShop">
        <points file="MSPetShop.points"/>
      </appdomain>
    </process>
  </probeconfig>
```

## Restarting IIS

After you have modified the instrumentation and configuration for the Probe you must restart IIS. To restart IIS from the command line or from the **Start > Run** dialog box, enter **iisreset** and hit return. This command will restart the web publishing service but will not immediately start the Probe.

Your applications will be instrumented and the Probe will connect to the Commander that you specified the next time that the application is run, that is when a page is requested.

You can verify that the Probe is connected by opening a browser and browsing to <http://<CommanderHost>:<CommanderPort>/registrar>. Click on the 'View / Edit Registered Components' link and look for your application listed in the Name column.

---

**Note:** ASP.NET is designed to restart 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 happening, there may be a period of overlap where there are multiple Probes simultaneously registered with the Commander and connected to the Mediator. LoadRunner / Performance Center may report errors during the application restart sequence.

---

## Elements Used in the Probe\_config.xml File

The following table describes the elements that are used in the probe\_config.xml file:

### <id> element

Attributes	Valid Values	Default	Description
probeid	Letters/ digits/ underscore/ dash/ period	\$(AppDomain) .NET	Names the Probe used by LoadRunner / Performance Center and System Health
lanid		DefaultLAN	Defines the logical LAN for this probe (used by Commander and SH)

### <modes> element

Attributes	Valid Values	Default	Description
am	true false	false	Probe in AM mode
ad	true false	true	Probe in AD mode

**<registrar> element**

Attributes	Valid Values	Default	Description
URL	Registrar URL. http:// <host>: <port>/ registrar	none	URL to connect to registrar
delay	number	1000	Number of milliseconds to wait before registering
keepalive	number	15000	Number of milliseconds between keepalives
proxy	URL of proxy	none	Registrar connection proxy.
registered_host name	string	none	Name of host to register as (external name for firewall traversing)

**<mediator> element**

Attributes	Valid Values	Default	Description
host	host name	none	Name of mediator
port	number	2612	Mediator port
block	true/false	false	Block until mediator connection established

**<webserver> element**

Attributes	Valid Values	Default	Description
start	number	35000	Starting port for webserver
end	number	35100	Ending port for webserver

**<bufferpool> element**

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

**<lwmd> element**

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
growth	number	15	Number of collections to growth track
size	number	15	Number of collections to size track
include	string	none	include
exclude	string	none	exclude



**<instrumentation><logging> element**

Attributes	Valid Values	Default	Description
level1	off info severe warning debug	off	Sets the logging level for instrumentation
threadids	true false	false	Include thread ids in log

**<logging> element**

Attributes	Valid Values	Default	Description
level	off info severe warning debug	off	Sets the logging level for capture

**<points> element**

Attributes	Valid Values	Default	Description
file	string	none	Name of instrumentation points file

**<sample> element**

Attributes	Valid Values	Default	Description
method	percent count	percent	Sets the sampling method
rate	number	0	Sets the sampling rate for percent type

**<trim><depth> element**

Attributes	Valid Values	Default	Description
enabled	true false	false	Enables depth trimming
depth	number	10	Sets the depth for depth trimming

**<trim><latency> element**

Attributes	Valid Values	Default	Description
enabled	true false	false	Enables latency trimming
throttle	true false	false	Enables latency trimming throttling
min	number	50	Minimum latency threshold
max	number	100	Maximum latency threshold
increment	number	10	Threshold increment
increment threshold	number	90	
decrement threshold	number	40	

**<appdomain> element**

Attributes	Valid Values	Default	Description
name	string	none	Name of the .NET AppDomain setting apply to

**<appdomain<logging> element****<appdomain><points> element****<appdomain><webserver> element****<appdomain><modes> element****<appdomain><registrar> element****<appdomain><mediator> element****<appdomain><sample> element****<appdomain><trim> element****<process> element**

Attributes	Valid Values	Default	Description
name	string	none	Name of the process settings apply

**<process><appdomain> element**

**<process><logging> element**

**<process><points> element**

**<process><webserver> element**

**<process><modes> element**

**<process><registrar> element**

**<process><mediator> element**

**<process><sample> element**

**<process><trim> element**

**<probeconfig><instrumentation> element**

**<probeconfig><id> element**

**<probeconfig><process> element**

**<probeconfig><logging> element**

**<probeconfig><webserver> element**

**<probeconfig><modes> element**

**<probeconfig><registrar> element**

**<probeconfig><mediator> element**

**<probeconfig><sample> element**

**<probeconfig><trim> element**

## Disabling Logging

You can disable probe application logging by changing the logging 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 tag of the instrumentation section as shown in the following example:

```
<instrumentation>  
  <logging level="off" threadids="no"/>  
</instrumentation>
```

## Overriding the Default Probe Host Machine 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 Commander to communicate with the Probe. The registered\_hostname property allows you to override the default host machine name that the Probe uses to register itself with the Commander.

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 will let the Commander communicate with the Probe (for example, <registrar url="http://foo:2006/registrar" registered\_hostname="bar"/>).



# F

---

## Configuring Diagnostics Components to Work with a Firewall

This appendix describes the configuration steps that you must perform to enable Mercury Diagnostics for J2EE & .NET to work correctly in an environment where a firewall is present. This additional configuration is required when the firewall separates the Probes and Mediator from the Commander and the components of LoadRunner / Performance Center 8.1.

---

**Note:** The following configuration instructions should only be used by experienced users with in-depth knowledge of Diagnostics for J2EE & .NET. Please use caution when modifying any configuration settings for the Diagnostics components.

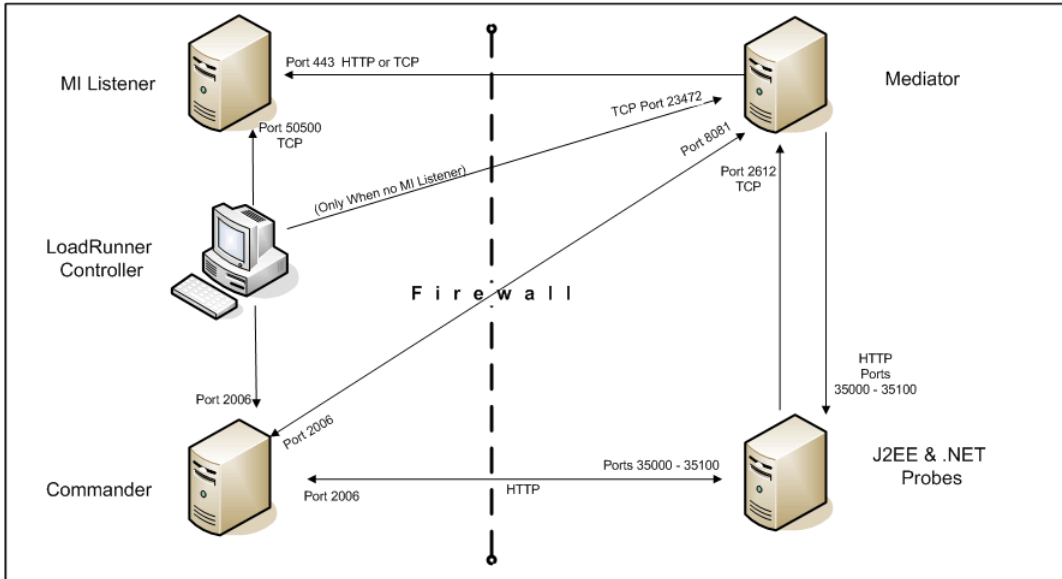
---

The appendix includes the following sections:

- ▶ Overview of Configuring Diagnostics for a Firewall
- ▶ Collating Mediator Offline Files over a Firewall
- ▶ Installing and Configuring the Mercury MI Listener
- ▶ Configuring the Mediator to Work with a Firewall
- ▶ Configuring LoadRunner and Performance Center to Work with Diagnostics Firewalls

## Overview of Configuring Diagnostics for a Firewall

The diagram below shows a typical Diagnostics topology where a firewall separates the Mediator and 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 your firewall to allow the Diagnostics components to communicate with each other.

**To configure your firewall to enable the communications between the Diagnostics components open the ports that will:**

- Allow HTTP requests from the Mediator to the Commander on port 2006.
- Allow HTTP requests from the Probe and the Mediator to the Commander on port 2006.



- Allow TCP requests from the Probe to the Mediator on port 2612.
- Allow HTTP requests from the Commander to the Mediator on port 8081.
- Allow HTTP requests from the Commander 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. See “Configuring the Probes for Multiple Application Server Instances” on page 196 for information on setting the Probe port range.

---

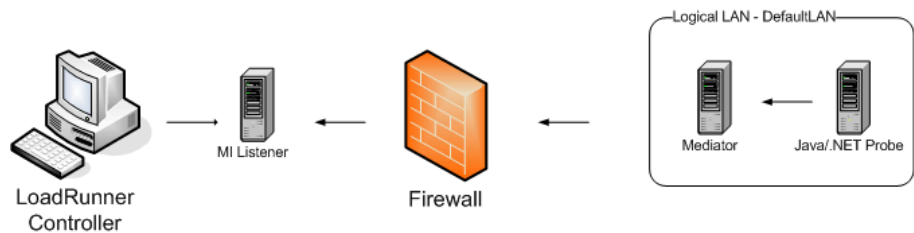
**Note:** In addition to the above topology, if you are using the LoadRunner Analysis Tool to view offline J2EE results, see “Collating Mediator Offline Files over a Firewall” to properly configure the Controller and the Mediators for offline file retrieval.

---

## Collating Mediator Offline Files over a Firewall

During a load test, the Diagnostics Mediator will generate an offline analysis file on the Mediator host machine. LoadRunner / Performance Center retrieves this file when it collates the results of a load test.

If your system has a firewall in between the Controller and the Mediator involved in a load test, you must configure the Controller and Mediator 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 Mediators that are behind a firewall:**

- ▶ Install and configure the Mercury MI Listener.
- ▶ Configure the Mediator 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 Mediator to Work with a Firewall

Once you have installed and configured the Mediator as described in Chapter 5, “Installing the Mercury Diagnostics Mediator,” you must complete additional configuration steps so that the Mediator will be permitted to work across a firewall.

### Configuring the Mediator for a Firewall on a Windows Machine

- 1 Modify the `<mediator_install_dir>\merc_asl\process.asl` file. Add the following lines:

```
nanny.exe=  
nanny=  
javaw.exe=  
javaw=
```

- 2 Modify the `<mediator_install_dir>\launch_service\dat\nanny\mediator.nanny` file.

- ▶ Locate the `start_nt` line:

```
start_nt="d:\Program Files\Mercury Interactive\Diagnostics\Mediator\jre\bin\  
javaw.exe"-server -Xmx256m -Xms256m.....
```

- ▶ Remove the quotes that surround the java.exe path and add the caret (^) after javaw.exe as shown below:

```
start_nt=d:\Program Files\Mercury Interactive\Diagnostics\Mediator\jre\bin\  
javaw.exe^server -Xmx256m -Xms256m .....
```

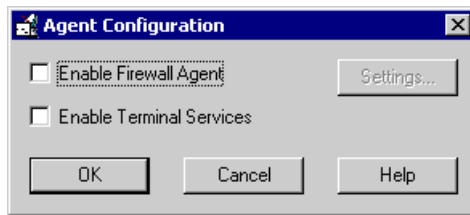
**3** Modify the `<mediator_install_dir>/etc/mediator.properties` file.

- ▶ 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 `<mediator_install_dir>/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 settings dialog box opens.

- 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.

The image shows a dialog box titled "Agent Configuration" with a close button (X) in the top right corner. Below the title bar, it says "Over Firewall Settings". There is a table with two columns: "Property" and "Value". The "MI Listener Name" property is highlighted in blue. Below the table, there is a section for "MI Listener Name" with a text box containing the instruction: "The name, full name or IP address of the redirection server". At the bottom right, there are "OK" and "Cancel" buttons.

Property	Value
MI Listener Name	
Local Machine Key	
Connection Timeout (seconds)	20
MI Listener User Name	
MI Listener Password	
Server Domain	
<input type="checkbox"/> Connection Type	
<input checked="" type="radio"/> TCP	
<input type="radio"/> HTTP	
Proxy Name	
Proxy Port	
Proxy User Name	
Proxy Password	

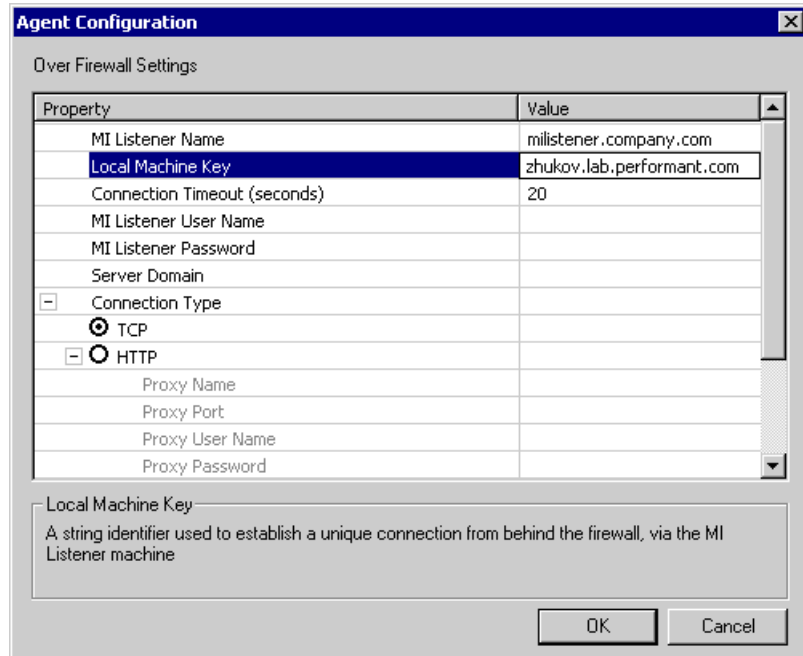
MI Listener Name  
The name, full name or IP address of the redirection server

OK Cancel

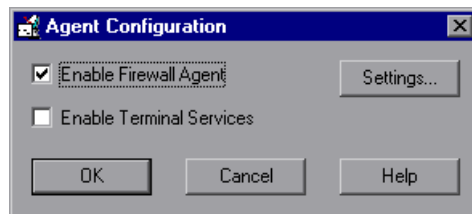
- 8 For the **Local Machine Key** property, enter the machine name of the Mediator host.

Use the System Health Monitor to determine the machine name for the Mediator host. For more information on the System Health Monitor, see Appendix A, “Using the System Health Monitor.”

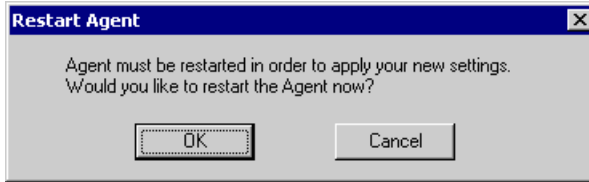
Click **OK** to close the setting dialog box.



- 9 Click **OK** again to close the Agent Configuration dialog box.



- 10 The Restart Agent dialog box opens. Click **OK** to restart the Agent.



- 11 Restart the Mercury Diagnostics Mediator service.

### Configuring the Mediator for a Firewall on a Solaris Machine

See “Installing the Mediator on a UNIX Machine” on page 67 for information on installing the Mediator on Solaris.

- 1 Modify the `<mediator_install_dir>/merc_asl/process.asl` file by adding the following lines:

```
nanny.exe=  
nanny=  
java=
```

- 2 Modify the `<mediator_install_dir>/launch_service/dat/nanny/mediator.nanny` file:

- ▶ Locate the `start_nt` line:

```
start_solv4= "/opt/Diagnostics/Mediator/jre/bin/java" -server -Xmx256m -  
Xms256m.....
```

- ▶ Remove the quotes that surround the java path and add the caret (^) after java as shown below:

```
start_solv4= /opt/Diagnostics/Mediator/jre/bin/java^ -server -Xmx256m -  
Xms256m.....
```

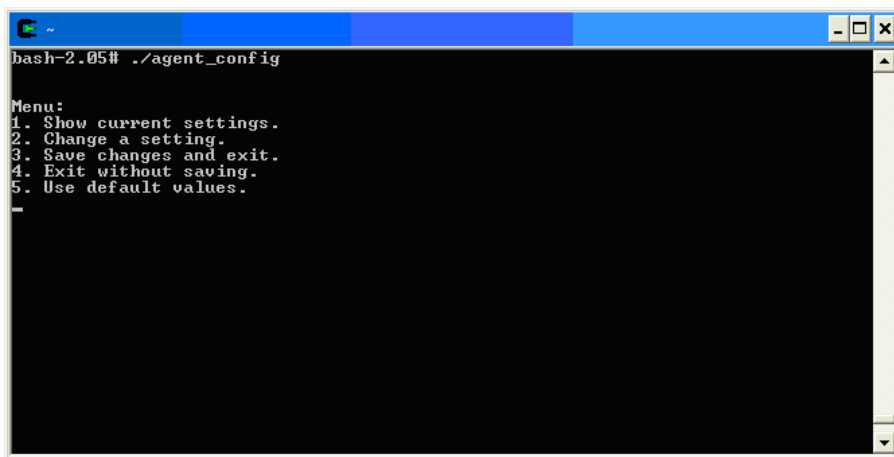
- 3 Modify the `<mediator_install_dir>/dat/br_Inch_server.cfg` file.

Change the value of the `FireWallServiceActive` property to 1.

- 4 Modify the `<mediator_install_dir>/etc/mediator.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:

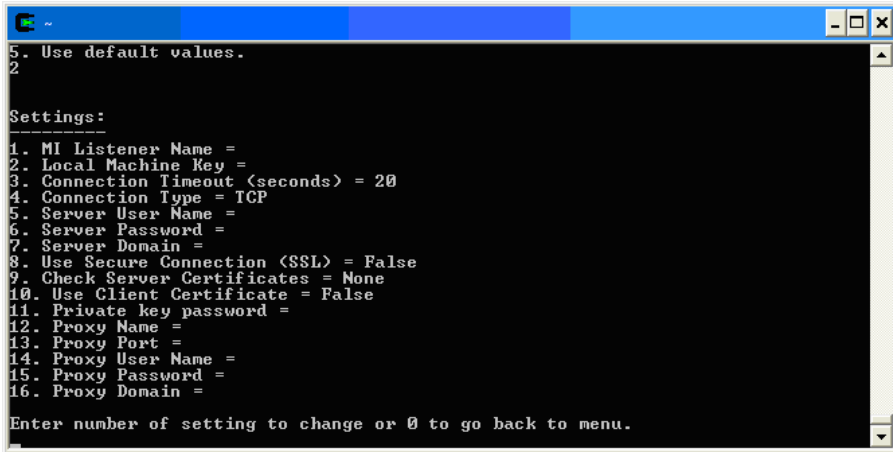
```
export LD_LIBRARY_PATH=.  
export M_LROOT=<mediator_install_dir>  
cd $M_LROOT/bin  
./agent_config
```
- 6 In the Agent Configuration Utility window, press **2**, **Change a Setting**.

A screenshot of a terminal window titled "bash-2.05# ./agent\_config". The terminal displays a menu with five options: 1. Show current settings., 2. Change a setting., 3. Save changes and exit., 4. Exit without saving., and 5. Use default values. The cursor is positioned at the end of the first option.

```
bash-2.05# ./agent_config  
Menu:  
1. Show current settings.  
2. Change a setting.  
3. Save changes and exit.  
4. Exit without saving.  
5. Use default values.  
-
```

**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.



```

5. Use default values.
2

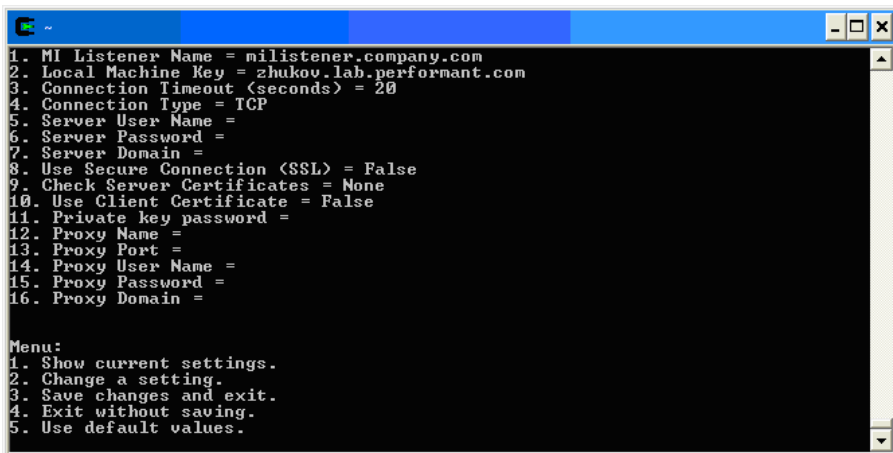
Settings:
1. MI Listener Name =
2. Local Machine Key =
3. Connection Timeout (seconds) = 20
4. Connection Type = TCP
5. Server User Name =
6. Server Password =
7. Server Domain =
8. Use Secure Connection (SSL) = False
9. Check Server Certificates = None
10. Use Client Certificate = False
11. Private 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 **Local Machine Key**, and enter the machine name of the Mediator host as displayed in the **Host:** field on the System Health Monitor.

Use the System Health Monitor to determine the machine name for the Mediator host. For more information on the System Health Monitor, see Appendix A, “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 Mediator.

For more information about restarting the Mediator, see “To start a Mediator on a Solaris machine:” on page 73 .

## **Configuring LoadRunner and Performance Center to Work with Diagnostics Firewalls**

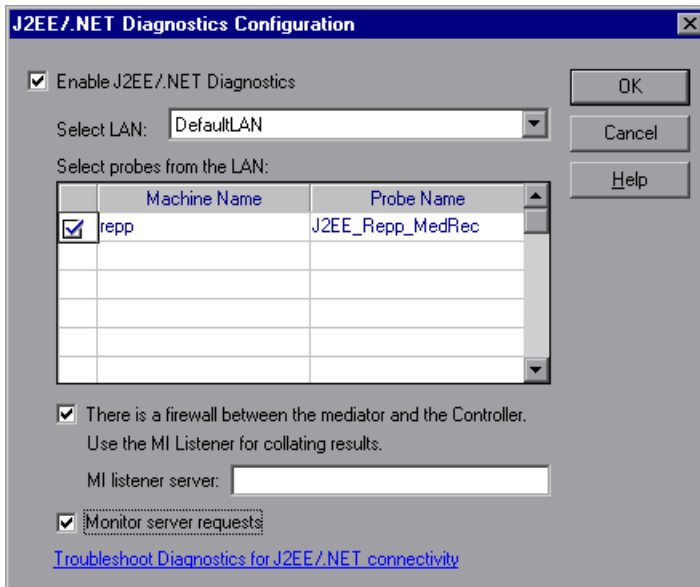
After the MI Listener has been installed and your Mediator machines have been configured, you must update the Diagnostics Configuration in LoadRunner / Performance Center so that the application knows to use the MI Listener when it is transferring the offline data from a Mediator that is outside of a firewall. This section describes:

- Configuring LoadRunner to work with a Diagnostics firewall
- Configuring Performance Center to work with a Diagnostics 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 checked.

- 3 In the Online and Offline Diagnostics section, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.



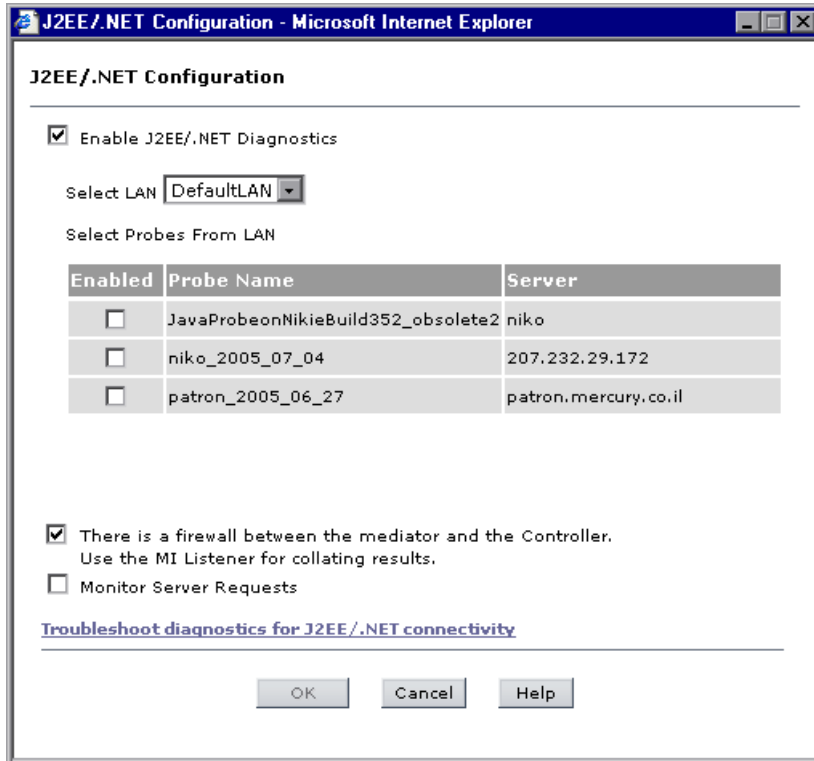
- 4 Select **There is a firewall between the mediator and the Controller**.
- 5 Specify the machine name of the MI Listener host in the **MI listener server** text box.

See “Setting Up Diagnostics for J2EE & .NET on LoadRunner 8.1” on page 207 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 checked.
- 5 In the **Offline and Online Diagnostics** section, click **Configure** to open the J2EE/.NET Diagnostics Configuration dialog box.



- 6 Check **There is a firewall between the mediator and the Controller**.

See “Setting Up Diagnostics for J2EE & .NET on Performance Center 8.1” on page 217 for more information about configuring Performance Center to use the Diagnostics components.



# G

---

## Configuring Diagnostics Components for HTTPS

This appendix describes the configuration steps that you must perform to enable HTTPS communications between the Diagnostics for J2EE & .NET components.

---

**Note:** The following configuration instructions are intended for experienced users with in-depth knowledge of Diagnostics for J2EE & .NET. Please use caution when modifying any configuration settings for the Diagnostics components.

---

The appendix includes the following sections:

- ▶ Configuring the Commander to Receive HTTPS
- ▶ Configuring a Mediator to Communicate via HTTPS with the Commander
- ▶ Configuring a J2EE Probe to Communicate via HTTPS with the Commander
- ▶ Configuring a .NET Probe to Communicate via HTTPS with the Commander
- ▶ Configuring the Mediator to Receive HTTPS
- ▶ Configuring the Commander to Communicate via HTTPS with the Mediator

## Configuring the Commander to Receive HTTPS

---

**Note:** The Diagnostics Commander cannot be accessed by Performance Center via HTTPS

---

**To configure the Commander for incoming HTTPS connections:**

- 1 Generate the keystore in the <commander\_install\_dir>/etc directory using the following command, replacing the strings between the "< >" with the value requested. Remove the "< >" from the actual values you use:

```
<commander_install_dir>/jre/bin/keytool -keystore  
<commander_install_dir>/etc/keystore -storepass <password> -alias  
SERVER -genkey -keyalg RSA -keypass <password> -dname  
"CN=<commander_hostname>, OU=Diagnostics, O=Mercury, L=Mountain  
View, S=CA, C=USA" -validity 3650
```

---

**Note:** The value specified for the <commander\_hostname> must be the machine name for the commander host and not the IP address.

---

- 2 Generate the Commander public.key file using the following command, replacing the strings between the "< >" with the value requested. Remove the "< >" from the actual values you use:

```
<commander_install_dir>/jre/bin/keytool -keystore  
<commander_install_dir>/etc/keystore -storepass <password> -alias SERVER -  
export -rfc -file <commander_install_dir>/etc/public.key
```

---

**Note:** The public.key file must be imported into the host machines of those Diagnostics Components that will be accessing the Commander. The instructions for importing the public.key for each Diagnostics component are provided below

---

- 3 Generate an OBF password string using the following, replacing the strings between the "<>" with the value requested. Remove the "<>" from the actual values you use:

```
<commander_install_dir>/jre/bin/java -cp  
<commander_install_dir>/lib/ThirdPartyLibs.jar org.mortbay.util.Password  
<password>
```

- 4 Enable the Commander's SSL listener in <commander\_install\_dir>/etc/jetty.xml by uncommenting the SSL Listener section.

By default the SSL Listener section in the jetty.xml file is commented out. To uncomment the section, remove the “<!--SSL Listener” line from the

beginning of the section and the “END SSL LISTENER -->” line from the end of the section.

```
<!-- SSL LISTENER

<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
      <Set name="Port">8443</Set>
      <Set name="PoolName">main</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>etc/keystore</Set>
      <Set name="Password">__STORE_PASSWORD__</Set>
      <Set name="KeyPassword">__KEY_PASSWORD__</Set>
      <Set name="NonPersistentUserAgent">MSIE 5</Set>
      <Set name="MinThreads">10</Set>
      <Set name="MaxThreads">100</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
    </New>
  </Arg>
</Call>

END SSL LISTENER -->
```



- 5 Replace the text "`__STORE_PASSWORD__`" and "`__KEY_PASSWORD__`" in `<commander_install_dir>/etc/jetty.xml` with the OBF password string that you generated in step 3.

For example, if the OBF password string that you generated was "OBF:a2cx9yvqwe9zv", the uncommented SSL Listener section of the `jetty.xml` file would look as shown below.

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
      <Set name="Port">8443</Set>
      <Set name="PoolName">main</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>/etc/keystore</Set>
      <Set name="Password">OBF:a2cx9yvqwe9zv</Set>
      <Set name="KeyPassword">OBF:a2cx9yvqwe9zv</Set>
      <Set name="NonPersistentUserAgent">MSIE 5</Set>
      <Set name="MinThreads">10</Set>
      <Set name="MaxThreads">100</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
    </New>
  </Arg>
</Call>
```

- 6 You may need to enable the normal, non-SSL HTTP listener for the Commander if your deployment involves the following situations:
  - ▶ The Commander is being used with LoadRunner. LoadRunner cannot connect to Commander using HTTPS.
  - ▶ A Probe cannot connect to the commander using SSL. If a Probe is running in a VM less than 1.4 or running without JSSE, the Commander will need to have the non-SSL HTTP listener enabled.

**Note:** You are responsible for securing network communications so that access to the non-SSL port is restricted to only the authorized systems.

Enable the Commanders non-SSL listener in `<commander_install_dir>/etc/jetty.xml` by uncommenting the non-SSL Listener section.

By default the non-SSL Listener section in the `jetty.xml` file is commented out.

---

**Note:** If the non-SSL section is missing from the jetty.xml file you should add the following immediately after the SSL section.

---

To uncomment the section delete the line "`<!-- NON-SSL LISTENERS`" from the beginning of the section and the line "`END NON-SSL LISTENER-->`" from the end of the section.

The following shows the non-SSL section of the jetty.xml file with the comment lines still in place.

```
<!-- NON-SSL LISTENER

<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">2006</Set>
      <Set name="MinThreads">10</Set>
      <Set name="MaxThreads">100</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
</Call>

END NON-SSL LISTENER -->
```

The following shows the `jetty.xml` file with the comment lines removed so that the non-SSL listener will be activated.

```
< <Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SocketListener">
      <Set name="Port">2006</Set>
      <Set name="MinThreads">10</Set>
      <Set name="MaxThreads">100</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
      <Set name="ConfidentialPort">8443</Set>
      <Set name="IntegralPort">8443</Set>
    </New>
  </Arg>
</Call>
```

- 7** To verify that you have updated the Commander configuration correctly:
  - ▶ Start the Commander.
  - ▶ Open your Web browser to `http://localhost:2006` and verify that the page successfully loads.
  - ▶ Open your Web browser to `https://localhost:8443` and verify that the page loads.

The other Diagnostics components should now be able to connect to Commander via HTTP or HTTPS.

## Configuring a Mediator to Communicate via HTTPS with the Commander

To configure the Mediator to enable HTTPS communications with the Commander:

- 1 Import the Commander's public.key into the Mediator JRE cacerts keystore. The instructions for generating the Commander's public.key can be found in step 2 in "Configuring the Commander to Receive HTTPS" on page 414.

► Execute the following command on the Mediator host machine.

```
<mediator_install_dir>/jre/bin/keytool -import -file  
<commander_install_dir>/etc/public.key -keystore <mediator  
_install_dir>/jre/lib/security/cacerts -alias SERVER
```

► You will be prompted to enter keystore password. Enter the value: changeit.

► You will be asked if the certificate should be trusted:

"Trust this certificate? [no]:" Enter the value: yes.

- 2 Set the following Mediator properties using the Mediator Configuration Web page located using the following URL:

[http://<mediator\\_host>:8081/configuration/Component+Communications](http://<mediator_host>:8081/configuration/Component+Communications)

---

**Note:** These properties are stored in the property file, <mediator\_install\_dir>/etc/mediator.properties. You should modify the properties using the configuration Web pages in order to make sure that your changes are made correctly.

---

Configuration Page Label	Property Value	Property
Commander URL	https:// <commander_hostname>:8443	commander.url
Diagnostics Server/ Commander Proxy Host	<commander_hostname>	dispatcher.server.delivery.url_host
Diagnostics Server/ Commander Proxy Port	8443	dispatcher.server.delivery.url_port
Diagnostics Server/ Commander Proxy Protocol	https	dispatcher.server.delivery.url_protocol

---

**Important!** Make sure that the <commander\_hostname> specified in the property settings above EXACTLY matches the name specified in the "CN=" value when you created the Commander keystore in step 1 in “Configuring the Commander to Receive HTTPS” on page 414. Also, ensure that both values are machine names and not IP addresses.

If these values do not match or if you used an IP address instead of a machine name, you will see error messages in the Mediator log file:

```
2005-04-14 22:36:30,727: WARNING event_listener    EventListener.register[522]
: error registering with registrar at
RemoteHttpComponent@{https://localhost:8443/registrar}:
com.mercury.diagnostics.common.net.registrar.RegistrarException: error registering:
java.io.IOException: HTTPS hostname wrong: should be <localhost>
```

---

- 3** Start the Mediator. For instructions on starting the Mediator, See “Starting and Stopping Mediators” on page 73.
- 4** Verify that the Mediator was properly configured.

Check that the Mediator icon in the System Health Monitor is colored green. If icon for the Mediator is not displayed or if the color of the icon indicates that there might be trouble, review the log for the Mediator for possible problems.

For instructions on accessing the System Health Monitor see “Accessing the System Health Monitor” on page 322.

## Configuring a J2EE Probe to Communicate via HTTPS with the Commander

To configure the J2EE Probe to enable HTTPS communications with the Commander:

---

**Note:** For a J2EE Probe to be able to communicate using HTTPS, the VM being instrumented with the J2EE Probe must be at least a 1.4 VM or have JSSE if the VM version is less than 1.4.

---

- 1 Import the Commander's public.key into the application servers cacerts keystore. The Instructions for generating the Commanders public.key can be found in step 2 in “Configuring the Commander to Receive HTTPS” on page 414.

- Execute the following command on the J2EE Probe host machine.

```
<applications_JRE_dir>/bin/keytool -import -file  
<commander_install_dir>/etc/public.key -keystore  
<applications_JRE_dir>/lib/security/cacerts -alias SERVER
```

- You will be prompted to enter keystore password. Enter the value “changeit”
- You will be asked if the certificate should be trusted:  
“Trust this certificate? [no]:” Enter the value “yes”.

- 2 Set the J2EE Probe's properties so that the Probe can communicate with the HTTPS port on the Commander.

In the <probe\_install\_dir>/etc/dispatcher.properties file set the property, registrar.url to "https://<commander\_hostname>:8443".

---

**Important!** <commander\_hostname> should EXACTLY match the name specified in the "CN=" value used when creating the Commander keystore in Configure Commander for HTTPS - step 2 above. Also, ensure that both values are machine names and not IP addresses.

If these values do not match or if you used an IP address instead of a machine name, you will see error messages in the Mediator log file:

```
2005-04-14 23:12:09,513 WARN com.mercury.opal.capture.dispatcher [shared
InfrequentEventScheduler] registrar.register:
RemoteHttpComponent@{https://localhost:8443/registrar}com.mercury.diaagnost
ics.common.net.registrar.RegistrarException: error registering:
java.io.IOException: HTTPS hostname wrong: should be <localhost>
```

---

- 3 Start the instrumented application. This will also start the J2EE Probe.
- 4 Verify that the J2EE Probe was properly configured.

Check that the icon for the J2EE Probe in the System Health Monitor is colored green. If it is not displayed or if problems are shown, review the log for the Probe to investigate possible problems.

For instructions on accessing the System Health Monitor see "Accessing the System Health Monitor" on page 322.

## Configuring a .NET Probe to Communicate via HTTPS with the Commander

To configure the .NET Probe to enable HTTPS communications with the Commander:

- 1 Copy the Commander's public key to the host machine for the .NET Probe.

Instructions for generating the Commander's public key are documented in "Configuring the Commander to Receive HTTPS" on page 414. If you followed the instructions as specified, the public key will be located at `<commander_install_dir>/etc/public.key` on the host machine for the Commander.

- 2 Update the .NET Probes configuration to point to the HTTPS URL.

Update the configuration of the .NET Probe by editing `<probe_install_dir>/etc/probe_config.xml`. Change the "url" attribute for the `<registrar>` tag to specify the HTTPS URL for the Commander. For example:

```
<registrar url="https://commander_host:8443/registrar"/>
```

- 3 Begin the process of adding the Certificates Snap-in by starting the Microsoft Management Console.

- ▶ Click **Start > Run**.
- ▶ Enter **mmc** into the Open text box and then click **OK**.

- 4 Select **File > Add/Remove Snap-in...** from the MMS Console.

The Add/Remove Snap-in dialog is displayed.

- 5 Click **Add...** on the **Standalone** tab of the Add/Remove Snap-in dialog.

The Add Standalone Snap-in dialog is displayed.

- 6 Select Certificates from the Available Standalone Snap-ins listbox and then click **Add...**

The Certificate Snap-in dialog is displayed.

- 7 Select **Computer account** and then click **Next**.

The Select Computer dialog is displayed.



- 8** Select **Local Computer: (the computer this console is running on)**, and then click **Finish**.
- 9** Click **Close** on the Add Standalone Snap-in dialog.
- 10** Click **OK** on the Add/Remove Snap-in dialog.
- 11** Begin the process of importing the Commander's startup key that you copied by double clicking on Certificates (Local Computer) in the left pane of the MMC snap-in. This should expand the entry to show the lower level components including Trusted Root Certification Authorities.
- 12** Double click Trusted Root Certification Authorities to expand the entry to show the lower level components including Certificates.
- 13** Right-click Certificates to display its pop-up menu and select **All Tasks > Import...** to start the Certificate Import Wizard.
- 14** Click **Next** to move past the Welcome dialog box of the Certificate Import Wizard.
- 15** Indicate the file to import by clicking **Browse...** to navigate to the Commanders public key file.
- 16** On the Open dialog, select "All Files (\*.\*)" in **Files of type:** drop down.
- 17** Navigate to the directory where the Commander's public.key was copied in step 1 and click **Open**.
- 18** Click **Next** to import the public key file.
- 19** Click **Next** to accept the default Certificate Store location of "Trusted Root Certification Authorities".
- 20** Click **Finish** on Completing the Certificate Import Wizard.
- 21** Click **OK** on the Certificate Import Wizard confirmation dialog.
- 22** Restart IIS.

**To verify that you have successfully configured the .NET Probe for HTTPS communication with the Commander:**

- 1** Activate the .NET Probe by browsing to your applications using your web browser.
- 2** Verify that the .NET Probe node for the reconfigured Probe is shown in System Health.

## Configuring the Mediator to Receive HTTPS

To configure the Mediator to receive HTTPS communications:

- 1 Generate the keystore in the <mediator\_install\_dir>/etc directory using the following command, replacing the strings between the "< >" with the value requested. Remove the "< >" from the actual values you use:

```
<mediator_install_dir>/jre/bin/keytool -keystore  
<mediator_install_dir>/etc/keystore  
-storepass <password> -alias MEDIATOR -genkey -keyalg RSA -keypass  
<password>  
-dname "CN=<mediator_hostname>, OU=Diagnostics, O=Mercury, L=Mountain View, S=CA, C=USA" -validity 3650
```

---

**Note:** The value specified for the <mediator\_hostname> must be the machine name for the mediator host and not the IP address.

---

- 2 Generate the Mediator public.key file using the following command, replacing the strings between the "< >" with the value requested. Remove the "< >" from the actual values you use:

```
<mediator_install_dir>/jre/bin/keytool -keystore  
<mediator_install_dir>/etc/keystore  
-storepass <password> -alias MEDIATOR -export -rfc -file  
<mediator_install_dir>/etc/  
public.key
```

---

**Note:** The public.key file must be imported into the host machines of those Diagnostics Components that will be accessing the Mediator. The instructions for importing the public.key for each Diagnostics component are provided below.

---

- 3 Generate an OBF password string using the following, replacing the strings between the "< >" with the value requested. Remove the "< >" from the actual values you use:

```
<mediator_install_dir>/jre/bin/java -cp <mediator_install_dir>/lib/ThirdPartyL-
ibs.jar org.mortbay.util.Password <password>
```

- 4 Enable the Mediator's SSL listener in <mediator\_install\_dir>/etc/jetty.xml by uncommenting the SSL Listener section.

By default the SSL Listener section in the jetty.xml file is commented out. To uncomment the section, remove the "<!--SSL LISTENER " from the beginning of the section and the "END SSL LISTENER-->" from the end of the section.

```
<!-- SSL LISTENER
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
      <Set name="Port">8445</Set>
      <Set name="PoolName">main</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>/etc/keystore</Set>
      <Set name="Password">__STORE_PASSWORD__</Set>
      <Set name="KeyPassword">__KEY_PASSWORD__</Set>
      <Set name="NonPersistentUserAgent">MSIE 5</Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
    </New>
  </Arg>
</Call>
END SSL LISTENER -->
```

- 5** Replace the text "`__STORE_PASSWORD__`" and "`__KEY_PASSWORD__`" in `<mediator_install_dir>/etc/jetty.xml` with the OBF password string that you generated in step 3.

For example, if the OBF password string that you generated was "OBF:a2cx9yvqwe9zv", the uncommented SSL Listener section of the `jetty.xml` file would look as shown below.

```
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
      <Set name="Port">8445</Set>
      <Set name="PoolName">main</Set>
      <Set name="Keystore"><SystemProperty name="jetty.home" default="."/>/etc/keystore</Set>
      <Set name="Password">OBF:a2cx9yvqwe9zv</Set>
      <Set name="KeyPassword">OBF:a2cx9yvqwe9zv</Set>
      <Set name="NonPersistentUserAgent">MSIE 5</Set>
      <Set name="MinThreads">1</Set>
      <Set name="MaxThreads">5</Set>
      <Set name="MaxIdleTimeMs">30000</Set>
      <Set name="LowResourcePersistTimeMs">5000</Set>
    </New>
  </Arg>
</Call>
```

- 6** To verify that you have updated the Mediator configured correctly:
- ▶ Start the Mediator.
  - ▶ Open your Web browser to `http://localhost:8443` and verify that the page successfully loads.
  - ▶ Verify that the hostname with which the Mediator registers is EXACTLY the same as the mediator hostname specified when creating the keystore in step 1 above.

If the hostname does not match, then the Commander will have problems starting a LoadRunner / Performance Center run.

## Configuring the Commander to Communicate via HTTPS with the Mediator

To configure the Commander to enable HTTPS communications with the Mediator:

- 1 Import the Mediator's public.key into the Commander's JRE cacerts keystore. The Instructions for generating the Mediator's public.key can be found in step 2 in "Configuring the Mediator to Receive HTTPS" on page 426.

► Execute the following command on the Commander's host machine.

```
<commander_install_dir>/jre/bin/keytool -import -file
<mediator_install_dir>/etc/public.key -keystore
<commander_install_dir>/jre/lib/security/cacerts -alias MEDIATOR
```

- You will be prompted to enter keystore password. Enter the value "changeit".
- You will be asked if the certificate should be trusted:  
"Trust this certificate? [no]:" Enter the value "yes".

- 2 Start the Mediator.
- 3 Verify that the Commander was properly configured.

Check that the icon for the Mediator in the System Health Monitor is colored green. If it is not displayed or if problems are shown, review the log for the Mediator to investigate possible problems.

For instructions on accessing the System Health Monitor see "Accessing the System Health Monitor" on page 322.



# H

---

## Configuring Diagnostics Components for HTTP/HTTPS Proxy

This appendix describes the configuration steps that you must perform to enable HTTP/HTTPS proxy communications between the Diagnostics for J2EE & .NET components.

---

**Note:** The following configuration instructions are intended for experienced users with in-depth knowledge of Diagnostics for J2EE & .NET. Please use caution when modifying any configuration settings for the Diagnostics components.

---

The appendix includes the following sections:

- ▶ Configuring the Mediator to Communicate Through an HTTP/HTTPS Proxy
- ▶ Configuring the J2EE Probe to Communicate Through an HTTP/HTTPS Proxy
- ▶ Configuring a .NET Probe to Communicate Through an HTTP/HTTPS Proxy
- ▶ Proxy from Commander To Mediator/Probes in Load Test Environment
- ▶ Proxy Server Configuration for HTTPS

## Configuring the Mediator to Communicate Through an HTTP/HTTPS Proxy

The following section describes how to configure the Mediator to communicate with the Commander through an HTTP/HTTPS proxy.

### To configure the Mediator for HTTP/HTTPS Proxy Communications:

- 1 Set the following Mediator properties in `<mediator_install_dir>/etc/mediator.properties`:
  - ▶ Set "proxy.host" to the hostname of the proxy server.
  - ▶ Set "proxy.port" to the port of the proxy server.
  - ▶ Set "proxy.protocol" to the protocol to use for proxy server (http or https).
- 2 If the proxy server's protocol is HTTPS, import the proxy server's certificate into the Mediator's JRE cacerts keystore.

```
<mediator_install_dir>/jre/bin/keytool -keystore -import -file <ProxyCertificateFile>  
-keystore <mediator_install_dir>/jre/lib/security/cacerts -alias PROXY
```

- ▶ Enter keystore password: changeit
  - ▶ Trust this certificate? [no]: yes
- 3 Restart the Mediator. For instructions see "Starting and Stopping Mediators" on page 73.



## Configuring the J2EE Probe to Communicate Through an HTTP/HTTPS Proxy

The following section describes how to configure the Java Probe to communicate with Commander through an HTTP/HTTPS proxy:

### To configure the J2EE Probe for HTTP/HTTPS Proxy Communications:

- 1 Set the following J2EE Probe properties in `<J2ee_install_dir>/etc/dispatcher.properties`:
  - ▶ Set "proxy.host" to the hostname of the proxy server.
  - ▶ Set "proxy.port" to the port of the proxy server.
  - ▶ Set "proxy.protocol" to the protocol to use for proxy server (http or https).
- 2 If the proxy server's protocol is HTTPS, import the proxy server's certificate into the cacerts keystore of the JRE used by the instrumented application.

```
<ApplicationJRE>/bin/keytool -keystore -import -file <ProxyCertificate-File> -keystore <ApplicationJRE>/jre/lib/security/cacerts -alias PROXY
```

- ▶ Enter keystore password: changeit
  - ▶ Trust this certificate? [no]: yes
- 3 Restart the instrumented application VM.

## Configuring a .NET Probe to Communicate Through an HTTP/HTTPS Proxy

To configure the .NET Probe for HTTP/HTTPS Proxy Communications:

- 1 Set the following .NET Probe property in `<.NET_probe_install_dir>/etc/probe_config.xml` to point to the Commander host machine:

```
<registrar url="http://<commander_host_name>:2006/registrar/"  
proxy="http://proxy:8080" />
```

- 2 Restart the instrumented application process.

## Proxy from Commander To Mediator/Probes in Load Test Environment

To configure the .NET Probe for HTTP/HTTPS Proxy Communications:

- 1 Make sure the Mediator and Probe have been configured with a LAN ID that permits the Commander to establish a proxy to them. For example, if the Commander must communicate with Mediator1 and Probe1 through one proxy and to Mediator2 and Probe2 through another proxy, make sure the Mediator1 and Probe1 are configured with a different LAN ID than Mediator2 and Probe2.
  - ▶ The .NET Probe LAN ID can be set in the `<ProbeHome>/etc/probe_config.xml` with the "lanid" attribute.
  - ▶ The Java Probe LAN ID can be set in `<ProbeHome>/etc/dispatcher.properties` with the "probe.lanid" property.
  - ▶ The Mediator LAN ID can be set in the `<MediatorHome>/etc/mediator.properties` with the "lanid" property.

- 2** Point your browser to `http://<commander_host>t:2006/registrar/add_proxy?action=display` to add a proxy for a specific LAN ID. Enter the "LAN ID" the proxy should be used for, the Host Name of the proxy host, the Port of the proxy and the protocol support of the proxy (http or https). Click Submit when finished. Repeat this process for each LAN ID -> Proxy combination.
- 3** If any proxy server entered in the previous step is HTTPS, import the proxy server's certificate into the Mediator's JRE cacerts keystore.

```
<mediator_install_dir>/jre/bin/keytool -keystore -import -file <ProxyCertificateFile>  
-keystore <mediator_install_dir>/jre/lib/security/cacerts -alias PROXY
```

- Enter keystore password: changeit
- Trust this certificate? [no]: yes

## Proxy Server Configuration for HTTPS

If you've configured any Diagnostics component to communicate with other components using HTTPS and you are proxying, you will need to import the public certificate of the remote component into your proxy server.

To extract the public certificate for any component, run the following command using the JRE for the component. For example, the following will extract the public certificate for the Commander:

```
<commander_install_dir>/jre/bin/keytool -keystore  
<commander_install_dir>/etc/key  
store -storepass <password> -alias SERVER -export -rfc -file <Commander  
Home>/etc/public.key
```

---

**Note:** Make sure that you have already setup the component for SSL. See “Configuring Diagnostics Components for HTTPS” on page 413 for more information.

If you need to export the certificate in a certain formation, instructions for how to do this can be found at:  
<http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/keytool.html>.

---



---

# Diagnostics Component Configuration and Communication Diagrams

This appendix provides you with a diagram to assist you as you install and configure the Diagnostics components.

---

**Note:** The following configuration information is intended for experienced users with in-depth knowledge of Diagnostics for J2EE & .NET. The diagrams are intended to provide a high level view, not provide an in depth knowledge of the working of the components.

---

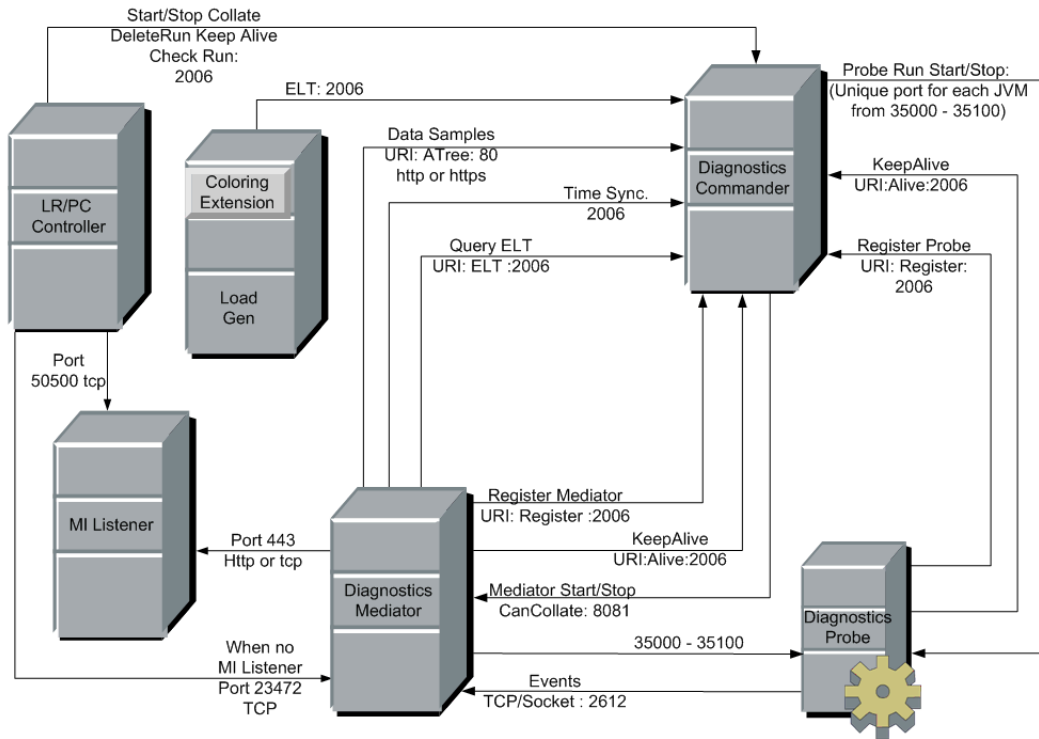
The appendix includes the following sections:

- ▶ Diagnostics Component Configuration and Communication

# Diagnostics Component Configuration and Communication



## AD: LoadRunner / PC Diagnostics Configuration/Communication





---

# **Diagnostics Upgrade Paths for Mercury Products**

This appendix provides you with the recommended upgrade paths for upgrading your current Mercury products so that you can use this version of Mercury Diagnostics.

The appendix includes the following sections:

- ▶ Overview of Product Upgrade Paths
- ▶ General Upgrade Recommendations
- ▶ Product Upgrade Paths

## Overview of Product Upgrade Paths

The following instructions have been provided to help you understand how to upgrade your current Mercury products so that you can take advantage of the new features of Mercury Diagnostics.

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 a case by case basis. Each product deployment is unique and the answer for one situation may not apply to a similar situation.
- ▶ When you determine that you must upgrade a Probe you should also upgrade Mediator to make sure that you can take advantage of all of the latest features.
- ▶ You should contact Mercury Customer Support when you need to upgrade from an earlier version of Mercury's Application Management Products, Topaz and Business Availability Center.
- ▶ Be sure to check the system requirements for the new components before you install them on the host that was used for the earlier version of the component



## Product Upgrade Paths

Current Product	Current Mercury Diagnostics Version	Upgrade Path
LoadRunner 7.8 with Transaction Breakdown	None	<ul style="list-style-type: none"> <li>• Upgrade LR to 8.1</li> <li>• Uninstall All Diagnostics Components</li> <li>• Install All D3.6 Diagnostics Components</li> </ul>
LoadRunner 8.0	None	<ul style="list-style-type: none"> <li>• Upgrade LR to 8.1</li> <li>• Install All D3.6 Diagnostics Components</li> </ul>
	D3.0, D3.5	<ul style="list-style-type: none"> <li>• Upgrade LR to 8.1</li> <li>• Uninstall Diagnostics Server, Database, Mediator</li> <li>• Install D3.6 Commander and Mediator</li> <li>• Upgrade J2EE Probes to D3.6 using Manual Upgrade Process</li> </ul>
LoadRunner 8.1	None	<ul style="list-style-type: none"> <li>• Install all D3.6 Diagnostics Components</li> </ul>
Business Availability Center 4.5 FP1 & FP2 Standard	None	<ul style="list-style-type: none"> <li>• Uninstall BAC 4.5</li> <li>• Install BAC 5.x</li> <li>• Install all D3.6 Diagnostics Components</li> </ul> <p>Note: There is no data upgrade path.</p>

Current Product	Current Mercury Diagnostics Version	Upgrade Path
Business Availability Center 4.5 FP2 Special Edition	R0, R1	<ul style="list-style-type: none"> <li>• Uninstall Probe</li> <li>• Install D3.6 Probe</li> <li>• Uninstall Mediator</li> <li>• Install D3.6 Mediator</li> </ul>
Business Availability Center 5.X	D3.0	<ul style="list-style-type: none"> <li>• Uninstall Probe</li> <li>• Install D3.6 Probe</li> <li>• Uninstall Mediator</li> <li>• Install D3.6 Mediator</li> </ul>
Deep Diagnostics 3.x	D3.x	<ul style="list-style-type: none"> <li>• Upgrade J2EE Probes to D3.6 using Manual Upgrade Process</li> </ul>

---

# Index

## A

- add-in
  - installing 49
- advanced Commander configuration 341
- advanced Mediator assignment 361
- advanced Mediator configuration 347
- application server configuration 147, 161
- Astra LoadTest Readme file xi
- Automatic Method Trimming
  - controlling 380
  - depth 381
  - latency 380

## B

- before you install 15
- breaking down J2EE/.NET diagnostics data 276

## C

- capture points file 369, 385
- changing java executable (Windows) 130
- ClassLoader class, recreating 318
- Commander 6
  - configuring for multi-home 343
  - installing 31, 32
  - UNIX installation 38
- Commander configuration
  - advanced 341
- configuring application servers 147, 161
  - generic procedure 194
  - JBoss 189, 192
  - multiple instances 196
  - Oracle9i 185
  - WebLogic 177
  - WebSphere 163

- configuring Mediators (advanced) 347

## D

- data flow diagram 7
- default Mediator assignment 361
- depth trimming 381
- Diagnostics Commander 6
  - installing 31
- Diagnostics Distribution dialog box 212, 323
- Diagnostics for J2EE
  - capture points file 369, 385
- Diagnostics for J2EE/.NET
  - about 4
  - data flow 7
  - diagram of data flow 7
  - overview 3
  - processing data 8
  - setup window 33, 52, 59, 81, 101
  - troubleshooting 315
  - viewing data 225
- Diagnostics for J2EE/.NET graphs
  - working with 225, 239
- Diagnostics Mediator 6
  - installing 57
- direct Assignment 361
- direct buffer allocation limit 349
- disable Probes 214, 223, 411

## E

- enable Probes 214, 223
- End of Logical Transaction 8

## G

### graph

- J2EE Average Method Response Time 295, 305
- J2EE Average Number of Exceptions 300, 310
- J2EE Average Number of Timeouts 301, 311
- J2EE Method Calls per Second 299, 309
- J2EE Time Spent in Element 296, 306
- J2EE Transaction Response Time Server Side 294, 304
- J2EE Transactions per Second 298, 308

### graph types, Analysis

- J2EE/.NET Diagnostics 273–311

## H

heap size 350

## I

- installing Commander 32
- installing the .NET Probe 79
- installing the J2EE Probe 99
- installing the LoadRunner Add-in 49
- installing the Mediator 58

## J

- J2E/.NET Diagnostics graphs
  - class level 279
- J2EE Average Method Response Time graph 295, 305
- J2EE Average Number of Exceptions graph 300, 310
- J2EE Average Number of Timeouts graph 301, 311
- J2EE Method Calls per Second graph 299, 309
- J2EE Probe
  - installing 99
- J2EE Probe throttling
  - controlling 382

## J2EE Probes

system requirements 20

J2EE Time Spent in Element graph 296, 306

J2EE Transaction Response Time Server Side graph 294, 304

J2EE Transactions per Second graph 298, 308

J2EE/.NET Diagnostics graphs 273–311

J2EE/.NET Diagnostics Configuration dialog box 213, 214, 222, 223, 323, 324, 410, 411

J2EE/.NET Diagnostics graphs

call stack 286

chain of calls 286

example 276

layer level 278

Measurements Tree window 286

method/query level 280

SQL logical name 293, 303

summary report 274

transaction level 277

viewing 276

java executable (Windows)

changing 130

JDK/JRE executable 116

## L

latency trimming 380

layers, specifying 369, 385

LoadRunner Add-in

installing 49

Logical LAN ID 71

## M

Mediator 6

configuring for multi-home 357

installing 57, 58

troubleshooting 76

tuning for POC 355

UNIX installation 67

Mediator assignment

advanced 361

- default 361
- Mediator configuration, advanced 347
- Mediator configuration, large installations 348
- Mediator garbage collection, tuning 353
- Mediators (Windows)
  - starting and stopping 73
- Mercury Interactive on the Web xi
- Multiple Assignment 362

**N**

- NET(.NET) Probe, installing 79
- NET(.NET) Probes
  - system requirements 20

**O**

- overview 3

**P**

- Probe
  - disable 214, 223, 411
  - enable 214, 223
- Probe event buffer size 348
- Probes
  - incorrect installation 318
  - installing on z/OS 130
  - UNIX installation 119
- processing J2EE/.NET data 8

**Q**

- QuickTest Readme file xi

**R**

- Remote Function Call (SAP)
  - offline diagnostics (Analysis) 290
  - online diagnostics 271
- requirements
  - .NET Probes 20
  - J2EE Probes 20
- resources
  - Astra LoadTest Readme file xi
  - Mercury Interactive on the Web xi

- QuickTest Readme file xi
- RFC (SAP), *See* Remote Function Call (SAP)

**S**

- SAP Remote Function Call, *See* Remote Function Call (SAP)
- Show Load Using Scale 336
- starting Mediators (Windows) 73
- stopping Mediators (Windows) 73
- Summary report
  - J2EE/.NET Diagnostics graphs 274
- System Health Check
  - additional component information 332, 334
  - adjusting refresh rate 328
  - Commander properties 330
  - disabling automatic refresh 328
  - enabling automatic refresh 328
  - exporting snapshots 337
  - icons 327
  - importing snapshots 338
  - invoking 322
  - invoking from browser 322
  - legend 327
  - manual refresh 328
  - Mediator properties 330
  - Probe properties 331
  - refreshing display 328
  - system log information 335
  - using 321
  - viewing 326
  - viewing component properties 329
  - viewing troubleshooting information 334

**T**

- throttling
  - J2EE Probe 382
- troubleshooting 315
- troubleshooting Mediators 76
- Troubleshooting Tips, viewing 334

## Index

### **U**

- Unique Probe Name 105
- UNIX installation of Commander 38
- UNIX installation of Mediator 67
- UNIX installation of Probes 119

### **V**

- View Log History 335
- View Troubleshooting Tips 334

### **W**

- Web Service support 313
- Web Service transactions
  - writing VuGen scripts 314
- Web Service transactions,breakdown 313
- Web Services 313
- Weblogic 6.1 177
- Weblogic 7 179
- WebLogic 8.1 181

### **Z**

- z/OS
  - installing the Probe 130