

OPTIMIZE

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

VERSION 4.2

User's Guide

MERCURY™

BUSINESS TECHNOLOGY OPTIMIZATION

Mercury Diagnostics

User's Guide

Version 4.2

Document Release Date: August 28, 2006

MERCURY™

Mercury Diagnostics User's Guide Version 4.2

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.

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

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

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

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

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

© 2004 - 2006 Mercury Interactive Corporation, All rights reserved

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

Table of Contents

Welcome to This Guide	ix
How This Guide Is Organized	ix
Who Should Read This Guide	x
Mercury Diagnostics Online Documentation	xi
Additional Online Resources	xiii
Documentation Updates	xiii
Typographical Conventions	xiv

PART I: INTRODUCTION

Chapter 1: Mercury Diagnostics Product Overview	3
Introducing Mercury Diagnostics	4
Diagnostics Solutions	5
Diagnostics Components and Data Flow	8
Interpreting Diagnostics Data	11
Diagnostics Views at a Glance	11

PART II: USING MERCURY DIAGNOSTICS

Chapter 2: Introducing Diagnostics Views	17
About Mercury Diagnostics Views	18
Accessing the Mercury Diagnostics Views	19
Common Diagnostics View Controls	23
Introducing Diagnostics Detail Views	28
Introducing Diagnostics Dashboard Views	29
Introducing the Diagnostics Status View	30
Exporting Views to HTML Reports	31
Using Table Header Controls	32
Viewing Messages in Diagnostics Message Box	36

Chapter 3: Detail Views: Layout and Controls.....	37
About the Diagnostics Detail Views.....	37
Common Features of the Diagnostics Detail Views.....	38
Working with View Filters	41
Working with the Metrics Inspector.....	48
Working with Graphs.....	56
Controlling the Graphed Metrics.....	65
Working with the Detail Table.....	72
Chapter 4: Alert Notification	81
About Alert Notification.....	81
Configuring Alert Notification.....	82
Working with Alert Notification Rules	85
Reviewing Alert Notification Events	93
Chapter 5: Using the Standard Detail Views	97
About the Standard Detail Views	98
Using the Hosts View	98
Using the Load View	103
Using the Probes View.....	109
Using the Server Requests View	115
Using the Transactions View.....	126
Using the Call Profile View	130
Using the Layer View	140
Using the Portal Components View.....	146
Chapter 6: Using the Status View	153
About the Predefined Status View.....	154
Accessing the Status View	155
Customizing the Status View	156
Interpreting the Status View	157
Drilling Down from the Status View.....	159
Chapter 7: Using the Specialized Views	163
Using the Portal Views	163
Using the Web Services Views.....	167
Using the SAP Views.....	173
Using the Oracle Views	179
Using the BEA WebLogic Views	184
Using the IBM WebSphere Views.....	185
Using the CICS Views.....	187

Chapter 8: Customizing Diagnostics Views	189
About Mercury Diagnostics Custom Views.....	190
Understanding Diagnostics Custom View Retention	190
Creating View Groups	191
Saving a Customized View	192
Creating a New View	194
Renaming a View.....	199
Deleting a View	200
Modifying a Custom View	201
Sharing Custom Views	201
Upgrade of Custom Views From Previous Versions.....	202
Chapter 9: Instance Trees	203
Introducing Instance Trees.....	204
Solving Problems with Instance Trees.....	205
Cross-VM Trees.....	209
When Instance Trees Are Not Sufficient	211
Aggregate Trees.....	212

PART III: USING THE MERCURY DIAGNOSTICS PROFILER FOR J2EE

Chapter 10: Using the J2EE Diagnostics Profiler.....	221
Accessing the Mercury Diagnostics Profiler for J2EE	222
Mercury Diagnostics Profiler for J2EE Processing	223
Common J2EE Diagnostics Profiler Tab Navigation and Display Controls	225
Chapter 11: Analyzing Method Latency with J2EE Diagnostics Profiler Screens	227
Analyzing Performance Using the Summary Tab	228
Analyzing Performance Using the Hotspots Tab	232
Analyzing Performance Using the Metrics Tab.....	234
Analyzing Performance Using the All Methods Tab.....	236
Analyzing Performance Using the All SQL Tab	240
Analyzing Performance Using the Exceptions Tab.....	242
Analyzing Performance Using the Server Requests Tab.....	244
Analyzing Performance Using the Call Profile Window.....	247
Analyzing Performance Using the Web Services Tab	254
Chapter 12: Analyzing Memory with J2EE Diagnostics Profiler Screens.....	257
Analyzing Memory Using the Collections Tab	257
Analyzing Memory Using the Heap Breakdown Tab.....	262

PART IV: USING THE MERCURY DIAGNOSTICS PROFILER FOR .NET

Chapter 13: Using the .NET Diagnostics Profiler.....267
Accessing the .NET Diagnostics Profiler.....268
Mercury Diagnostics Profiler for .NET Processing269
Common .NET Profiler Tab Navigation and Display Controls270

**Chapter 14: Analyzing Method Latency
with .NET Diagnostics Profiler Screens273**
Analyzing Performance Using the Server Requests Tab.....274
Analyzing Performance Using the SQL Tab279
Analyzing Performance Using the Methods Tab281
Analyzing Performance Using the Exceptions Tab.....284
Analyzing Performance Using the Call Tree Tab286

**Chapter 15: Analyzing Memory Using
.NET Diagnostics Profiler Screens.....293**
Analyzing Memory Using the Collections Tab293
Analyzing Memory Using the Heap Tab.....301

PART V: DIAGNOSTICS INTEGRATION WITH OTHER MERCURY PRODUCTS

**Chapter 16: Viewing Diagnostics Data in
Mercury Business Availability Center309**
About Viewing Diagnostics Data in
Mercury Business Availability Center310
Accessing the Diagnostics Screens311
Monitoring Diagnostics Performance Data from Dashboard311
Drilling down to Diagnostics from Dashboard.....319
Diagnostics Performance Reports in Mercury Business
Availability Center323
Drilling down to Diagnostics Data from Mercury Business
Availability Center Reports327

Chapter 17: Viewing Diagnostics Data in LoadRunner335
About Viewing Mercury Diagnostics Data in LoadRunner 8.1335
Configuring LoadRunner Scenarios to use Mercury Diagnostics336
Drilling Down to Diagnostics Data During a Load Test Scenario340
Analyzing Offline Diagnostics Data Using Mercury
LoadRunner Analysis343

Chapter 18: Viewing Diagnostics Data in Performance Center	345
About Viewing Mercury Diagnostics Data in Performance Center 8.1	346
Configuring Performance Center Load Tests to Use Mercury Diagnostics	347
Drilling Down to Diagnostics Data During a Load Test	351
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	355
Index	357

Table of Contents

Welcome to This Guide

Welcome to the *Mercury Diagnostics User's Guide*. This guide describes how to use Mercury Diagnostics to analyze the performance of your enterprise applications.

Mercury Diagnostics 4.2 enables you to monitor and diagnose the performance of your enterprise applications. The product gathers the metrics that you want to monitor and presents them in graphs and reports that enable you to see how your application is processing the data and how it is utilizing your systems resources. When you discover an anomaly or an issue that you want more information about, Mercury Diagnostics enables you to drill down to the graphs to find the transactions and methods that are contributing to the performance results you are seeing.

For more information about Mercury Diagnostics, see Chapter 1, “Mercury Diagnostics Product Overview.”

How This Guide Is Organized

This guide contains the following parts:

Part I Introduction

Provides a high level overview of the features, components, architecture, and outputs of Mercury Diagnostics.

Part II Using Mercury Diagnostics

Describes how to use Mercury Diagnostics once the components are installed and configured.

Part III Using the Mercury Diagnostics Profiler for J2EE

Describes how to use the Mercury Diagnostics Profiler for J2EE.

Part IV Using the Mercury Diagnostics Profiler for .NET

Describes how to use the Mercury Diagnostics Profiler for .NET.

Part V Diagnostics Integration with Other Mercury Products

Describes how to use Mercury Diagnostics when it is integrated with other Mercury products.

Who Should Read This Guide

This guide is intended for users of Mercury Diagnostics, responsible for optimizing application performance and availability.

This document assumes that you are moderately knowledgeable about enterprise application development and highly skilled in enterprise system and database administration.

Mercury Diagnostics Online Documentation

Your Mercury Diagnostics application comes with the following documentation:

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

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

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

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

Additional Online Resources

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

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

Documentation Updates

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

To download updated documentation:

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

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

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

Typographical Conventions

This guide uses the following typographical conventions:

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

Part I

Introduction

1

Mercury Diagnostics Product Overview

This chapter introduces you to Mercury Diagnostics 4.2. It provides an overview of its features, components, architecture, and output.

This chapter describes:	On page:
Introducing Mercury Diagnostics	4
Diagnostics Solutions	5
Diagnostics Components and Data Flow	8
Interpreting Diagnostics Data	11
Diagnostics Views at a Glance	11

Introducing Mercury Diagnostics

Mercury Diagnostics is a composite application triage and diagnostics solution that is designed to help you improve the performance of your J2EE, .NET, and ERP/CRM enterprise applications throughout the application lifecycle. Mercury Diagnostics enables you to:

- ▶ detect slow performing components and code in pre-production or production.
- ▶ gain end-to-end visibility of composite components through transaction tracing across multiple tiers.
- ▶ isolate the performance of incoming requests and correlate them with outbound requests.
- ▶ measure latency at service-consumer level and service-provider level.
- ▶ discover "rogue" code/components real-time as they are invoked.
- ▶ reduce Mean Time To Repair (MTTR) by shortening the Composite Application Triage time.

Mercury Diagnostics provides solutions for:

- ▶ **J2EE Applications** that run on most of the J2EE-compliant application servers. Diagnostics collects performance metrics on the servlets, JSPs, EJBs, JNDI, JDBC, JMS, and Struts method calls that are performed by your application. You can customize the instrumentation that Diagnostics uses to monitor your applications so that it will capture specific business logic methods.
- ▶ **.NET Applications** that run on the Microsoft .NET Framework. Diagnostics uses runtime instrumentation to capture method latency information from specified applications. By default, Diagnostics captures methods from ASP, ADO, and MSMQ. You can customize the instrumentation that Diagnostics uses to monitor your applications so that it will capture specific business logic methods.
- ▶ **ERP/CRM Systems** including Oracle 10g Database and SAP R/3 systems.
- ▶ **SAP NetWeaver.** Mercury Diagnostics provides end-to-end analysis of SAP Enterprise Portal transactions starting from portal pages and iViews, while maintaining the business context of services provided through the

portal. Diagnostics also supports tracing and diagnosis of Java WebDynPro applications hosted on the Portal.

Mercury Diagnostics has been integrated with Mercury's Application Delivery and Application Monitoring solutions to provide you with the insight and information that you need to build, develop, test, and monitor applications that perform efficiently and effectively.

Diagnostics Solutions

You can configure Mercury Diagnostics to work with one of Mercury's Application Delivery or Application Monitoring products or you can use it as a stand alone diagnostics tool.

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

Integration with Mercury Business Availability Center

Mercury Diagnostics is integrated with Mercury Business Availability Center, allowing you to monitor the availability and performance of your production enterprise application. This integration enables you to significantly reduce the mean time to resolution of problems and thus increase the availability and value of the business applications.

From within Mercury Business Availability Center, you can track the performance status of your applications that are being monitored by Mercury Diagnostics.

The Diagnostics integration with Mercury Business Availability Center allows you to drill down to Diagnostics data from specific Mercury Business Availability Center configuration items and reports. You can also generate high level reports in Mercury Business Availability Center about the performance of applications and Business Process Monitor (BPM) transactions that are monitored by Diagnostics.

For more information about the Diagnostics integration with Mercury Business Availability Center, see Chapter 16, “Viewing Diagnostics Data in Mercury Business Availability Center.”

Integration with LoadRunner / Performance Center

Mercury Diagnostics is integrated with LoadRunner and Performance Center to provide QA teams the power of load testing with the added advantage of developer ready reporting that facilitates collaboration across silos.

During a load test, you can drill down to Mercury Diagnostics data for the whole scenario or for a particular transaction. After you have run your scenario, you can use Mercury LoadRunner Analysis to analyze offline Diagnostics data generated during the scenario.

For more information about the Diagnostics integration with LoadRunner, see Chapter 17, “Viewing Diagnostics Data in LoadRunner.” For more information about the Diagnostics integration with Performance Center, see Chapter 18, “Viewing Diagnostics Data in Performance Center.”

Diagnostics Standalone

You can also work with Diagnostics as a standalone product. When you work in Diagnostics Standalone, you access the Diagnostics views directly from the Diagnostics Server in Commander mode.

When you are using Diagnostics Standalone, the views that are available and the information displayed in the views is similar to what you would see if Diagnostics was set up to work with another Mercury product except that there is no metric information displayed for user defined business processes known as *transactions*. The transaction metrics are not available in Diagnostics Standalone because the transactions are generated in LoadRunner, Performance Center or the Mercury Business Availability Center Business Process Monitor.

Note: If you are using Diagnostics Standalone and would like to be able to see the transaction metrics, contact your Mercury Representative to enquire about integrating Diagnostics with one of Mercury's Application Delivery and Application Monitoring products listed above.

Mercury Diagnostics Profiler

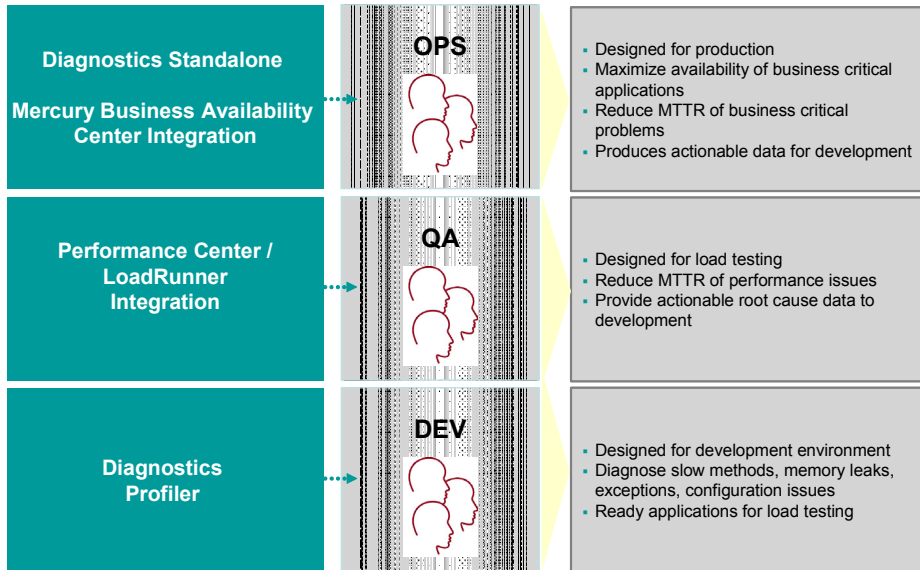
Mercury Diagnostics Profiler uses the Diagnostics Probe to provide a development ready profiling capability that can be integrated into the product/JVM as part of the application lifecycle. Among other things, the Profiler helps you identify:

- ▶ where time is spent in an application.
- ▶ memory problems and garbage collection issues.
- ▶ the slowest layers.
- ▶ the slowest server request or application entry points.
- ▶ outliers to help diagnose intermittent problems.
- ▶ the fastest growing and largest size collections.
- ▶ leaking objects, object growth trends, object instance counts, and the byte size for objects.
- ▶ the slowest SQL query and report query information.
- ▶ exception counts and trace information which often go undetected.

For more information about using the Mercury Diagnostics Profiler for J2EE, see Chapter 10, "Using the J2EE Diagnostics Profiler." For more information about using the Mercury Diagnostics Profiler for .NET, see Chapter 13, "Using the .NET Diagnostics Profiler."

Overview of Diagnostics Solutions by Role

The following diagram illustrates how the Diagnostics solution can be used across different parts of the organization.



Diagnostics Components and Data Flow

Mercury Diagnostics consists of the following components:

- **Mercury Diagnostics Probe.** Responsible for capturing events from your application and sending the performance metrics to a Diagnostics Server. The Probe captures events such as method invocations, collection sites, the beginning and end of business transactions and server transactions. The .NET and J2EE Probes also provide the Profiler functionality which provides detailed diagnostics information about the application that is being monitored by the Probe. This information can be viewed from within the Diagnostics UI or from the Profiler's own UI.

To gather data from external ERP/CRM environments (SAP R/3 system or Oracle 10g Database), you install the Diagnostics Collector and define specific instances of Oracle 10g and SAP R/3 systems to be monitored.

Each instance of a collector is represented as a Probe in the Mercury Diagnostics UI.

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

The Diagnostics Server pre-processes and aggregates the raw data that it receives from the Probes, and formats the information so that it can be displayed in the views of the user interface. Since the Diagnostics Server does the processing for the Probes, the Probes have a negligible effect on the performance of the applications that they monitor.

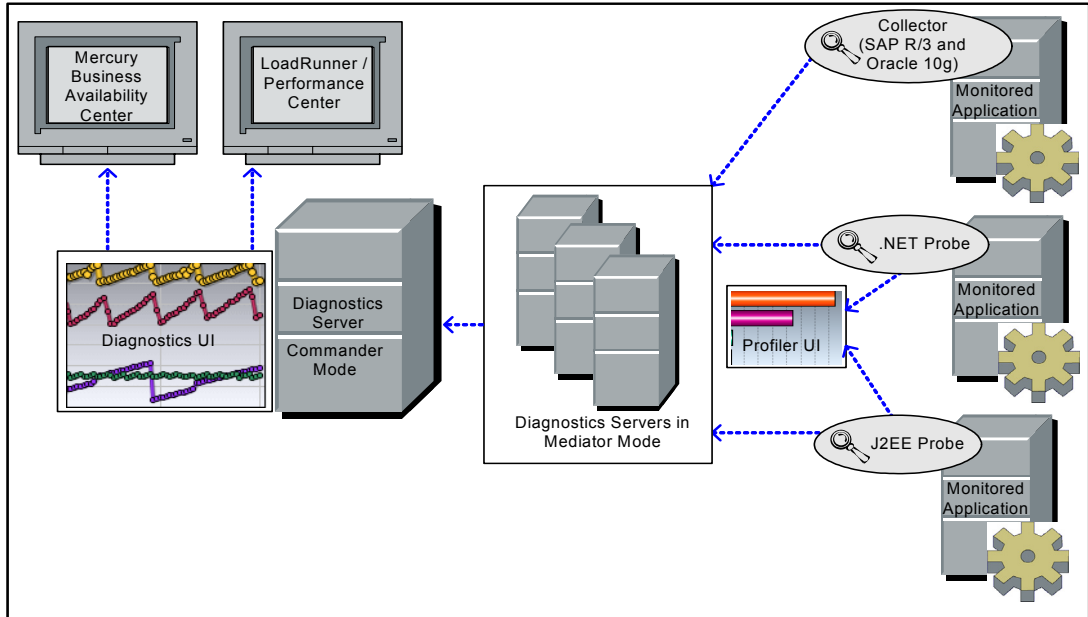
A Diagnostics deployment may consist of one or many Diagnostics Servers. If there is only one Diagnostics server in your deployment, it is configured in Commander mode and must perform both the Commander and Mediator roles. If there is more than one Diagnostics Server in a deployment, one must be configured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

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

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

Diagnostics Data Flow

The following diagram illustrates a distributed Diagnostics deployment consisting of more than one Diagnostics Server.



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

The Probes or Collectors send these captured performance metrics to the Diagnostics Server in Mediator mode where the events are filtered and aggregated. This information is sent to the Diagnostics Server in Commander mode, which displays the processed metrics using graphs and tables that make up the views of the customizable user interface.

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

You access the J2EE Diagnostics Profiler and the .NET Diagnostics Profiler directly from the relevant probe or through the Diagnostics UI.

Interpreting Diagnostics Data

Mercury Diagnostics groups the performance metrics for the classes and methods invoked by your applications into *layers* and *sub-layers* based upon the resources that the application invokes to perform the processing. These layers make it easier for you to isolate and identify the areas of the system that may be contributing to performance issues.

The methods and classes that are associated with each layer are defined in the `<probe_install_dir> auto_detect.points` file where the instrumentation for the probe is specified.

For more information about Diagnostics layers and how classes and methods are assigned to layers in the Capture Points file, see the *Mercury Diagnostics Installation and Configuration Guide*.

For more information on viewing the performance metrics by layer, see “Using the Layer View” on page 140.

Diagnostics Views at a Glance

You access Diagnostics either directly from the Diagnostics Server or via the integrated Mercury Application (LoadRunner, Performance Center, or Mercury Business Availability Center).

Diagnostics presents performance metrics for your monitored applications in the user interface views using both tabular and graphical formats. Diagnostics provides a variety of display controls that allow you to customize the way that you see the data. Diagnostics also displays navigation controls that allow you to drill down into the metrics displayed in a view.

Diagnostics displays data in the following types of views:

- ▶ **Detail view.** Focuses on a specific aspect of your application's performance. This view contains a detailed list of performance metrics for the relevant entities—for example, monitored server requests. A color-coded graph displays trend lines or stacked areas representing selected performance metrics for these entities.

You can control which metrics appear in the graph, and you can set thresholds that determine when to issue performance alerts. You can also filter the graph view based on certain criteria—for example, time range.

From many of the views, you can drill down from one particular entity to other detail views where more specific information for that entity is displayed.

- ▶ **Call profile view.** Displays the method calls and their latency for a particular instance of a monitored server request in your application. The Call Profile view displays the metrics in a graphical call profile, and in a table as a call tree.
- ▶ **Status view.** Displays status and summary information about the monitored applications and host machines in a table. For each level in the table, Diagnostics provides a status indicating how the components that make up the level are performing compared to the thresholds that you set for the performance metrics.
- ▶ **Dashboard view.** Displays two or more detail views and status views in one overall view. The dashboard views allow you to see key performance characteristics of your applications at a glance.

Diagnostics provides many different ways for you to customize the views so that you can filter and focus the information displayed as you analyze performance issues.

For detailed information about working with Diagnostics views, see Part II, “Using Mercury Diagnostics.”

Part II

Using Mercury Diagnostics

2

Introducing Diagnostics Views

This chapter introduces Diagnostics views and explains how to control the form and content of the information displayed in these views.

This chapter describes:	On page:
About Mercury Diagnostics Views	18
Accessing the Mercury Diagnostics Views	19
Common Diagnostics View Controls	23
Introducing Diagnostics Detail Views	28
Introducing Diagnostics Dashboard Views	29
Introducing the Diagnostics Status View	30
Exporting Views to HTML Reports	31
Using Table Header Controls	32
Viewing Messages in Diagnostics Message Box	36

About Mercury Diagnostics Views

Diagnostics presents performance metrics for your applications in views with three main layouts: detail layouts, dashboard layouts, and status layouts. The views present metrics in both tabular and graphical formats and provide a variety of display controls that allow you to customize the way that you see the data, and navigation controls that allow you to drill down to the metrics displayed in a view.

The same display and navigation controls appear on most of the Diagnostics views. This chapter explains how to use these user interface controls. Instructions for the controls that are unique to a particular view are explained in the description for that view in a later chapter in this documentation.

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

About Time Measurements in Diagnostics

It is important to understand how the time measurements displayed in the Diagnostics views are presented. When analyzing the Diagnostics metrics and comparing them with the times reported by other Mercury products, you should consider the following:

- ▶ Response time measurements are not the same in Diagnostics and LoadRunner / Performance Center. Response time on the Diagnostics views 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, a comparison of the response times in a Diagnostics view with the response times on a LoadRunner / Performance Center screen may not match.

- ▶ The transaction times and average times that are displayed on the Diagnostics views include only the time that the transaction spent in the J2EE/.NET server. This means that transaction time does not include the user's think time or the network and Web server latency.
- ▶ Average BP Time is computed at the Business Process Monitor generating the synthetic load and therefore includes the user's think time as well as the network and Web server latency.

Accessing the Mercury Diagnostics Views

You can access Mercury Diagnostics views directly from the Diagnostics Server in Commander mode or from the user interface of other Mercury applications that have been integrated with Diagnostics:

Accessing Diagnostics from Mercury Business Availability Center

Note: Before you can access the Diagnostics views from Mercury Business Availability Center, you must specify the Diagnostics Server details in Mercury Business Availability Center. For more information, see the *Mercury Diagnostics Installation and Configuration Guide*.

You can access Mercury Diagnostics views from Mercury Business Availability Center in one of the following ways:

- ▶ Click **Applications > Diagnostics**.
- ▶ On the **Site Map** menu, click **Diagnostics**.
- ▶ Drill down to Diagnostics data from relevant Mercury Business Availability Center configuration items and reports.

For more information, see Chapter 16, "Viewing Diagnostics Data in Mercury Business Availability Center."

Accessing Diagnostics from LoadRunner

Note: Before you can use Mercury Diagnostics with LoadRunner, you need to ensure that you have specified the Diagnostics Server details in LoadRunner, according to the guidelines set out in the *Mercury Diagnostics Installation and Configuration Guide*.

You can access the Mercury Diagnostics views from LoadRunner in one of the following ways:

- ▶ Click the **Diagnostics for J2EE/.NET** tab at the bottom of the LoadRunner Controller window.
- ▶ Drill down to Mercury Diagnostics data from a particular transaction.

For more information, see Chapter 17, “Viewing Diagnostics Data in LoadRunner.”

Accessing Diagnostics from Performance Center

Note: Before you can use Mercury Diagnostics with Performance Center, you need to ensure that you have specified the Diagnostics Server details in Performance Center, according to the guidelines set out in the *Mercury Diagnostics Installation and Configuration Guide*.

You can access the Mercury Diagnostics views from Performance Center in one of the following ways:

- ▶ Click **View Diagnostics** on the right side of the Performance Center window.
- ▶ Drill down to Mercury Diagnostics data from a particular transaction.

For more information, see Chapter 18, “Viewing Diagnostics Data in Performance Center.”

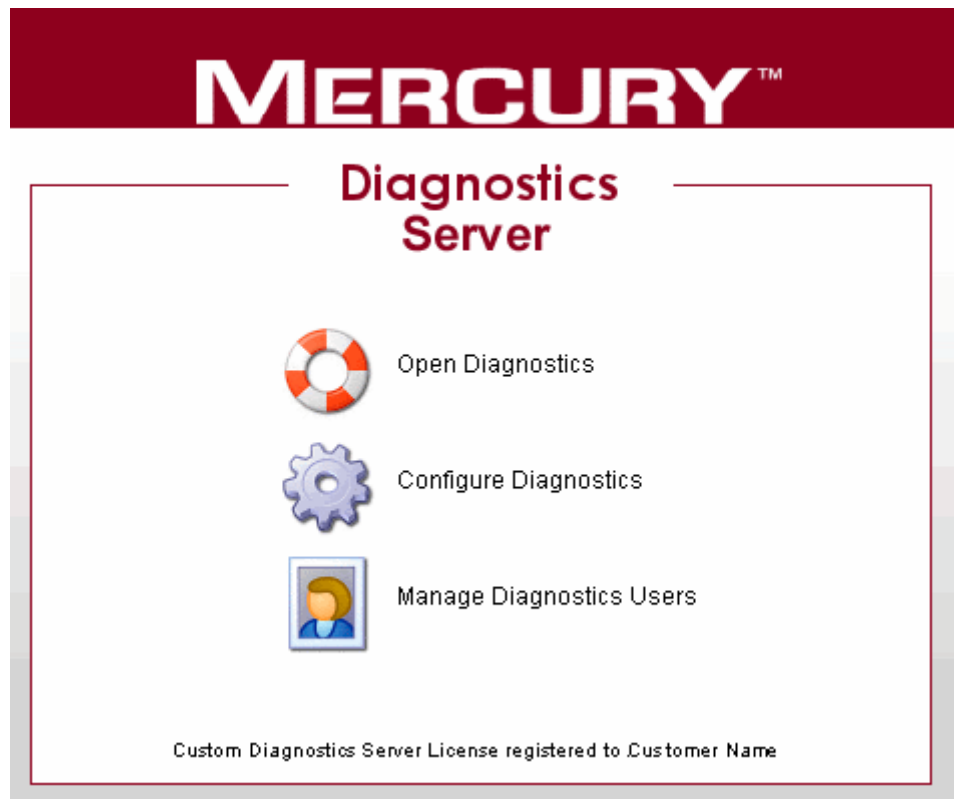
Accessing Diagnostics from the Diagnostics Server in Commander Mode

You can display the Diagnostics views directly from the Diagnostics Server in Commander mode.

To access Mercury Diagnostics from the Diagnostics Server:

- 1 In your browser, navigate to http://<diagnostics_server_host>:2006, or select **Start > Programs > Mercury Diagnostics Server > Administration**. The port number 2006 is the default port for the Diagnostics Server in Commander mode. If the Diagnostics Server was installed and configured to use an alternate port, specify that port number in the URL.

The Diagnostics Server administration page opens.



2 Select **Open Diagnostics**.

If you have not already signed into the Diagnostics Server, you are prompted to provide a user name and password.

Enter the user name and password for a user that has been granted the permissions necessary to open Diagnostics.

A user that has been granted **view** privileges will be able to see the Diagnostics views. One of the default user names that you can use is **admin**; the password for this user is **admin**. For more information about user names and user permissions, see the *Mercury Diagnostics Installation and Configuration Guide*.

Click **OK**.

Note: Diagnostics continues to prompt for a user name and password until you enter valid values.

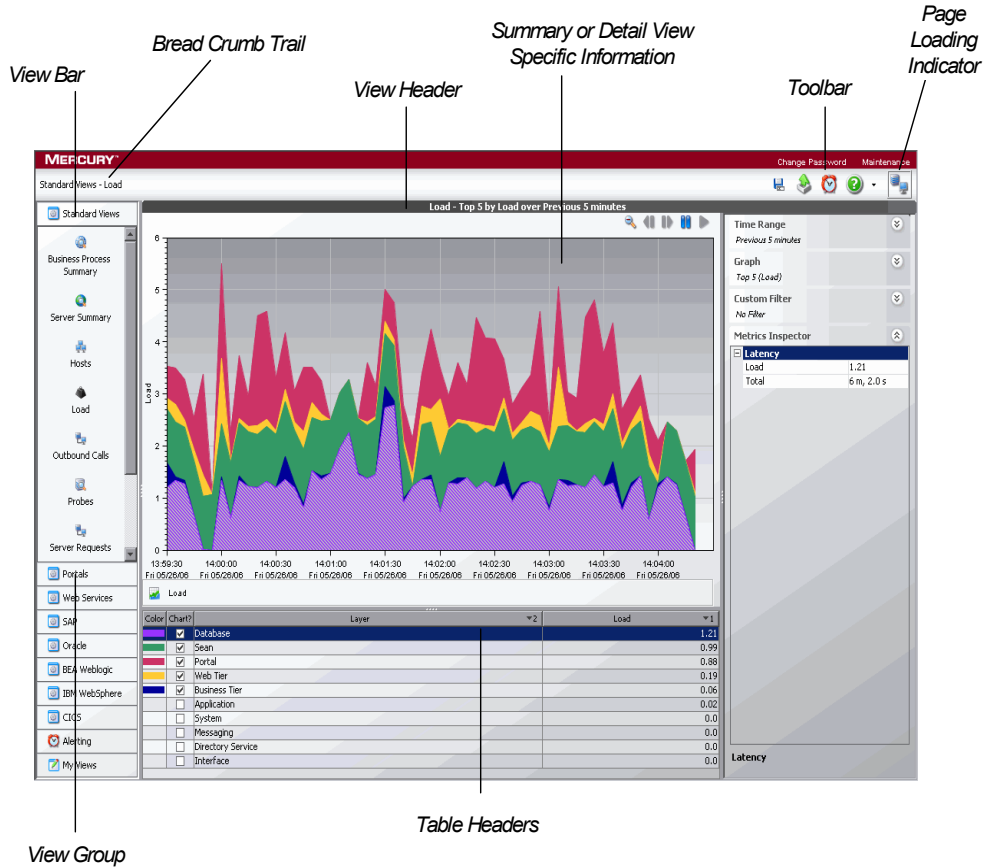
If you click **Cancel**, Diagnostics displays the following error message in your browser: **Access denied. You must specify a valid username and password.**

If the user name and password that you entered are for a valid Diagnostics user that does not have permission to see the Diagnostics views, Diagnostics displays the following error messages in your browser: **Access denied. You do not have the required permission to view this screen.**

- 3** If you are prompted for your user name and password a second time, enter the same information again, and click **Yes**.

Common Diagnostics View Controls

There are features and controls that are common to most of the Diagnostics views. These common features and controls are described in the sections that follow using the following annotated screen image:



Bread Crumb Trail

The bread crumb trail at the top of most of the Diagnostics views provides a convenient way to determine the context for the information that is presented in the view. It also provides a handy method of navigation when you want to retrace your steps in your performance analysis. Clicking a level in the bread crumb trail takes you back to the view for the selected level.

The last level displayed in the bread crumb is not a link because it represents the Diagnostics view that is currently displayed.

For information on drilling down from one view to another, see “Drilling Down To Entities in the Detail Table” on page 78.

View Bar and View Group

The **View Bar** contains icons that represent the Diagnostics views that are available when analyzing the performance metrics for your application. The views are organized into groups to make it easier for you to locate the views that you want to use.

There are predefined view groups that are installed with Diagnostics that contain views that will assist you in diagnosing problems in particular situations or for specific environments. The **Standard Views** view group contains views that are useful in most situations. The other view groups contain views that are useful in more specific situations. All of the view groups, except for **My Views**, contain views that have been predefined to provide you with the ability to look at your application’s performance information and begin analyzing the information the first time that you use Diagnostics. The predefined detail, dashboard, and status views are described in Chapter 5, “Using the Standard Detail Views,” and in Chapter 2, “Introducing Diagnostics Dashboard Views.”

You may save customized versions of the predefined views or create original views using the functions of the view bar. The My Views group is a group in which you may save the views that you have customized. The processes for maintaining customized views and view groups are described in Chapter 8, “Customizing Diagnostics Views.”

Summary or Detail View Specific Information

This section of the Diagnostics view contains the information that is specific to the dashboard, detail, or status view that Diagnostics has displayed. The title in the **View Header** describes the information presented in the view.

For information about the dashboard views, see “Introducing Diagnostics Dashboard Views” on page 29.

For information about the detail views, see “Introducing Diagnostics Detail Views” on page 28.

For information about the status views, see “Introducing the Diagnostics Status View” on page 30.

View Toolbar

The **View Toolbar** contains the following buttons that provide quick access to Diagnostics features that you may want to use frequently:

Save This View



If you have customized the Diagnostics view that is displayed, you may want to save the custom view to use later. The **Save This View** button provides a convenient way for you to save your changes. For more information on saving custom views, see Chapter 8, “Customizing Diagnostics Views.”

Export to HTML



Clicking **Export to HTML** allows you to save the metrics displayed on the current view to a report that can be recalled for viewing or sharing with others. For more information, see “Exporting Views to HTML Reports” on page 31.

Turn On/Off Active Alert Notifications



Turn On/Off Active Alert Notifications is a toggle button that allows you to control whether a message is to be displayed in the Diagnostics message box each time an alert notification is sent. For more information on alerts, see Chapter 4, “Alert Notification.”

Show Help



The **Show Help** button gives you access to the user documentation, to information about the version of Diagnostics that you are using, and to the version of some of the third-party components that have been used in the product. Select the desired option from the menu that Diagnostics displays when you click **Show Help**.

Page Loading Indicator



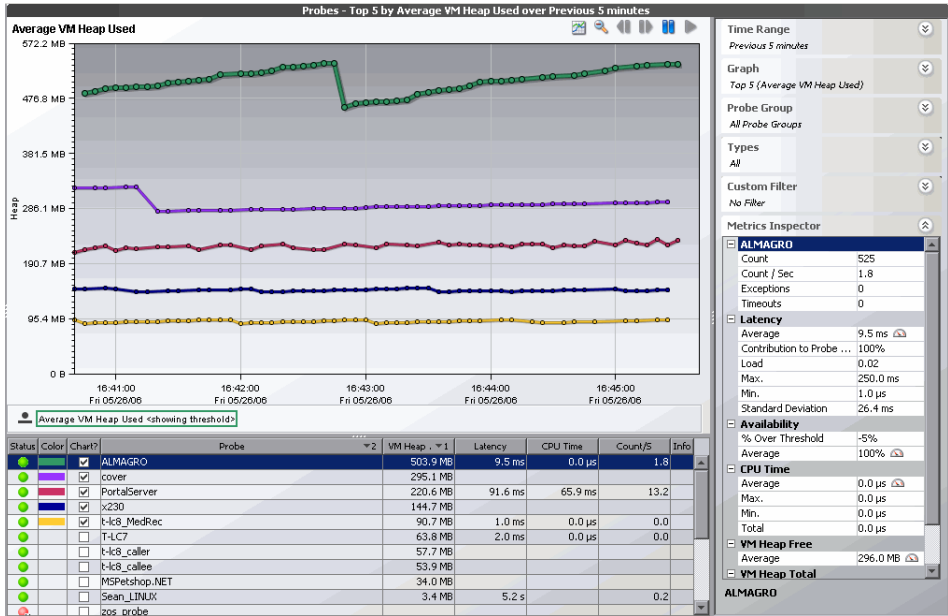
When Diagnostics is retrieving and formatting the data that is to be displayed in a view, the **Page Loading** icon displays two blue arrows spinning in a loop. When the view is refreshed with the formatted data, the arrows are no longer displayed in the icon.

Table Headers

The table headers in the Status and detail views provide controls that enable you to specify which columns are to appear in the table and the order they are to be displayed, as well as which columns are to determine the sort order for the rows in the table. For more information on how to use table header controls to customize the appearance of the data displayed, see “Using Table Header Controls” on page 32.

Introducing Diagnostics Detail Views

Diagnostics comes with a set of standard predefined detail views that each focus on a specific aspect of your application's performance. These views include the *Load view*, *Probes view*, and *Server Requests view*. From many of the views, you can drill down on one particular entity to other detail views where performance information for that entity is displayed. The Probes view shown below is an example of a detail view:

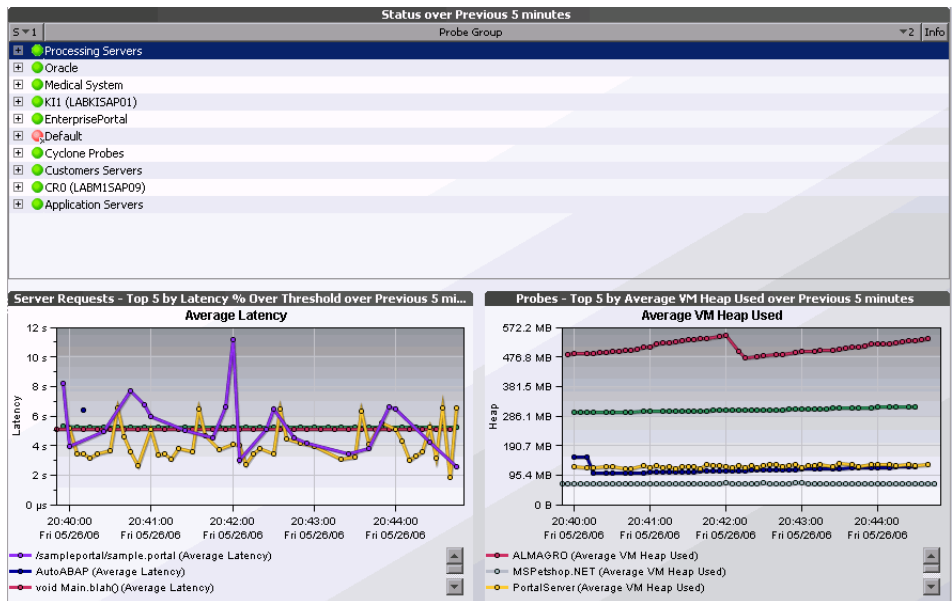


For more information on the features and controls of the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

For information about using a specific standard detail view, see Chapter 5, “Using the Standard Detail Views.”

Introducing Diagnostics Dashboard Views

Dashboard views allow you to see key performance characteristics of your applications at a glance. The Diagnostics predefined dashboard views that are installed with Diagnostics include the *Business Process Summary* view (*LR / PC Summary* view in LoadRunner and Performance Center) and the *Server Summary* view. The dashboard views are made up of two or more detail views and status views. The detail views are displayed in an abbreviated format when they are included in a dashboard view. The Server Summary view displayed below is an example of a dashboard view:



When a standard or customized view appears in a dashboard, it is displayed as a monitoring version of the view where only the graph and the graph legend are shown. Hold your mouse pointer over the nodes of the charted trend lines or over the edges of the stacked areas to display a tooltip with more information about the metrics in the monitoring versions of the detail views. You can also hold your mouse pointer over the items in the legend in the monitoring views to see additional information.

To see the full-size version of a detail view that appears as a monitoring version on a dashboard, double-click the monitoring version of the view. The full-size version of the detail view that you selected is displayed, and the bread crumb indicates that you navigated from the Dashboard view.

For more information on detail views, see Chapter 5, “Using the Standard Detail Views.”

Introducing the Diagnostics Status View

The *Status view* is a table of the probe groups, the probes that are assigned to each probe group, and the hosts for those probes. For each level in the table, Diagnostics provides a status indicating how the components that make up the level are performing compared to the thresholds that you set for the performance metrics.

You can determine the reason for any alert statuses by drilling into the probe group, probe, or host to see the detail view for the component. Below is an example of the Status view:

The screenshot shows the 'Status over Previous 5 minutes' window in Mercury Diagnostics. It displays a hierarchical tree view of the system's status. The tree is expanded to show the 'Default' probe group, which contains several probes. The 'PortalServer' probe is selected, and its details are shown in a table below. The table has columns for VM Heap Used, Latency, Count/S, Timeouts, Exceptions, and Info. The 'PortalServer' row shows 150.7 MB VM Heap Used, 91.8 ms Latency, 12.4 Count/S, 0 Timeouts, and 54 Exceptions. Below the probe details, the 'Host' table is expanded to show the status of various hosts. The 'x231.lab.performant.com' host is highlighted in yellow, showing 75% CPU usage. Other hosts include x230.lab.performant.com (3% CPU), l-ic8.lab.performant.com (10% CPU), l-ic7.lab.performant.com (6% CPU), largent (2% CPU), and balboa.lab.performant.com (4% CPU). The table also shows Disk Bytes/S and Network Bytes/S for each host. The tree view on the left shows other probe groups like 'Processing Servers', 'Oracle', and 'Medical System'. The 'Medical System' group is expanded to show the 'AMKILAB03' probe, which has 28.1 MB VM Heap Used and 307.0 µs Latency. Its details table shows 1 Count/S, 0 Timeouts, and 0 Exceptions. Below the AMKILAB03 details, the 'Host' table is expanded to show the 'lankilab03.lab.performant.com' host with 1% CPU usage, 27.6 KB Disk Bytes/S, and 7.0 KB Network Bytes/S. The tree view on the left also shows other probe groups like 'KI1 (LABK1SAP01)', 'EnterprisePortal', 'Cyclone Probes', 'Customers Servers', 'CR0 (LABM1SAP09)', and 'Application Servers'.

Probe	VM Heap Used	Latency	Count/S	Timeouts	Exceptions	Info
PortalServer	150.7 MB	91.8 ms	12.4	0	54	
x230	124.4 MB	3.1 ms	0.0	0	0	
T-LC7	54.2 MB	16.0 ms	0.0	0	0	
StoreCSVS.NET	68.8 MB					
Sean_LINUX	3.3 MB	5.2 s	0.2	0	0	
MSPetshop.NET	68.8 MB					
JavaTrader.WebClient.NET	44.6 MB	62.0 ms	0.0	0	0	

Host	CPU	Disk Bytes/S	Network Bytes/S	Info
x231.lab.performant.com	75%	40.5 KB	42.4 KB	
x230.lab.performant.com	3%	3.9 KB	2.4 KB	
l-ic8.lab.performant.com	10%	5.5 KB	7.3 KB	
l-ic7.lab.performant.com	6%	14.8 KB	29.6 KB	
largent	2%	3.7 KB	1.7 KB	
balboa.lab.performant.com	4%	3.1 KB	1.7 KB	

Probe	VM Heap Used	Latency	Count/S	Timeouts	Exceptions	Info
AMKILAB03	28.1 MB	307.0 µs	1.9	0	0	

Host	CPU	Disk Bytes/S	Network Bytes/S	Info
lankilab03.lab.performant.com	1%	27.6 KB	7.0 KB	

For more information on the features and controls of the Status view, see Chapter 6, “Using the Status View.”

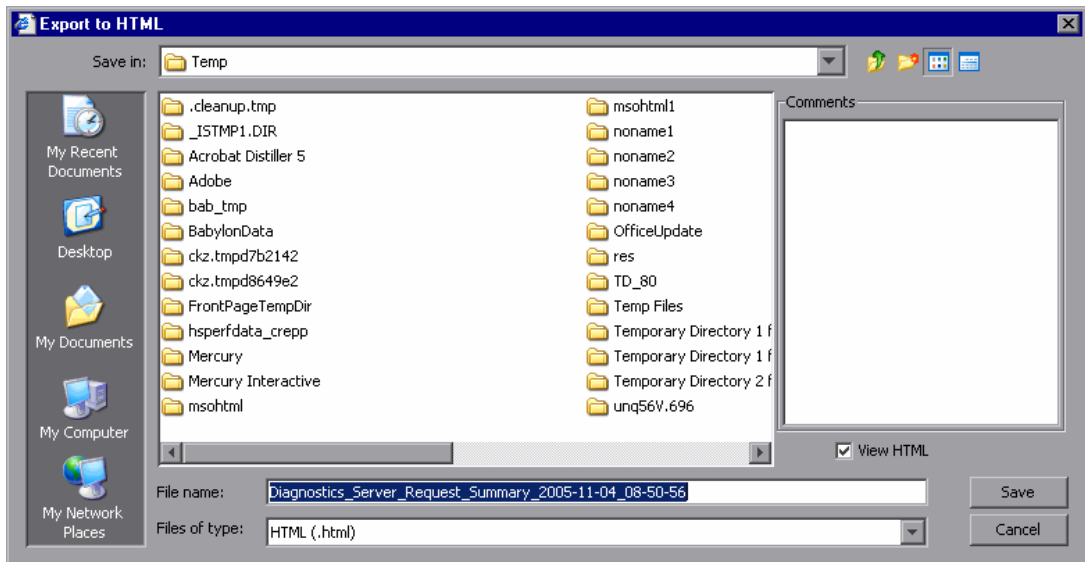
Exporting Views to HTML Reports



You may save the performance metrics displayed in a Diagnostics view by clicking the **Export to HTML** button in the View toolbar on the upper-right side of the view. When the view is exported, the information displayed in the view is formatted as a report and exported to an HTML file. The HTML reports can be saved to view in your browser at a later time, or can be used to share system performance metrics with others.

To export view information to an HTML Report:

- 1 Click **Export to HTML**. The Export to HTML dialog box opens.



- 2 Enter a name for the file to which you are exporting the view's performance metrics.
- 3 In the **Comments** box, enter any comments that you would like to have appear on the report. Your comments may include instructions or questions for the person who is to receive the report.

- 4 If you would like to view the HTML file, select **View HTML**. This causes Diagnostics to display the HTML file in your browser after it has been saved.
- 5 Click **Save** to save the performance metrics of the view, along with any comments that you entered. If you selected **View HTML**, Diagnostics displays the HTML report in your browser.

Using Table Header Controls

The table headers that appear in the Status view and in the detail views provide controls that enable you to specify which columns are to appear in the table and the order in which the columns are to be displayed, as well as specifying which columns are to determine the sort order for the rows in the table.

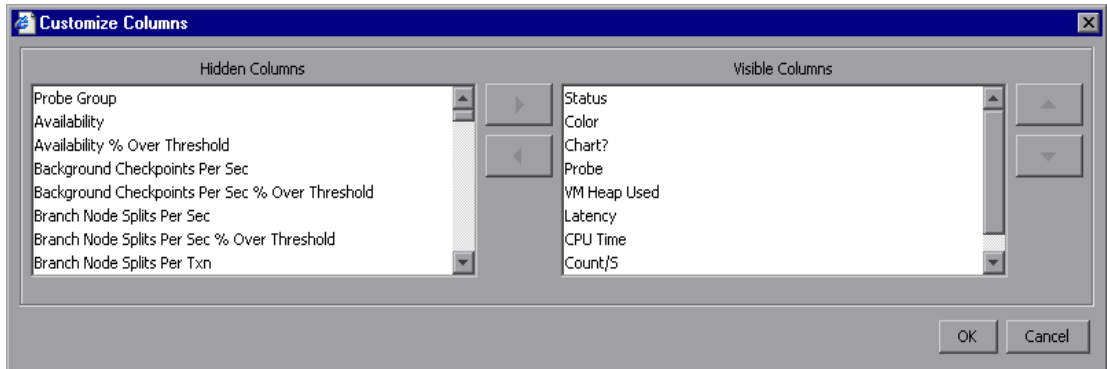
Customizing the Table Columns

The default columns that appear in the Status view and in the detail table of the detail views help you identify the entities that are included in the view, understand the status of each entity's performance, and see some of the performance metrics that have been gathered for each entity.

You may customize the table by choosing the columns that are to appear, and the order in which they are to appear.

To customize the table columns:



- 1 Right-click the table header to display the header menu.
- 2 Click **Customize Columns** to open the Customize Columns dialog box. The Customize Columns dialog box for the Probes view is shown in the following example:



- 3 Make additional columns visible in the table by selecting one or more columns in the **Hidden Columns** list and using the right-arrow button to move your selection to the **Visible Columns** list. Repeat this step until all of the columns that you want to make visible in the table have been moved.

Note: Diagnostics automatically adjusts the width of the columns in the table to make all of the selected columns visible. Some columns may appear too small to see the data. You can resize the column or hold the mouse pointer over the column to see the tooltip with the value of a particular metric in the table.

- 4 Hide columns from the table by selecting one or more columns in the **Visible Columns** list and using the left-arrow button to move your selection to the **Hidden Columns** list. Repeat this step until all of the columns that you want to hide in the table have been moved.

-  **5** Rearrange the order in which the columns are displayed in the table by selecting one or more columns in the **Visible Columns** list, and using the up arrow or down arrow buttons to move the column. Repeat this step until the columns in the table are in the desired sequence.
-  **6** Click **OK** to close the Customize Columns dialog box and save your changes. Click **Cancel** to close the Customize Columns dialog box and discard your changes.
- 7** When the table columns are organized to your satisfaction, you can resize columns or change the sort order to refine your column customization.

Note: If you customize one of the standard detail views, and would like to save the customization for future use, remember to save the view as a custom view. If you do not save the changes as a custom view, they will be lost when you close the Diagnostics UI.

For information on saving custom views, see “Saving a Customized View” on page 192.

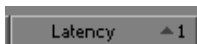
Sorting Rows in the Table

The default sort order for the table is usually ascending order on the first metric column in the table, and then ascending order on the entity name column. For example, in the Probes view, the default primary sort is based on the values in the **VM Heap Used** column, and the secondary sort is based on the values in the **Probe** column. The following example shows the column headers for the Probes view:



Status	Color	Chart?	Probe ^{▼2}	VM Heap Used	Latency ^{▲1}	Count/5
--------	-------	--------	---------------------	--------------	-----------------------	---------

The superscript sort indicators in the column headers. In the example above, the **Latency** column has the superscript up arrow followed by the number **1** as shown in this close-up view:



The up arrow indicates that the column is sorted in ascending order, and the number **1** indicates that this column is the primary sort for the table.

In the same way, you can tell which columns are next in the sort sequence. In the example above, the **Probe** column has the superscript down arrow followed by the number **2** as shown in this close-up view:



The down arrow indicates that the column is sorted in descending order, and the number **2** indicates that this column is the secondary sort for the table.

You can customize the sort order using the controls built into the column headers in the table.

To create a new custom sort order for the table:

- 1** Click the header of the column that will be used for the primary sort. The indicators for the previous sort sequence are removed from all columns in the table. The selected column is marked with a superscript up arrow and the number **1**, indicating that the column is sorted in ascending order, and that the values in the column determine the primary sort sequence.
- 2** To switch from ascending order to descending order click on the column header again. The up arrow switches to a down arrow, indicating that the column is sorted in descending order.
- 3** To remove the column from the sort, click the header one more time. The superscript sort indicator is removed from the column.

Note: When you remove the primary sort column from the sort, any secondary sort orders that you have defined are removed as well.

To add secondary sort sequences for the table:

- 1** Press CTRL and click the header of the column that will be used for the secondary sort. The selected column is marked with an superscript up arrow and a number. The up arrow indicates that the column is sorted in ascending order, and the number indicates where the column falls in the sort sequence.

If this is the first column that you have defined after defining the primary sort column, then the number **2** appears next to the arrow. To any subsequent columns that you add to the sort sequence, an increment superscript is displayed to indicate the sorting order.

- 2** To switch from ascending order to descending order, press CTRL and click the header again. The up arrow changes to a down arrow, indicating that the column is sorted in descending order.
- 3** To remove the column from the sort, press CTRL and click the header one more time. The superscript sort indicator is removed from the column.

Note: When you remove a secondary sort from a column, any other columns that are used as secondary sort columns are renumbered to maintain the sorting sequence.

Viewing Messages in Diagnostics Message Box

Diagnostics displays error and warning messages in the Diagnostics views using the Diagnostics Message box in the bottom right hand corner of the views. The box increases in size as more error messages are added so that all of the messages can be displayed. Closing the Message box clears all of the previous messages.

3

Detail Views: Layout and Controls

This chapter describes the layout of the Diagnostics detail views and explains how to use the controls that are common to most of the detail views.

This chapter describes:	On page:
About the Diagnostics Detail Views	37
Common Features of the Diagnostics Detail Views	38
Working with View Filters	41
Working with the Metrics Inspector	48
Working with Graphs	56
Controlling the Graphed Metrics	65
Working with the Detail Table	72

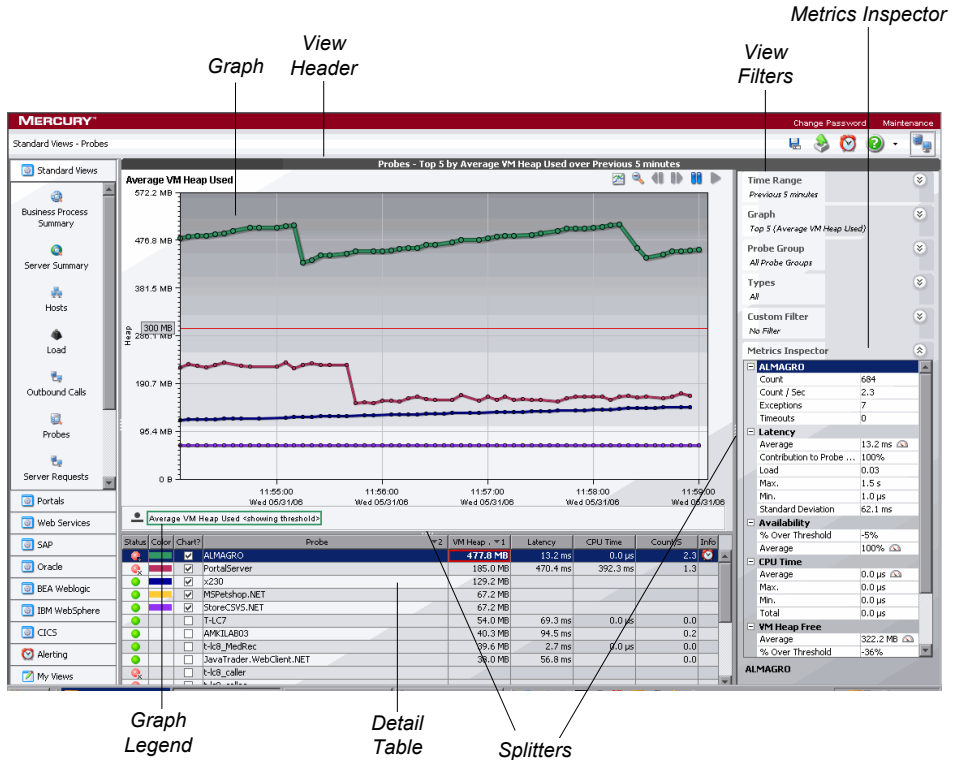
About the Diagnostics Detail Views

Diagnostics comes with a set of predefined detail views, each of which focuses on a specific aspect of your application's performance. These views include the *Load view*, the *Probes view*, and *Server Requests view*. From many of the views, you can drill down on one particular entity to other detail views where performance metrics for that entity are displayed. For example, you can drill down on a server request listed on the Server Request view to see the Layers view for that particular server request.

The detail views share many common features and controls. The sections that follow introduce these common features and controls.

Common Features of the Diagnostics Detail Views

The Diagnostics detail views share many characteristics. These are highlighted in the following image:



Graph

The *graph* on the detail views contains trend lines or stacked areas that represent the performance metrics for the processing represented by your selections from the View Filters, the Metrics Inspector, and the detail table. When you select an entity from the detail table, a trend line or stacked area is charted on the graph for each metric selected from the Metrics Inspector. For more information about the detail view graph in the detail views, see “Working with Graphs” on page 56.

View Header

The *view header* describes the information that is currently displayed in the view. The view name is listed along with the current choice from the **Graph** view filter. For information on the **Graph** view filter, see “Working with View Filters” on page 41.

Graph Legend

The *graph legend* displays the line patterns used to represent metrics on the graph. All of the lines on the graph with a particular pattern are associated with a particular metric from the Metrics Inspector.

You may also use the graph legend to remove a particular metric from the graph. For more information on using the graph legend, see “Working with Graphs” on page 56.

View Filters

The *view filters* are a set of menus that allow you to select the entities whose metrics are to be charted based on time ranges, probe groups, type of probe, and other criteria depending on the view that Diagnostics has displayed. For more information on view filters, see “About View Filters” on page 41.

Metrics Inspector

The *Metrics Inspector* lists the metrics associated with the entity selected in the detail table. From the Inspector, you can control which metrics appear in the graph, and you can set thresholds that determine at what point an alert should be issued to indicate that the value of a performance metric is of concern. For more information about the Metrics Inspector, see “Working with the Metrics Inspector” on page 48. For more information on investigating alert conditions for thresholds that have been exceeded, see “Investigating a Threshold Alert” on page 54.

Detail Table

The *detail table* lists the entities in a particular view for which metrics have been gathered. For example, the entities displayed in the Probes view are probes, and in the Load view are layers. You can select entities to be represented on the graph from the detail table.

The detail table is highly customizable, allowing you to control the columns that appear in the table, as well as the sort order for the table. For more information about working with and customizing the detail table, see “Working with the Detail Table” on page 72.

Splitters

Many of the views have *splitter controls* that enable you to change the proportions of different areas of the view. They allow you to resize parts of the view so that areas of less interest are minimized to provide more area for the information that you want to see.

By dragging the splitter that divides the graph and the detail table, you can change the amount of area used for each of these features.

By dragging the splitter that divides the pane containing the view filters and Metrics Inspector from the graph and detail table pane, you can alter the amount of space used to display these features.

Working with View Filters

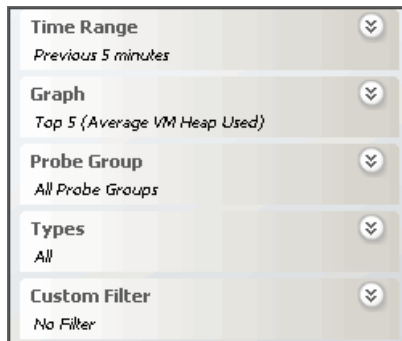
The view filter menus give you control over the information that is displayed in the graph, detail table, and Metrics Inspector. Your selection from the view filter menus that are displayed in each detail view control the time range for the metrics that are to be displayed and which entities are to be charted in the graph along with several other controls that are specific to the view that Diagnostics is displaying.

About View Filters



The view filter area is made up of a set of drop-down menus that you open and close by clicking the double arrow button to the right of each menu title. The filter menus that appear in each view, and the filters that appear in each menu, vary depending on the view that is displayed.

Below is an example of the view filter that was taken from the Probes view. It shows the **Time Range**, **Graph**, **Probe Group**, **Types**, and **Custom** filter menus, with each menu closed.



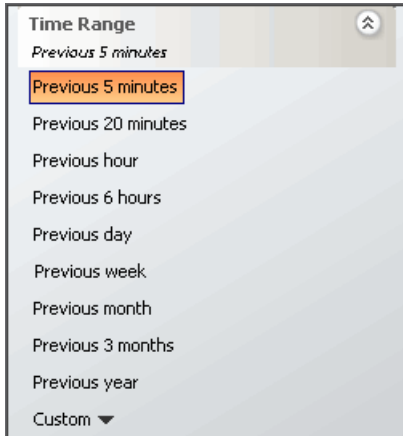
When a filter menu is closed, the current selection for the filter is displayed under the filter title. In the example above, the current selection for the **Graph** filter is *Top 5 (Average VM Heap Used)*.

Filter settings applied on a view are also applied to each of the views to which you drill down from the view where you set the filters. Filter settings that you made at lower levels, after you drilled down, are not applied as you navigate back to higher level views above the view where the filter settings were made.

Each of the available view filter menus is described in the sections that follow.

Filtering the View by Reported Time Range

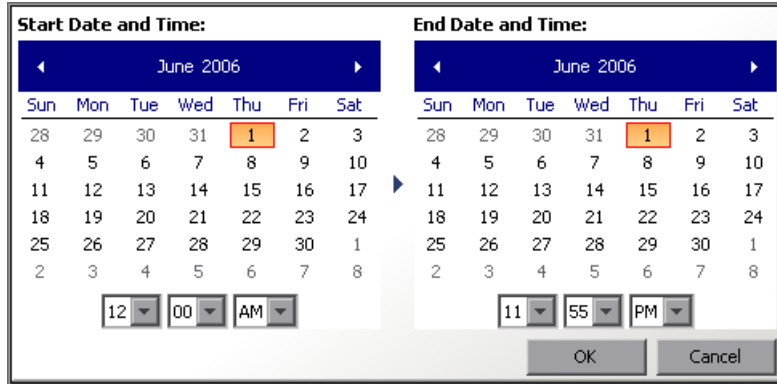
You can control the time range for which performance data is displayed by selecting the desired time period from the **Time Range** view filter. The following image shows an example of the **Time Range** filter options:



Note: If you are using Diagnostics with LoadRunner or Performance Center as part of a load test, the default time range is **Whole Scenario**, which reflects the time that has elapsed since the beginning of the load test.

When you select a **Time Range**, the information on the graph, the detail table, and the Metrics Inspector are updated to correspond to the selected time range resolution. When the metrics for the new time range are displayed in the view, Diagnostics continues to update the view so that the most recent information for the selected time range resolution is always displayed.

When you select the Custom time range filter option two calendars are displayed to allow you to indicate the starting date and time and ending date and time for the metrics that you want Diagnostics to display in the view. The following image shows an example of the Custom time range filter:



When you have made your selections for the start and end dates, click **OK** to instruct Diagnostics to get the entities and metrics that satisfy your request to display in the view.

Chart Update Frequency

When you view the performance metrics it is important to note that the charted metrics are updated at different rates, depending on the time range resolution that you selected. The following table provides the update frequency for each time range resolution:

Time Range Resolution	View Update Frequency
5 minutes	5 seconds
20 minutes	3 minutes
1 hour	4 minutes
6 hours	8 minutes
1 day	11 minutes
1 week or greater	29 minutes

Note: If you are using Diagnostics with LoadRunner or Performance Center, and your selected time range is **Whole Scenario**, the update frequency decreases as the scenario progresses. For example, at the beginning of the scenario, the update frequency will be approximately 5 seconds. However, after an hour into the scenario, the update frequency will have decreased to approximately 4 minutes.

Understanding Time Range Resolution

The actual time period included in the metrics that is displayed in a view are not always limited to the time range resolution that you requested.

When you select **Previous 5 minutes**, the time range resolution for the performance metrics displayed in the other parts of the view is changed so that only information from the previous 5 minutes is included.

However, for any other **Time Range** option, the actual metrics displayed include information for *no less* than the selected time range. The metrics normally include the information for a much wider time period.

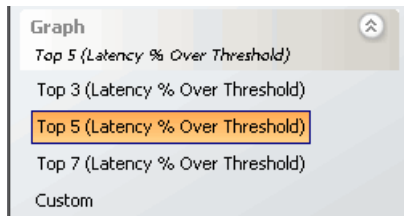
For example, when you select **Previous 20 minutes**, the metrics displayed in the view are never for less than the previous 20 minutes. They almost always include data for the previous 40 to 50 minutes.

If you want to focus on performance for an exact 20-minute period, you can use the zoom features described in “Controlling the Graphed Metrics” on page 65 to select the desired zoom range.

Filtering Charted Entities Based on Relative Metric Values

You use the **Graph** filter to control which entities in the detail table are to have their metrics charted on the graph. The filter selects a given number of entities from the table based on a specific metric or threshold value. When the information in the view is refreshed, the filter is reapplied and the entities that are charted in the graph may change.

The choices in the **Graph** filter vary for each detail view. The following example shows the **Graph** filter menu for the Server Requests view:



When you select any option (except **Custom**) from the **Graph** filter menu, the entity selection in the **Chart** column of the detail table is modified so that only the entities that correspond to the graph filter are selected. At the same time, the graph is updated so that only the metrics for the entities selected in the **Chart** column are charted.

When you select **Custom**, the entities that are selected in the **Chart** column of the detail table become fixed and change only when you select different entities, or when you select another option from the **Graph** filter menu. For more information about selecting entities in the detail table to be charted on the graph, see “Adding and Removing Entities from the Graph” on page 75.

Note: An entity selected in the **Chart** column of the detail table may disappear from the table and the graph if no metrics for that entity were captured during the time range that is displayed in the view.

Filtering the View by Probe Group

You use the **Probe Group** filter to filter metrics in a view by the probe group for which the metrics were captured. The **Probe Group** filter options include all of the probe groups defined for the applications being monitored. You can also select **All Probe Groups** to see metrics for all probe groups at one time. The following is an example of a **Probe Group** filter menu:

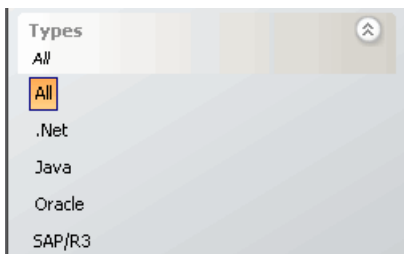


When you select a probe group filter option, the metrics displayed in the view are limited to those for the probes in the selected probe group.

When you select **All Probe Groups**, all performance metrics for all probe groups are displayed.

Filtering the View by Type

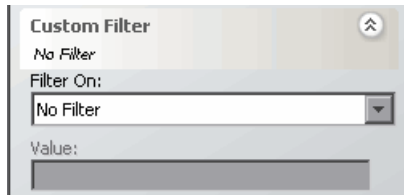
Some of the views include the Type filter in the view filters. The Type filter lets you filter the entities whose metrics are displayed in the view. The filtering criteria listed in the Type filter depends on the entities listed in the detail view. The following is an example of the **Type** filter menu for the Probes view:



The options in this Type filter allow you to filter the probes that are displayed in the view to just those of a certain probe type.

Filtering the View by Custom Filters

The Custom View filter allows you to filter the entities that are displayed in a view based upon the value of particular attributes of the entities. The available custom filters are unique for each detail view. The following is an example of the **Custom** filter menu:



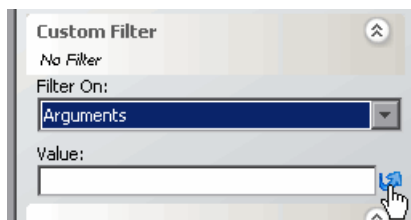
To select a Custom Filter:



- 1** Open the Custom Filter by clicking the double-arrow button in the View Filter header.
- 2** From the Filter On drop-down menu, choose the attribute to use as the filter.
- 3** Enter the value for the attribute that you selected to use as the filter. When you update the value for the attribute, the update icon becomes visible to remind you to apply the custom filter.



- 4** After you have entered the filter value, you can apply the custom filter by pressing ENTER or by clicking the **Apply Custom Filter** button. The following image shows an example the Custom Filter after a filtering attribute has been selected.



The values associated with the entities that are used to filter when you have made a selection from the Customer Filter, can be found in the detail table, in the Metrics Inspector. Alternatively, hold your mouse pointer over the listed entity in the detail table until a tooltip opens.

Working with the Metrics Inspector

The **Metrics Inspector** lists performance metrics for the entity that you selected from the detail table. In addition to providing you with metrics that are useful in your performance analysis, the Metrics Inspector also gives you a way to control the metrics that Diagnostics charts in the graph and to set thresholds for many of the metrics listed. Diagnostics issues a visual threshold violation indicator when the value of a metric exceeds the threshold that you have set.

The following sections provide you with a detailed description of the Metrics Inspector, and explain how to use the controls in the Metrics Inspector to adjust the way that your application's performance is displayed in the detail views.

About the Metrics Inspector

The Metrics Inspector consists of two sections. The top portion of the inspector is a list of metrics and their values. Beneath the list of metrics is the control area where you can indicate how Diagnostics displays a selected metric in the rest of the detail view. The metrics displayed in the inspector can be Java metrics, System metrics, or JMX metrics. The following annotated image is used to describe the features of the Metrics Inspector.

Metric Categories


Threshold Violation Indicator

Metrics Inspector


[-] /topaz/topaz_api/api_GetLicenseInfo.asp

Name	/topaz/topaz_api/api_GetLicenseInfo.asp
Signature	void doPost(javax.servlet.http.HttpServletR...
Package	com.mercury.topaz.tmc.networking.http
Root Method	ServletDispatcher.doPost()
URI	/topaz/topaz_api/api_GetLicenseInfo.asp
Count	1
Count / Sec	0.0
Exceptions	0
Timeouts	0

[-] **Latency**


Average	282.0 ms	
Contribution to Probe	3%	
% Over Threshold	41%	
Load	0.0	
Max.	282.0 ms	
Min.	282.0 ms	
Standard Deviation	4.1 ms	

[-] **CPU Time**

Average	0.0 μs	
Max.	0.0 μs	
Min.	0.0 μs	
Total	0.0 μs	

Average

Chart This Metric

 Threshold: 200.0 ms

Metric Charting Control

Threshold for selected metric

Selected metric

Threshold Icon

Selecting a Row from the Detail Table

The **Metrics Inspector** displays the metrics for the entity that you selected from the detail table.

The header of the first metric category in the **Metrics Inspector** displays the name of the selected entity. In the image above, the performance metrics for the entity `/topaz/topaz_api/api_GetLicenseInfo.asp` are displayed.

Viewing Metrics in the Metrics Inspector Area

Diagnostics groups the metrics in the Metrics Inspector into categories to make it easier to understand the performance characteristics of the selected entity.

In the example above, the metric category `/topaz/topaz_api/api_GetLicenseInfo.asp` contains more details about the server request entity that was selected from the detail table, the metric category **Latency** contains the metrics that depict the latency for the server request, and the **CPU Time** metric category contains the metrics that depict the CPU usage for the selected server request.

The metrics that are included in a metric category can be hidden or displayed by expanding or collapsing the list of metrics using the plus sign (+) and minus sign (-) next to the category name. Alternatively, you can double-click the category name to expand or collapse the list of metrics.

Reviewing a Metric's Details and Settings

When you select a metric in the Metrics Inspector, the metric's details appear in the metric control area at the bottom of the Metrics Inspector. The details that are displayed in the metric control area vary depending on the metric that you select. In the following example, the Average metric in the Latency category of the Metrics Inspector has been selected and the details for this metric are displayed in the metric control area.

The screenshot shows the Metrics Inspector window with the following details:

/topaz/topaz_api/api_GetLicenseInfo.asp	
Name	/topaz/topaz_api/api_GetLicenseInfo.asp
Signature	void doPost(javax.servlet.http.HttpServletR...
Package	com.mercury.topaz.tmc.networking.http
Root Method	ServletDispatcher.doPost()
URI	/topaz/topaz_api/api_GetLicenseInfo.asp
Count	1
Count / Sec	0.0
Exceptions	0
Timeouts	0
Latency	
Average	282.0 ms
Contribution to Probe	3%
% Over Threshold	41%
Load	0.0
Max.	282.0 ms
Min.	282.0 ms
Standard Deviation	4.1 ms
CPU Time	
Average	0.0 μ s
Max.	0.0 μ s
Min.	0.0 μ s
Total	0.0 μ s

Average

Chart This Metric

Threshold:

The details include:

- ▶ **Metric Name:** For all metrics, the name of the selected metric.

In the image above, the name of the selected metric is **Average**. Since this metric is part of the **Latency** category the metric is actually Average Latency.

- ▶ **Chart This Metric:** If the selected metric can be charted on the graph, the metric control area includes the **Chart This Metric** check box. When selected, the metric is charted on the graph for each entity selected from the detail table.

In the image above, **Chart This Metric** is selected, so the Average Latency metric is charted on the graph for each entity selected in the detail table.



- ▶ **Threshold:** If a metric displays a **Threshold** icon in the Metrics Inspector, the metric control area includes the current threshold setting.

In the image above, the threshold for Average Latency is set to **200 ms**.

Displaying a Metric in the Graph

When the **Chart This Metric** check box appears in the metric control area for a selected metric, you can select or clear the check box to indicate whether you want that metric to be charted in the graph or not. When you chart a metric for the selected entity in the Metrics Inspector, the metric is charted for each entity selected in the **Graph?** column of the detail table.

Setting Metric Thresholds



When you select a metric that has a threshold value, a **Threshold** box appears below the Metrics Inspector in the metric control area. You can adjust the threshold value by typing over the existing value. When the threshold value is changed, the **Apply Threshold Change** button appears following the **Threshold** box.

Click **Apply Threshold Change** to apply a change to a threshold value. The **Apply Threshold Change** button disappears after the requested change has been applied. If the value of the selected metric exceeds the new threshold value, a visual threshold violation indicator is immediately shown in the Metrics Inspector.

For information about charting the threshold for a metric in the graph, see “Graph Legend” on page 61.

Note: Assigning a negative threshold to a metric causes the threshold to be violated when the metric falls below the absolute value of the defined threshold.

Investigating a Threshold Alert

When the value of a metric exceeds the threshold value that you have set, or in the case of a negative threshold, dips under the value, Diagnostics displays a visual threshold violation indicator. In the **Metrics Inspector** a violation of a threshold is indicated by outlining the cell in the inspector that contains the metric that has exceeded its threshold.

In the following example, the cell containing the value of the **Average Latency** metric is outlined in red because the metric value, **296.0 ms**, exceeded the threshold of **200 ms**.

The screenshot shows the Metrics Inspector interface. It is divided into two main sections: Latency and CPU Time. The Latency section is expanded, showing a table of metrics. The 'Average' metric is highlighted with a red border, indicating a threshold violation. The CPU Time section is also expanded, showing a table of metrics. At the bottom, there is a section for 'Average' with a checkbox for 'Chart This Metric' and a 'Threshold' field set to '200.0 ms'.

Latency	
Average	296.0 ms
Contribution to Probe	5%
% Over Threshold	48%
Load	0.0
Max.	297.0 ms
Min.	297.0 ms
Standard Deviation	24.2 ms
CPU Time	
Average	0.0 μs
Max.	0.0 μs
Min.	0.0 μs
Total	0.0 μs

Average

Chart This Metric

Threshold: 200.0 ms

Configuring Metrics

You can control the metrics that appear in the Metrics Inspector by setting the appropriate Diagnostics properties. The following section contains information on configuring the Java metrics that appear in the Metrics Inspector. For information on configuring the system metrics or JMX metrics that appear in the Metrics Inspector, see the *Mercury Diagnostics Installation and Configuration Guide*.

Configuring CPU Time Java Metric

The CPU Time metric appears in the Metrics Inspector for the Transaction view, the Probes view, and the Call Profile view.

The CPU Time Java metric relies on CPU timestamping which is supported on the following platforms:

- Windows
- Solaris 8+
- AIX 5+

Use caution when enabling the collection of CPU timestamps because of the increase in Diagnostics overhead. The increased overhead is caused by an additional call for each method that is needed to collect the timestamp.

The capture of CPU Time is controlled by two properties:

- The **use.cpu.timestamps** property in the **capture.properties** file located in `<probe_install_directory>\etc` must be set to **true**
- The **cpu.timestamp.collection.method** property in `<probe_install_dir>\etc\dynamic.properties` must be set to one of the following valid values:
 - 0 - No CPU timestamping
 - 1 - CPU timestamps will be collected only for server requests
 - 2 - CPU timestamps will be collected for all methods

The default value is 1, which means that CPU times can be reported at the server request level but not at the transaction level.

If you want to turn off all CPU timestamping so that CPU time will not be gathered and reported for server requests, set the property to 0. If you want to gather CPU timestamping for all methods so that CPU time will be gathered and reported for all methods, set the property to 2.

For BPM transactions, the **cpu.timestamp.collection.method** property is ignored. CPU timestamping will always be collected for all methods for a BPM transaction.

Working with Graphs

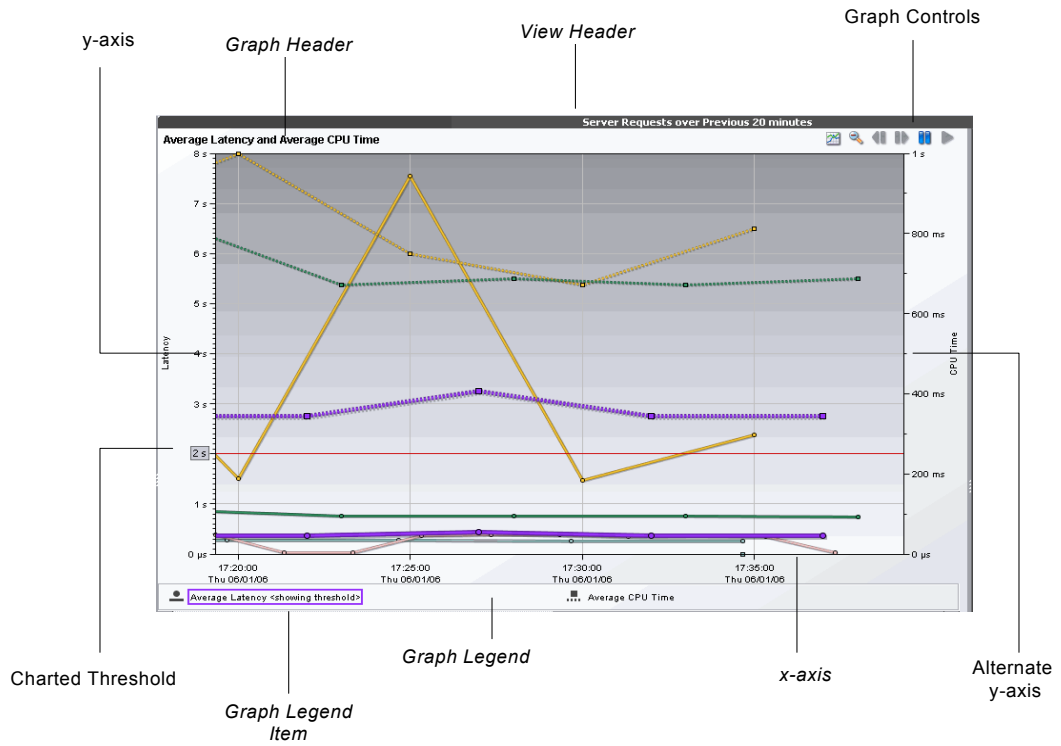
The graph in the detail views contains trend lines or stacked areas that represent the metrics that have been selected to be charted using the controls of the view filters, Metrics Inspector, and detail table. For each entity selected in the detail table a trend line or stacked area is charted for each metric selected from the Metrics Inspector.

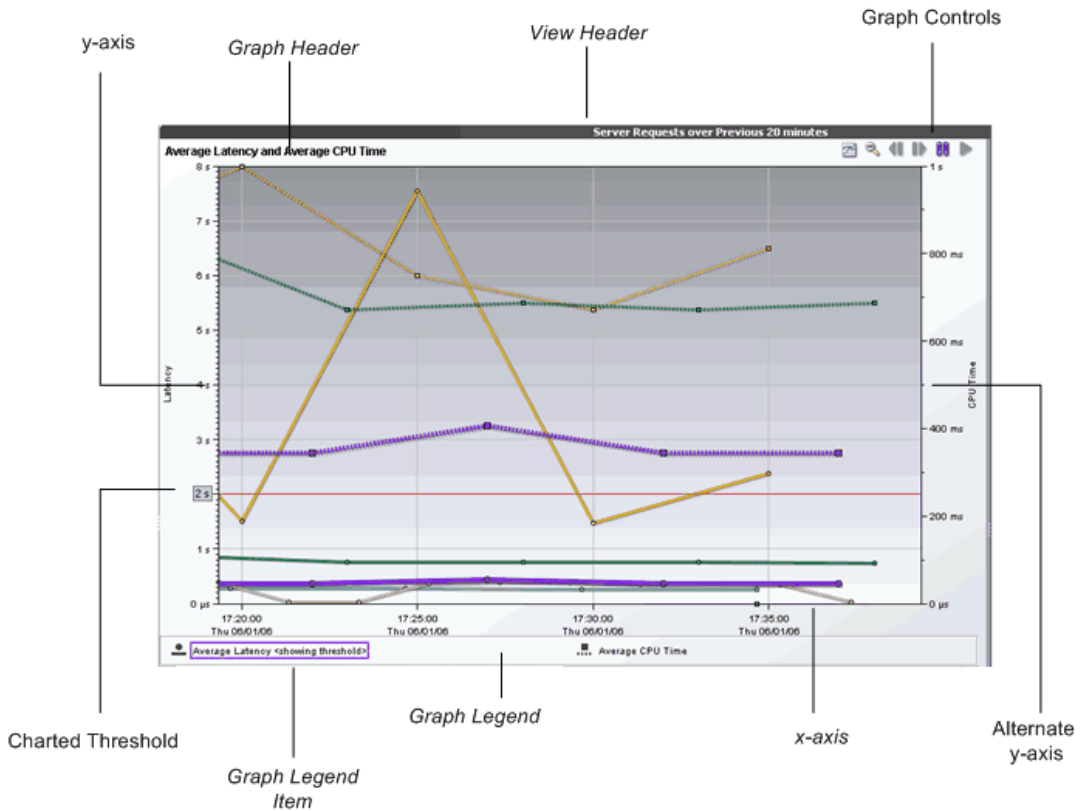
About the Detail View Graph

The following annotated screen image provides an overview of the common navigation and display controls that are used in graphs on the Diagnostics

Part II • Using Mercury Diagnostics

detail views:





View Header

The *view header* contains the name of the view and the current selection from the **View** filters. If you hold the mouse pointer over the header, a tooltip appears, displaying the name of the view and the current view filter settings.

Graph Header

The *graph header* contains the names of the metrics that you selected to chart from the **Metrics Inspector**. In the image above, the graph header indicates that the Average Latency and Average CPU Time metrics are charted.

X-Axis

The *x-axis* on the graph indicates the time range and scale used to plot the metrics on the graph. In the image above, the x-axis indicates that **Previous 20 Minutes** was selected from the **Time Range** filter, and that each interval across the axis represents a five minute time interval for a specific date and time.

Note: If you are using Diagnostics with LoadRunner or Performance Center as part of a load test, the x-axis represents the time that has elapsed since the beginning of the load test.

Y-Axis

The *y-axis* on the graph indicates the metric type, units, and scale for one or more of the related metrics that are charted on the graph. In the image above, the y-axis indicates that one or more of the metrics charted on the graph represents Latency, that Latency is charted in units of time, and that each tick mark on the axis represents 0.1 seconds.

Alternate Y-Axis

An *alternate y-axis* is displayed when one or more metrics on the graph is measured with different units or scale than those plotted on the y-axis. The alternate y-axis works just like the y-axis. In the image above, the alternate y-axis indicates that one or more of the metrics charted on the graph represents time and that each tick mark on the axis represents 100 milliseconds.

Note: Make sure to use the correct y-axis to determine the value for a particular graphed data point when more than one y-axis is displayed in the graph. In the image above, the two charted metrics each use a different y-axis.

Graph Legend

The *graph legend* displays the line patterns used to represent metrics on the graph. All of the lines on the graph that were charted with a particular pattern are associated with the metric indicated in the legend.

The graph legend lets you control the information that is charted in the graph. By right-clicking on a graph legend item, you can access a menu with the following control options:

- ▶ **Remove From Chart.** Selecting this option removes all trend lines or stacked areas for the indicated metric from the graph. This is a shortcut for removing the metric from the graph using the **Chart This Metric** checkbox in the metric control area of the Metrics Inspector.
- ▶ **Display Threshold.**
 - Selecting this option when the threshold has not already been charted in the graph causes a line to be charted in the graph for the threshold that you have established for the metric. If the threshold for another metric had been charted, the threshold for that metric is removed from the graph.
 - Selecting this option when the threshold is already charted causes the threshold line to be removed from the graph.

Graph Controls

The *graph controls* are a set of buttons that allow you to control the way that the metrics are charted on the graph. Each of these buttons and the method for zooming in on the charted metrics are described in “Pausing and Panning the Graph” on page 71.

Displaying Entities and Metrics in the Graph

You select entities to view in the graph in one of the following ways:

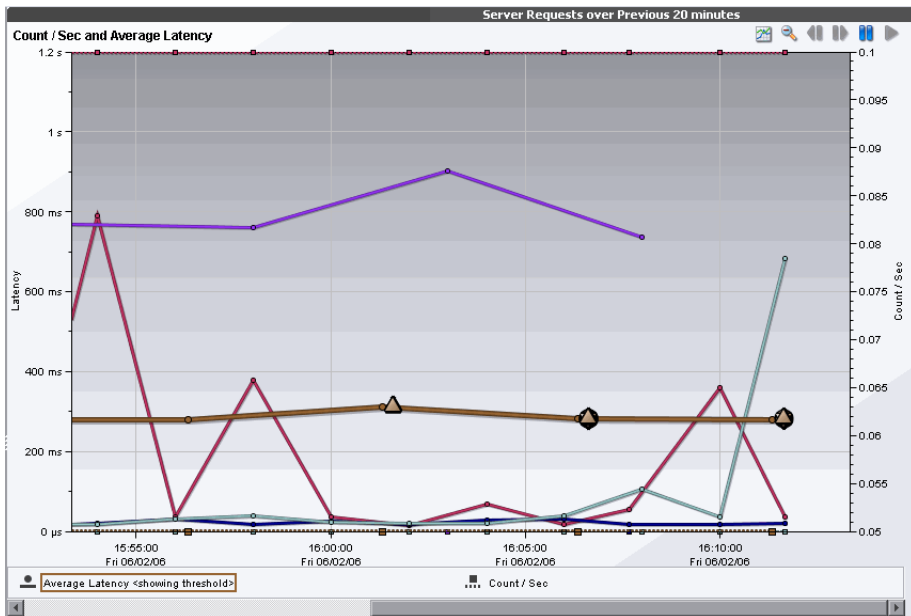
- ▶ Using the **Chart** column in the detail table. See “Adding and Removing Entities from the Graph” on page 75.
- ▶ From the **Graph** filter in the View Filters area. See “Filtering Charted Entities Based on Relative Metric Values” on page 45.

Viewing Multiple Metrics on a Graph

When more than one metric is charted in the graph, Diagnostics helps you identify each metric by charting each with a different pattern. The patterns that are used for each metric are recorded in the **Graph Legend**.

Each entity is color coded, and can be identified in the graph by a line charted in the same color that is indicated for the entity in the Color column of the details table. Each metric selected from the Metrics Inspector is charted for each entity selected from the detail table, so long as a valid metric value exists for the entity.

In the following image, both the Average Latency and Count per Second metrics are charted for the four entities shown.



Viewing the Data Behind a Charted Metric on the Graph

From the graph, there are several ways to determine the entities and metrics that are behind the trend lines and stacked areas that you can see in the graphs.

Identifying the Entity in Detail Table for a Charted Metric

From the graph, you can associate a trend line or stacked area with the entity in the detail table that it represents. You do this in one of the following ways:

- ▶ Hold the mouse pointer over a data point in the graph. A tooltip appears, displaying the details for the entity and metric represented by that point in the trend line. This is discussed in more detail in the next section, “Identifying the Metric Details for a Charted Point”.
- ▶ Click anywhere on a trend line or stacked area in the graph. The trend line or stacked area is highlighted in the graph, and the entity is highlighted in the detail table. At the same time, the Metrics Inspector is updated to display all of the metrics for the selected entity.

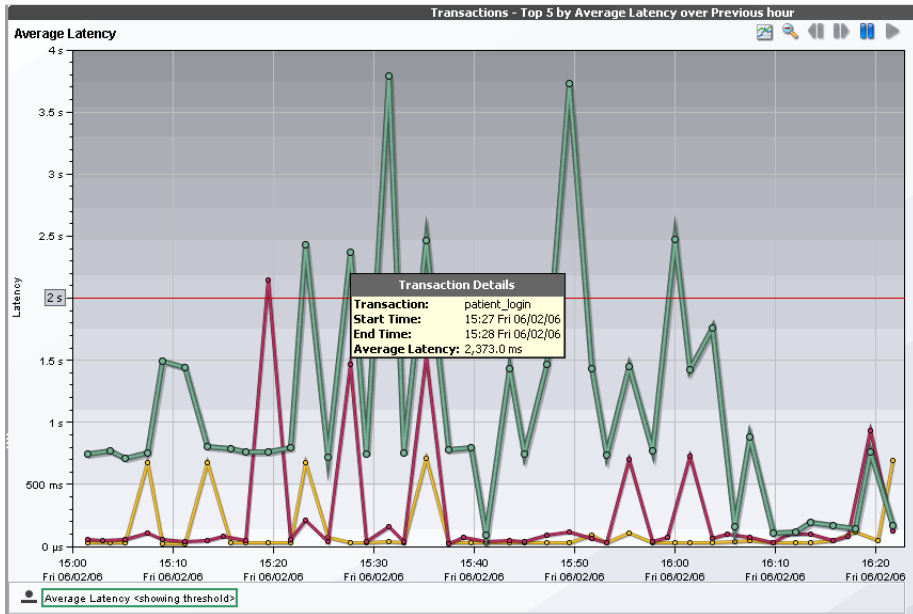
When you select a trend line in the graph, all the trend lines charted for metrics that are related to the entity are also highlighted.

When you select a stacked area in the graph, it is displayed as shaded rather than solid.

Identifying the Metric Details for a Charted Point

The data points that were used to chart the trend lines on the graph are highlighted with a point marker on the trend line. On the stacked area charts, you can recognize these data points when the slope of the boundary of the stacked area changes.

When you hold the mouse pointer over a data point, a tooltip appears, displaying details for that data point, as shown in the following example:



The information in the tooltip varies depending on the detail view with which you are working. This information includes:

Title Bar: The title bar for the tooltip identifies the type of entity to which the data point belongs. In the example above, the title bar is **Transaction Details**, which indicates that the tooltip contains the details for a metric displayed in the Transactions view.

Entity Name: The entity name identifies the type and the name of the displayed entity. In the example above, the type of entity is **Transaction**, and the name of the entity is **patient_login**.

Aggregation Period Start Time: The **Start Time** entry in the tooltip displays the starting time of the aggregation of the metrics for the data point.

Aggregation Period End Time: The **End Time** entry in the tooltip displays the time that metrics for the data point were last gathered.

Metric Name: The metric name identifies the metric and the value of the metric charted. In the example above, the metric is **Average Latency** and its value is **2,373.0 ms**.

Controlling the Graphed Metrics

Diagnostics enables you to control the way that the performance metrics are charted in the graphs. You can zoom in and out on the charts, pause the graph so that you can get a closer look at the metrics that are currently charted, and keep the y-axis scaling fixed so that the charted metrics are easier to comprehend.

Zooming on the Graph

You can zoom in on the metrics charted in the graph in order to get a closer look at the performance portrayed. Diagnostics has two different types of zooming: *magnification zooming* and *resolution zooming*. It automatically determines which type of zooming to use based upon the time range that you selected in the view filters and the *zoom range* that you choose.

More information about the types of zooming is provided in the following sections.

Understanding Magnification Zooming

Magnification zooming provides you with a magnified view of the data points that are charted in the current graph. Diagnostics does not retrieve any additional information when portraying a magnified view of the performance metrics.

There are two situations when Diagnostics performs a magnification zoom instead of a resolution zoom. The first is when the graph has been plotted using data that had been aggregated into the highest resolution data points. The second situation is when you select a zoom range that spans a time period that requires too much data to be retrieved for an efficient resolution zoom.

When you select a time range of **Previous 5 Minutes**, the resolution of the data used to display the metrics is based on 5-second aggregations. When these 5-second data points are plotted on the graph, they are initially charted so that all the data for the previous 5 minutes is visible in the detail view graph. If the data points for the metric on the graph are too close to each other, you can zoom into an area of interest to get a closer look. In this situation, where the graph has been built with data that has been aggregated at a very high resolution, Diagnostics zooms in to give you a magnified view of the metrics. No additional data is retrieved to chart the magnified view of the data.

When you select a time range that is higher than the **Previous 5 Minutes**, the resolution of the data that is used to plot the metrics is based on aggregations of longer periods of time. If the data points for the metric on the graph are too close together, you can zoom in to get a closer look. If you select a smaller zoom range, Diagnostics zooms in using a resolution zoom. However, if the zoom range is larger and requires Diagnostics to retrieve more higher-resolution data than can be done efficiently, Diagnostics zooms in using the magnification zoom. No additional data is retrieved to chart the magnified view of the data.

Understanding Resolution Zooming

Resolution zooming gives you a higher resolution view of the performance metrics. Diagnostics retrieves additional data to chart metrics at a higher resolution.

When the selected time range is larger than **Previous 5 Minutes**, Diagnostics retrieves data that has been aggregated into lower resolution data points. That is, the points charted on the graph represent larger time periods. For example, when you select a time range of **Previous Hour**, the data is aggregated into one-minute aggregations. These points are initially charted so that all of the data for the previous hour is visible in the graph. If the data points are too close together to analyze the performance for a particular area on the graph, you can zoom into that area. In this situation, where the graph displays data that has been aggregated at a lower resolution, Diagnostics retrieves data that has been aggregated at a higher resolution to zoom into the metrics charted.

Note: If the selected zoom range is large and requires Diagnostics to retrieve more high-resolution data than can be done efficiently, the zoom is performed using the magnification zoom instead.

Zooming In on Charted Metrics

You can zoom in to see more detail for the metrics in a particular time period on the graph.

To zoom in on the graph:

- 1 Click the graph at the beginning of the time range on which you want to zoom in, and drag the pointer to select the zoom range.

As soon as you move your mouse after you have clicked, the pointer changes to a zoom icon that indicates that you are zooming in on the metrics, and the background of the selected zoom range darkens.

The zoom icon that you see indicates whether Diagnostics is using a magnification zoom or a resolution zoom.



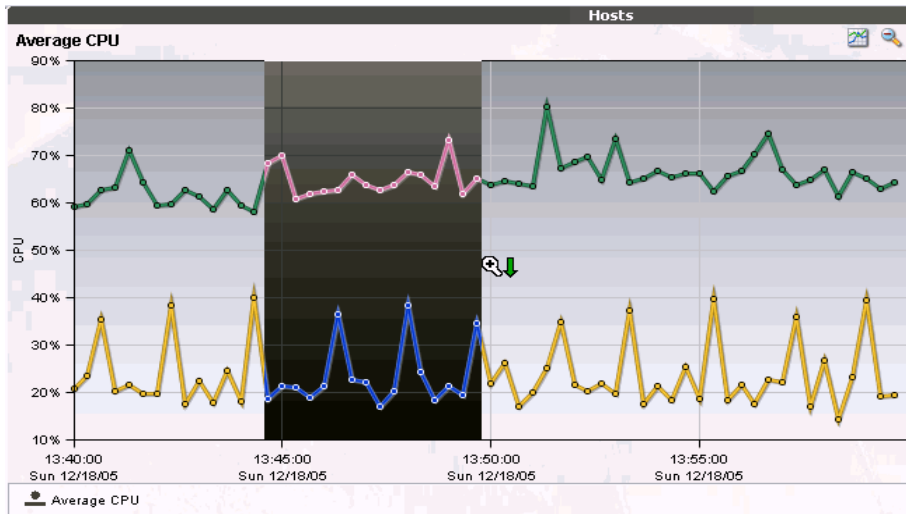
- ▶ The **Magnification Zoom** icon is displayed when Diagnostics is zooming by charting a closer look at the metrics currently charted in the graph. For a description of magnification zooming, see “Understanding Magnification Zooming” on page 66.



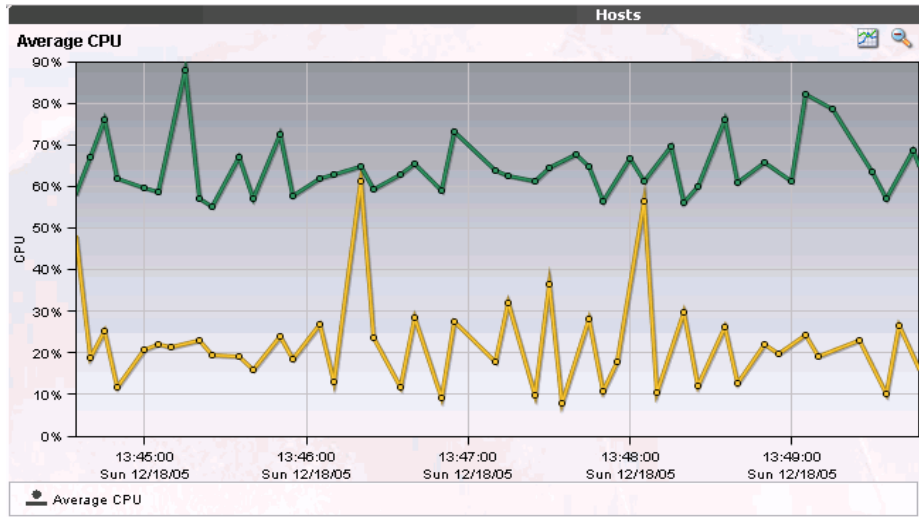
- ▶ The **Resolution Zoom** icon is displayed when Diagnostics is zooming by charting higher resolution data for the selected zoom range. For a description of resolution zooming, see “Understanding Resolution Zooming” on page 67.

The **Resolution Zoom** icon switches to the **Magnification Zoom** icon when Diagnostics determines that the zoom range that you specified requires too much higher-resolution data to be retrieved. Select a smaller zoom range if you would rather zoom using the resolution zoom.

The following example shows a zoom range request where Diagnostics is performing a resolution zoom:



- 2 Release the mouse button to indicate that you have marked the desired zoom range. Diagnostics zooms in on the selected range and displays the metrics. The following image shows the graph after zooming in:



Note the following on the graph:

- Diagnostics displays the metrics for the time range that you requested when zooming in. The requested zoom range fills the entire visible graph. Use the x-axis labels on the graph to determine the range and resolution of the metrics. Both the x-axis and y-axis are rescaled to correspond to the data in the zoomed range. In the example above, the zoom range is approximately **13:45** through **13:50**.

- Diagnostics retrieves the higher resolution metrics for at least the time range that you indicated. It may also retrieve metrics for a larger time range. Diagnostics makes these additional metrics available to you by adding a scroll bar under the graph. You may use the scroll bar to view the data outside of the zoom range that you requested. In the example above, the scroll bar that was added allows you to scroll to view data prior to **13:45** and after **13:50**.

Note: When the scroll bar is visible under the graph, you can also right-click the graph and drag the pointer back and forth to scroll across graph.

- If Diagnostics zoomed using a resolution zoom, the selection in the **Time Range** view filter is set to **Custom** to indicate that you have manually set the time range charted in the graph by specifying a zoom range.

Zooming Out of Charted Metrics

You may want to zoom out after you have zoomed in on an area of the graph.



To zoom out of a graph that you previously zoomed in on, click the **Zoom Out** button.

- The first time that you click **Zoom Out**, Diagnostics adjusts the zoom range so that all of the metrics that were retrieved when the higher resolution data was retrieved, are charted in the graph. This means that metrics for a larger period of time are visible on the graph without scrolling. For this reason, the scroll bar is removed from the view.
- Subsequent clicks of **Zoom Out** cause Diagnostics to zoom out to lower resolution data. The time period for the original zoom range that was displayed when you started to zoom out continues to be included in the wider, lower resolution data charted on the graph.
- Clicking **Zoom Out** multiple times causes Diagnostics to zoom out to progressively slower resolutions and wider periods of time. When Diagnostics has zoomed back to the time ranges listed in the Time Range filter, it adjusts the selection in the Time Range filter to correspond to the time range charted in the graph as a result of zooming out.

Pausing and Panning the Graph

The **Pause** and **Pan** buttons at the top of the graph allow you to stop the updating of metrics so that you can analyze those that are currently charted before they are replaced by more current metrics. To stop the charting, click **Pause**. When you have stopped the charting of new metrics, click **Pan Left** and **Pan Right** to cause Diagnostics to display the charted metrics on either side of the data currently displayed in the graph. When you have completed your analysis of the charted metrics, click **Resume** to continue charting live metrics. Each of these buttons is described below:



- ▶ Clicking **Pause** causes Diagnostics to stop displaying new data in the graph. The graph remains unchanged when this button is selected. In addition, the **Pan Left**, **Pan Right**, and **Resume** buttons are activated, and the **Autoscale Y Axis** is selected so that new data coming from the server will not impact the way that the data is displayed in the graph while the pause is in effect. The scroll bar may also appear if there was additional data from the past available to be displayed that was part of the data already charted in the graph.



- ▶ Clicking **Pan Left** causes the graph to smoothly scroll from the time period currently displayed to the time period with the same duration that immediately preceded the one displayed. If the scroll bar was displayed, it disappears because all of the data that is retrieved by the pan request is charted in the visible graph. In addition, the Graph view filter is reset from the settings that you selected to avoid distracting changes to the colors of the charted lines.



- ▶ Clicking **Pan Right** works just like **Pan Left** except that the graph scrolls to a later time period than the one that is currently displayed.



- ▶ Clicking **Resume** causes Diagnostics to activate the graph and begin charting the current performance metrics as they become available. The Time Range filter is reset to show the same granularity that you were viewing when you first clicked **Pause**.

Working with the Detail Table

The detail table on the detail views lists the entities for the particular view for which metrics have been gathered. For example, on the Probes view the entities are probes and on the Load view, the entities are layers. From the detail table you can select the entities that are to be charted on the graph. The detail table is highly customizable allowing you to control the columns that appear in the table and the sort order for the table.

About the Detail Table

The following annotated screen image provides an overview of the common navigation and display controls that are included the detail table:

Status	Color	Chart?	Server Request	Probe	Latency % Over Thresh...	Latency	Count/S	Info
		<input checked="" type="checkbox"/>	/sampleportal/sample.portal	PortalServer	7%	3.2 s	0.017	
		<input checked="" type="checkbox"/>	/sampleportal/sample.portal	PortalServer	-95%	3.1 s	0.017	
		<input checked="" type="checkbox"/>	/topaz/topaz_api/api_GetLicenseInfo.asp	ALMAGRO	-99%	625.0 ms	0.003	
		<input checked="" type="checkbox"/>	/patent/login.do	AMKILAB03	-99%	557.5 ms	0.013	
		<input checked="" type="checkbox"/>	/patent/editprofile.do	AMKILAB03	-99%	358.5 ms	0.007	
		<input type="checkbox"/>	TraderService:buy	TALCF	-100%	104.0 ms	0.003	
		<input type="checkbox"/>	/topaz/topaz_api/api_invoke.asp	ALMAGRO	-100%	84.2 ms	0.107	
		<input type="checkbox"/>	ProfileFactory.getProfile()	PortalServer	-100%	81.9 ms	0.033	
		<input type="checkbox"/>	/topaz/topaz_api/api_bytearray_invoke	ALMAGRO	-100%	70.0 ms	0.007	
		<input type="checkbox"/>	SettingsImpl.setGlobalVariable()	ALMAGRO	-100%	57.3 ms	0.010	
		<input type="checkbox"/>	/JavaTrader.WebClient/JavaTrader.WebClient.aspx	JavaTrader.Web...	-100%	56.5 ms	0.007	

Detail Table Header

The *detail table header* labels the columns that are displayed in the table. From the detail table header, you control which columns are displayed in the table and the order in which they appear. You also control the sort order of the entities displayed in the table.

For more information about customizing the columns, see “Customizing the Table Columns” on page 32.

For more information about sorting the rows in the detail table, see “Sorting Rows in the Table” on page 34.

Detail Table Entity

The rows in the detail table contain the *detail table entities* whose performance characteristics are presented in the view. The detail table entity shows you the details of the performance of your application and enables to drill down to even more performance details.

Understanding the Columns in the Detail Table

Several of the columns that appear in the detail table are the same for most of the predefined detail views. Those common columns are defined in this section. Other columns may be specific to a particular view and contain metrics and configuration information that apply only to the predefined view in which they are presented. These columns are defined in Chapter 5, “Using the Standard Detail Views.”

The columns common to most of the predefined detail views are:



Status. The **Status** column contains the status indicator that lets you see, at a glance, how the listed entity is performing relative to the thresholds that you set for the metrics of the entity or of its child entities.

The color of the status indicator tells you the status of the entity. If you hold the mouse pointer over the status indicator in a row of the table, a tooltip displays a description of the current status:

- ▶ Green: Good - The component is performing within defined thresholds.
- ▶ Yellow: Warning - The component is occasionally but not consistently exceeding defined thresholds.
- ▶ Red: Critical - The component is consistently exceeding defined thresholds.
- ▶ Grey: No status information available. Either no data has been received for the entity metrics or no threshold has been set for the metrics.

For information about setting thresholds, see “Setting Metric Thresholds” on page 53.

Color. This column contains a colored block. When the **Chart** check box for an entity in the detail table is selected, trend lines or stacked areas for the entity are included in the graph. The color of the block in the Color column is the color that Diagnostics uses to chart the metrics for the entity. When Diagnostics has charted more than one metric for an entity, the trend lines or stacked areas for each metric are the same color in the graph.

Chart. This column contains a check box that indicates whether Diagnostics should chart the entity's metrics in the graph. When the check box is selected, Diagnostics charts a trend line or stacked area for each metric that you have indicated in the Metrics Inspector. For more information about charting the metrics for an entity in the detail table, see “Displaying a Metric in the Graph” on page 52.

Info. This column contains icons that remind you that you have entered alert rules or comments for the entity depicted in the row. One or more of the following icons can appear in this column:



► **Active Alert Rule.** This icon indicates that you have created an alert notification rule for the selected entity and the rule is active.



► **Disabled Alert Rule.** This icon indicates that you have created an alert notification rule for the selected entity but the rule is currently disabled.



► **Custom Comments.** This icon indicates that you entered comments for the selected entity.

For more information about Alert Notification Rules, see Chapter 4, “Alert Notification.” For more information on entering comments for an entity, see “Maintaining Entity Comments” on page 79.

Customizing the Detail Table

The detail table headers contain controls that allow you to specify which columns Diagnostics is to display in the table, and the order in which they are to be displayed. You can also select which columns Diagnostics is to use when sorting the rows displayed in the table. For more information on how to use the controls in the table headers, see “Using Table Header Controls” on page 32.

Working with the Detail Table and the Graph

From the detail table, you can control the entities for which Diagnostics charts metrics in the graph, and you can find more information about the entities and metrics that are listed in rows of the table.

Identifying Entity Metrics in the Detail Table

You can determine which trend lines or stacked areas in the graph are associated with a particular entity in the detail table by clicking the row in the table to select the entity. Diagnostics highlights the selected entity and its associated metrics in the graph.

Adding and Removing Entities from the Graph

The **Chart** column on the detail table lets you control which entities' metrics Diagnostics is to chart in the graph.

- ▶ To chart the metrics for an entity in the graph, select the **Chart** check box for that entity.
- ▶ To remove the charted metrics for an entity from the graph, clear the **Chart** check box for that entity.
- ▶ To remove the charted metrics for all of the entities selected in the **Chart** column, right-click the **Chart** column header and select **Remove All Series from Graph**. This causes all of the selected check boxes to be cleared and an empty graph to be displayed. You may then repopulate the graph by selecting the **Chart** check box for the entities whose metrics you would like to have graphed. You can also use the **Graph** filter in the **View Filters** to let Diagnostics pick significant sets of entities for which it will chart metrics.

When you select the **Chart** check box for any entities in the detail table, Diagnostics changes the selection for the **Graph** filter in the **View Filters** to **Custom** to indicate that you have selected the entities manually. For more information about the **Graph** filter, see “Filtering Charted Entities Based on Relative Metric Values” on page 45.

Searching for an Entity in the Detail Table

You can search for a specific entity in the detail table.

To find an entity in the detail table:

- 1 Press CTRL + F. Diagnostics displays the **Search For:** popup above the detail table column headers as shown in the following image:

Status	Color	Chart?	Server Request	Probe	Latency % Over Thresh...	Latency	Count/S	Info
x	■	<input checked="" type="checkbox"/>	/sampleportal/sample.portal	PortalServer	3%	3.1 s	0.017	
●	■	<input checked="" type="checkbox"/>	/topaz/topaz_api/api_GetLicenseInfo.asp	ALMAGRO	-94%	3.7 s	0.003	
●	■	<input checked="" type="checkbox"/>	/sampleportal/sample.portal	PortalServer	-95%	3.2 s	0.013	
●	■	<input checked="" type="checkbox"/>	a.execute()	ALMAGRO	-97%	2.1 s	0.003	
●	■	<input checked="" type="checkbox"/>	/patient/login.do	AMKILAB03	-100%	202.0 ms	0.013	
●	■	<input type="checkbox"/>	TraderService:buy	T-LC7	-100%	103.0 ms	0.007	
●	■	<input type="checkbox"/>	/topaz/topaz_api/api_invoke.asp	ALMAGRO	-100%	94.2 ms	0.100	

- 2 Type in all or part of the name of the entity that you would like to locate. As you type each character, Diagnostics begins the search.
- 3 If a match is found, Diagnostics selects the first row fitting the search criteria from the detail table.
- 4 If no match is found, Diagnostics changes the color of the text in the **Search For:** popup to red, and adds the **(no match)** string, following your search criteria.
- 5 To exit the search, press ESC.

Viewing Data about an Entity in the Detail Table

The columns in the detail table contain information about the performance of the listed entities. However, this information is just a high-level summary of the information available to help you analyze the performance of your applications. Diagnostics provides many ways for you to dig deeper into the metrics associated with the entities listed on the detail table.

Viewing Tooltips for the Detail Table Cells

Many of the cells in the rows of the detail table display tooltips when you hold the mouse pointer over the cell.

- The tooltip for the **Status** column is discussed in “Understanding the Columns in the Detail Table” on page 73.
- The tooltip for the entity column contains configuration information for the listed entities. The content of the tooltip varies depending on the entities that are listed in the table. The following example shows the tooltip for the entities in the Probes view:

Status	Color	Chart?	Probe	VM Heap	Latency
		<input checked="" type="checkbox"/>	cover	178.6 MB	
		<input checked="" type="checkbox"/>	PortalServer	155.2 MB	251.7 ms
		<input checked="" type="checkbox"/>	x230	139.1 MB	
		<input checked="" type="checkbox"/>	MSPetshop.NET	8.1 MB	
		<input checked="" type="checkbox"/>	StoreCSV5.NET	8.1 MB	
		<input type="checkbox"/>	t-lc8_MedRec	6.9 MB	3.0 ms
		<input type="checkbox"/>	AMKILAB03	6.4 MB	43.2 ms
		<input type="checkbox"/>	JavaTrader.WebClient.N	6.9 MB	58.0 ms
		<input type="checkbox"/>	T-LC7	27.1 MB	52.5 ms

Probe Details

Probe: PortalServer
Probe Group: Default
Host: x231.lab.performant.com
Probe Type: Java

- The tooltip for the columns that contain metric values mimics the value in the selected cell. This is useful if you have resized the column and want to know the value of the listed metric without changing the width of the column again.

Viewing Metrics for a Selected Entity in the Metric Inspector

When you select an entity in the detail table, Diagnostics displays the metrics for the selected entity in the Metrics Inspector. For more information about the Metrics Inspector, see “Working with the Metrics Inspector” on page 48.

Drilling Down To Entities in the Detail Table

Many of the Diagnostics views allow you to drill down to the entities listed in the detail table to analyze the performance of the lower-level components of the entities. As you drill down from one view to the next, the breadcrumb at the top of the view keeps track of the path that you have taken so that you can navigate back to where you started, or to an intermediate view that you passed along the way.

Diagnostics provides two ways to drill down from the entities in the detail table:

Note: Not all views have drilldown navigation options.

- ▶ Right-click an entity in the table. Diagnostics displays a menu with the available drilldown navigation options. The following example shows the menu that is displayed when you right-click an entity in the Probes view:

Status	Color	Chart?	Probe	VM Heap	Latency	CPU Time	Cour
●	Green	<input checked="" type="checkbox"/>	cover	175.6 MB			
●	Red	<input checked="" type="checkbox"/>	PortalServer	167.0 MB	235.4 ms	210.3 ms	
●	Blue	<input checked="" type="checkbox"/>	x230	26.8 MB			
●	Purple	<input checked="" type="checkbox"/>	StoreCSV5.NET	73.2 MB			
●	Yellow	<input checked="" type="checkbox"/>	MSPetshop.NET	73.2 MB			
●		<input type="checkbox"/>	t-lc8_MedRec	36.9 MB	3.0 ms	0.0 μs	
●		<input type="checkbox"/>	AMKILAB03	36.5 MB	20.8 ms		
●		<input type="checkbox"/>	JavaTrader.We	34.9 MB	78.2 ms		
●		<input type="checkbox"/>	T-LC7	27.3 MB	70.3 ms	0.0 μs	
●		<input type="checkbox"/>	t-lc8_caller	24.2 MB			
●		<input type="checkbox"/>	t-lc8_callee	22.5 MB			
●		<input type="checkbox"/>	ALMAGRO		10.8 ms	0.0 μs	

View Server Requests							
View Probe Summary							
View Portal Components							
View Load							
Open Mercury Diagnostics Profiler							
Create Alert Rule...							
Create Comments...							

From this menu, you can:

- ▶ Drill down to four different views.
- ▶ Drill down to the Mercury Diagnostics Profiler to view more detailed metrics for the selected probe.

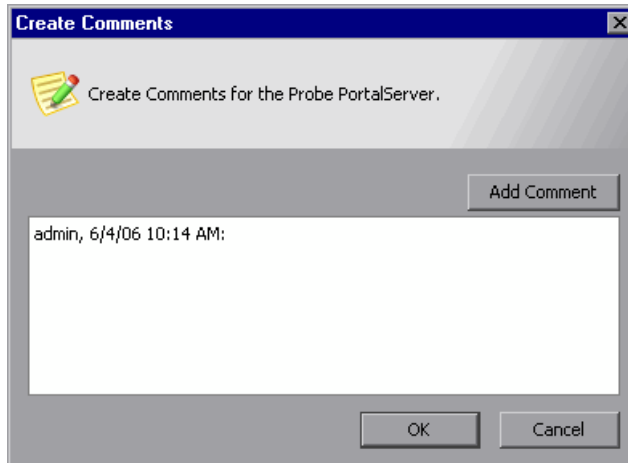
- ▶ Create or Maintain the Alerting Notification Rules for the probe's performance metrics. For instructions to create and maintain Alert Notifications, see Chapter 4, "Alert Notification."
- ▶ Create a comment that can serve as a reminder or note concerning the configuration, status, or performance of the probe. For instructions to create and maintain the notes for an entity, see "Maintaining Entity Comments" on page 79.
- ▶ Double-click a row in the table to drill down directly into the default lower level detail view. This is a shortcut that allows you to drill down without stopping at the right-click menu. The default drilldown is bolded in the entity popup menu. In the example above, the default drilldown would be **View Server Requests** since this is the bolded item listed on the menu.

Maintaining Entity Comments

You can create and maintain notes for an entity listed in the detail table by right-clicking on the entity and selecting one of the comment menu options from the popup menu.

Creating a Comment

If a comment has not been previously created for the selected entity, select **Create Comments** to add a comment. Diagnostics displays the Create Comments dialog:



Diagnostics creates a comment header by inserting your user name and the date and time stamp. Type your comment. If you would like to enter a second comment, click **Add Comment**. Diagnostics starts a new paragraph with a new comment header. Click **OK** to close the Create Comments dialog.

Editing a Comment

If a comment has previously been created for the selected entity, select **Edit Comments** to update an existing comment or add a new comment. Diagnostics displays the Edit Comments dialog that appears as shown in the previous example.

Deleting a Comment

If a comment has been previously created for the selected entity, select **Delete Comments** to delete the comment. Diagnostics displays a confirmation dialog. Click **Yes** to delete all comments for the selected entity.

4

Alert Notification

This chapter describes:	On page:
About Alert Notification	81
Configuring Alert Notification	82
Working with Alert Notification Rules	85
Reviewing Alert Notification Events	93

About Alert Notification

Diagnostics enables you to set threshold values for metrics so that it can issue an alert when the metric values have exceeded the threshold. Diagnostics indicates that an alert condition has occurred by changing the color of the status indicator displayed in the Status view and in the detail table of the detail views to red. Alerts are also indicated by adding a red border to the cells of the detail table and Metrics Inspector containing the metric values that have exceeded their thresholds. For more information on setting thresholds, see “Setting Metric Thresholds” on page 53.

In addition to the alert indicators that Diagnostics displays in the views, you can set up rules that instruct Diagnostics to send alert notifications via e-mail or SNMP. Users can be notified when alert conditions are encountered for probes, probe Groups, hosts, and server requests. Diagnostics can send alert notifications when the status of the entity changes to red and also when the entity becomes unavailable.

Note: Alert notifications for a given entity are sent once at the time that the entity enters critical status. They are not sent again until the status reverts to normal and then once more becomes critical. In other words, an entity that remains in critical status will not continuously cause the generation of alert notifications.

Configuring Alert Notification

Before you can set up alert notification rules, you must first set the values of the properties that enable alerting notification. You configure alert notifications from the Diagnostics Server Alert Properties page for each Diagnostics Server in Mediator mode. The Alert Properties page is shown in the following example:

MERCURY			
Alert Properties			
Name	Value	Description	Default Value
Alerting Enabled	<input type="text" value="true"/>	This switches alerting on and off entirely for this server.	true
SNMP Server	<input type="text"/>	The IP address or fully-specified hostname to which the Diagnostics Server will send SNMP traps.	
SNMP Port	<input type="text" value="162"/>	The port to which the Diagnostics Server will send SNMP traps.	162
SNMP Community Key	<input type="text" value="public"/>	The SNMP community key which the Diagnostics Server will use when sending SNMP traps.	public
SMTP Server	<input type="text"/>	The IP address or fully-specified hostname to which the Diagnostics Server will send email via SMTP.	
SMTP Port	<input type="text" value="25"/>	The port to which the Diagnostics Server will send email via SMTP.	25
SMTP From Address	<input type="text"/>	Email address from which alerts will be sent	
SMTP Default Email Addresses	<input type="text"/>	Default email addresses for SMTP alerts (comma-separated list)	

Accessing the Alert Properties Page

You should always update Diagnostics alert notification properties from the Alert Properties page for each Diagnostics Server in Mediator mode to ensure that the property values are set correctly.

To access the Alert Properties page from the Diagnostics views:

- 1 From the menu on the top, right hand side of the view, click **Maintenance**. Diagnostics displays the Diagnostics Server Maintenance menu in a new browser window.
- 2 Select **configuration > Alert Properties**. Diagnostics displays the Alert Properties page.

Note: Changing the alert properties in this manner for a given Diagnostics Server will only impact the alert notifications for entities that are using this same Diagnostics Server as a Diagnostics Server in Mediator mode.

To access the Alert Properties page from your browser:

- 1 Open the Diagnostics Server in Mediator mode administration page by navigating to the URL http://<diagnostics_server_host>:2006 in your browser, or by selecting **Start > Programs > Mercury Diagnostics Server > Administration** on the host for the Diagnostics Server in Mediator mode. The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.
- 2 Access the Diagnostics Server Maintenance menu by selecting **Configure Diagnostics**
- 3 Select **configuration -> Alert Properties**. Diagnostics displays the Alert Properties page.

Enabling and Disabling Alert Notifications

Diagnostics enables alert notification by default. This means that if you have created alert notification rules and an alert triggering event has occurred, Diagnostics sends the notifications that you specified in the rules.

When alert notification is disabled, you will still see the alert conditions displayed in the Diagnostics views using the color of the status indicator and the border of the detail table and Metrics Inspector cells that contain the metric that exceeded its threshold. Only the external alert notifications to SNMP or e-mail are disabled.

To disable alert notifications:

- 1** Access the Alert Properties page using one of the methods specified in “Accessing the Alert Properties Page” on page 82.
- 2** Set the value of the **Alerting Enabled** property to false.
- 3** Click **Submit** to disable alert notifications. It can take up to 30 seconds for the new property value to take effect.

To enable alert notifications:

- 1** Access the Alert Properties page using one of the methods specified in “Accessing the Alert Properties Page” on page 82.
- 2** Set the value of the **Alerting Enabled** property to true.
- 3** Click **Submit** to enable alert notifications. It can take up to 30 seconds for the new property values to take effect.

Configuring SNMP Alert Notifications

SNMP alert notifications are disabled until you configure the SNMP alert notification properties.

Note: Note that all SNMP alert notifications are sent as SNMP v2c traps. To help properly parse these traps on the receiving end, please refer to the MercuryStatusAlerts.mib file included with the server installation.

To configure SNMP alert notifications:

- 1** Access the Alert Properties page using one of the methods specified in “Accessing the Alert Properties Page” on page 82.
- 2** Set the value of the **SNMP Server**, **SNMP Port**, and **SNMP Community Key** properties to the appropriate values based on the instructions on the Alert Properties page.
- 3** Click **Submit** to enable SNMP alert notifications. It can take up to 30 seconds for the new property value to take effect.

Configuring SMTP E-mail Alert Notifications

SMTP alert notifications are disabled until you configure the SNMP alert notification properties.

To configure SMTP E-mail alert notifications:

- 1** Access the Alert Properties page using one of the methods specified in “Accessing the Alert Properties Page” on page 82.
- 2** Set the value of the **SMTP Server**, **SMTP Port**, **SMTP From Address**, and **SMTP Default E-mail Addresses** properties to the appropriate values based on the instructions on the Alert Properties page.
- 3** Click **Submit** to enable SMTP alert notifications. It can take up to 30 seconds for the new property value to take effect.

Turning Alert Notification Messages In the Views On and Off



The **Turn On/Off Active Alert Notifications** button in the Diagnostics toolbar lets you control whether a message is to be displayed in the Diagnostics Message box each time an alert notification is sent. Note that Active Alert Notifications are off by default.

Working with Alert Notification Rules

To receive an alert notification by SNMP or by e-mail, you must define an *Alert Notification Rule* for a given entity. Alert notification rules can be created for probes, probe groups, hosts, and server requests. When an alert notification rule has been created for one of these Diagnostics entities, the rule is used to determine whether a notification is to be sent and how the notification is to be delivered.

Note that nature of status in Diagnostics does not dictate that you must set the alert on the same entity for which you set a threshold. Since status propagates from lower-level entities to higher-level entities, a single high-level alert notification rule can be used to notify you of a variety of lower-level issues.

As an example, a "red-status" alert notification rule on a probe group would be triggered any time a probe or server request in that probe group enters critical status, because the status of the probe group will also become critical at that time. Similarly, an alert on a probe can certainly be triggered by probe metrics, but also will be triggered if any metric on any of the probe's server requests crosses a threshold. In these cases, the alert notification will provide details on the metric that triggered the alert, even if that metric is actually on a lower-level entity.

Maintaining Alert Notification Rules

You can create, edit, and delete the alert notification rules using the right-click pop-up menu on the views that list probes, probe groups, hosts, and server requests. Only one alert notification rule can be created for each entity.

Note: You cannot create an alert notification rule until you have configured either the SNMP properties or the e-mail properties as described in "Configuring Alert Notification" on page 82.







To create an alert notification rule:

- 1 Right-click on a Probe, Probe Group, Host, or Server Request listed in a Status view or the detail table of a detail view. Diagnostics displays the pop-up menu for the selected entity.

When an alert notification rule has not yet been created for the selected entity, the pop-up menu appears as shown in the following example from the Probes view:

Status	Color	Chart?	Probe	VM Heap	Latency
●	■	<input checked="" type="checkbox"/>	BAC_61	442.0 MB	8.3 ms
●	■	<input checked="" type="checkbox"/>	Oslo2	207.6 MB	632.2 μs
●	■	<input checked="" type="checkbox"/>	WL81_Portal_Sample	206.4 MB	49.4 ms
●	■	<input checked="" type="checkbox"/>	Dublin2		645.5 μs
●	■	<input checked="" type="checkbox"/>	Joyride2		612.2 μs
●		<input type="checkbox"/>	Budapest1		628.2 μs
●		<input type="checkbox"/>	Seine2		594.0 μs
●		<input type="checkbox"/>	Seine1		606.8 μs
●		<input type="checkbox"/>	Oslo1		672.0 μs
●		<input type="checkbox"/>	Zazu1		781.5 μs
●		<input type="checkbox"/>	Antwerp1		678.0 μs
●		<input type="checkbox"/>	Rhine2		2.2 ms
●		<input type="checkbox"/>	Tagus2		906.3 μs
●		<input type="checkbox"/>	Rhine1	192.5 MB	2.3 ms

View Server Requests

-  View Probe Summary
-  View Portal Components
-  View Load
-  Open Mercury Diagnostics Profiler
-  Create Alert Rule...
-  Create Comments...

- 2 Open the Create Alert Rule dialog by selecting the **Create Alert Rule** menu option. The dialog is displayed as shown in the following example:

Create Alert Rule [X]

 Create an Alert Rule for the Probe Dublin2.

Name:

Description:

Alert Triggers

- Alert when entity status turns red
- Alert when no entity data has been received for 5 minutes

Alert Notification Options

- SNMP
- E-mail

Addresses (Comma-Separated List):

OK Cancel

- 3 Enter the information requested on the Alert Notification Rule dialog:
 - ▶ Enter a **Name** that will help you to remember the entity and the reason why you wanted to receive alert notifications. For example, if the alert rule was for a host named Prod_010 that had been experiencing CPU usage spikes, you may enter a name such as Prod_010 CPU Check.

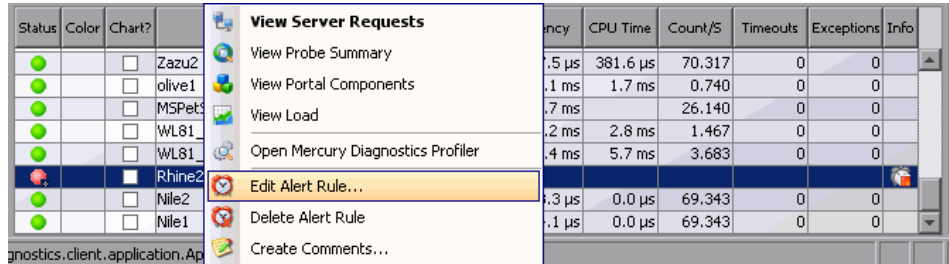
Note: You may want to have more generic names if there is more than one metric with a threshold value that could trigger alert notification. A specific name that includes the metric could be misleading.

- ▶ Enter a short **Description** for the rule.
- ▶ Select one or more **Alert Triggers** that will cause notifications to be sent.
 - If you select **Alert when entity status turns red**, the notifications are sent whenever the status of the entity turns red. A red status is triggered whenever one of the metric thresholds on the entity has been exceeded, or when any of the metric thresholds have been exceeded on its sub-entities. In other words, a probe can turn red if any of its server requests turn red, and a probe group turns red if any of its probes turn red.
 - If you select **Alert when no entity data has been received for 5 minutes**, the notifications are sent whenever the Diagnostics Server has received no data from the entity for 5 minutes.
 - If you do not select an Alert Trigger, the alert notification rule is created but remains disabled.
- ▶ Select one or more **Alert Notification Options** that determine how the alert notifications are to be delivered.
 - If you select **SNMP**, the notifications are sent via SNMP traps (v2c). Note that the server installation includes a MIB file to help parse these traps.
 - If you select **E-mail**, the notifications are sent via e-mail.
 - If you do not select an Alert Notification Option, the alert notification rule is created but remains disabled.

To maintain alert notifications rules:

Right-click on a Probe, Probe Group, Host, or Server Request listed in a Status view on the detail table of a detail view. Diagnostics displays the pop-up menu for the selected entity.

When an alert notification rule has been created for the selected entity, the pop-up menu appears as shown in the following example from the Probe view:



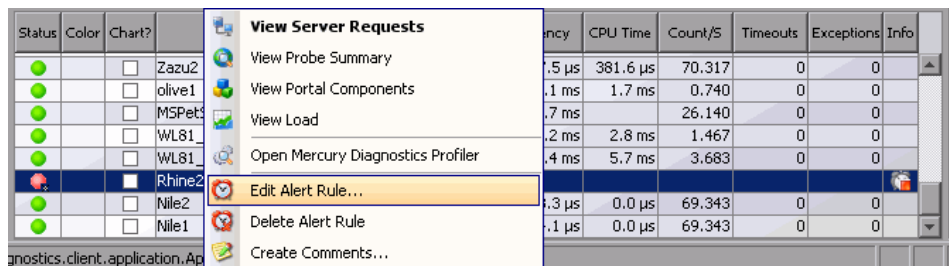
Open the Edit Alert Rule dialog by selecting the **Edit Alert Rule** menu option.

The Edit Alert Rule dialog is the same as the Create Alert Rule dialog. For instructions on using the dialog see “To create an alert notification rule:” on page 87.

To delete alert notifications:

- 1 Right-click on a Probe, Probe Group, or Host listed in a Status view or the detail table of a detail view. Diagnostics displays the pop-up menu for the selected entity.

When an alert notification rule has been created for the selected entity, the pop-up menu appears as shown in the following example from the Probe view:



- 2 Select **Delete Alert Rule** to indicate that the alert notification rule for the selected entity is to be deleted.
- 3 Click **Yes** in the Delete Alert verification dialog box to confirm the rule deletion.

Reviewing Alert Notification Rules

Diagnostics provides a view that allows you to review and maintain all of the alert notification rules that you have created in one convenient and powerful view. The **Alert Rules** view displays a list of all of the alert notification rules sorted by default in alphabetical order of the entity name. From the list, you can maintain the alert notification rules and navigate to the Diagnostics views that are associated with the entities that have alert notification rules. The following is an example of the Alert Rule view:

Entity Name	Entity Type	Alert Name	Last Fired	Info
/estore/control/commitorder	Server Request	/estore/control/commitorder ...		
Joyride1	Probe	Joyride1 Status Alert		
rhine.lab.performant.com	Host	Rhine2 Status Alert		

In this example, the Info column indicates that there are three alert notification rules that have been defined: one for a server request and two for a probe. Two of the alert notification rules are currently active and one has been disabled. In addition, the example shows that two of the entities have comments associated with them.

Understanding the Columns in the Alert Rules View

The columns in the Alert Rules view provide the same information and navigation options as their counter parts in the detail table of the detail view.

When you right-click on the values in the columns, Diagnostics displays the same pop-up menu that appears in the detail view for the entity. If an alert notification has been issued based on the listed alert rule, the Last Fired column is not blank and the pop-up menu contains the **View Threshold Violation** option.

The columns are:

Entity Name: This column contains the name of the entity listed in the view. When you hold your mouse pointer over the values in the column the tooltip that Diagnostics displays is the same tooltip displayed in the detail view for the entity.

Entity Type: This column contains the type of the entity listed in the row.

Alert Name: This column contains the name of the alert notification rule.

Last Fired: This column contains the timestamp for the last time the listed alert notification fired.

Note: In reality, this represents the last time that the alert event fired recently. There is a limited number of alert notification events that are cached in the server.

Info. This column contains icons that remind you that you have entered alert rules or comments for the entity depicted in the row. One or more of the following icons can appear in this column:



- **Active Alert Rule.** This icon indicates that you have created an alert notification rule for the selected entity and the rule is active.



- **Disabled Alert Rule.** This icon indicates that you have created an alert notification rule for the selected entity but the rule is currently disabled.



- **Custom Comments.** This icon indicates that you entered comments for the selected entity.

Reviewing Alert Notification Events

You can review the events that triggered alert notifications in the log files and in the Alert Events view. The information in the alert notification event can help you know where to begin looking for the cause of the performance problem that triggered the alert.

Using the Alert Events View

Diagnostics lists each of the events that triggered an alert in the Alert Events view. By default, this table is sorted with the most recent alert events at the top. This can be especially helpful when the Alert Events view is included in a custom dashboard view.

Understanding the Columns in the Alert Events View

The columns in the Alert Events view provide the same information and navigation options as their counterparts in the detail table of the detail view.

When you right-click on the values in the columns, Diagnostics displays the same pop-up menu that it displays in the detail view for the entity except that the **View Threshold Violation** option is also included. This menu option is discussed in “Viewing Threshold Violations” on page 95.

The columns are:

Entity Name: This column contains the name of the entity listed in the view. When you hold your mouse pointer over the values in the column, the tooltip that Diagnostics displays is the same tooltip displayed in the detail view for the entity.

Alert Name: This column contains the name of the alert notification rule that was triggered by the alert event.

Timestamp: This column contains the timestamp for the time at which the alert event was triggered.

Threshold: This column contains the threshold value for the metric of the listed entity that triggered the alert event. Exceeding the threshold value is one of the possible triggers for an alert event.

Value: This column contains the value for the metric when the alert was triggered.



Status: The **Status** column contains the status indicator that lets you see, at a glance, how the listed entity is performing relative to the thresholds that you set for the metrics of the entity.

The color of the status indicator tells you the status of the entity. If you hold the mouse pointer over the status indicator in a row of the table, a tooltip displays a description of the current status:

- ▶ Red: Critical - The component is consistently exceeding defined thresholds.
- ▶ Grey: No status information available. Either no data has been received for the entity.

Viewing Alert Event Logs

You may view the alert events for all of the Diagnostics Servers in Mediator mode in the Alert Event view. If you want to review the alert events for a single Diagnostics Server in Mediator mode, you can view the server alerting log files for the particular Diagnostics Server:

- 1** Open the Diagnostics Server in Mediator mode administration page by navigating to the URL http://<diagnostics_server_host>:2006 in your browser, or by selecting **Start > Programs > Mercury Diagnostics Server > Administration** on the host for the Diagnostics Server in Mediator mode. The port number in the URL, **2006**, is the default port for the Diagnostics Server. If you configured the Diagnostics Server to use an alternative port, use that port number in the URL.
- 2** Access the Diagnostics Server Maintenance menu by selecting **Configure Diagnostics**
- 3** Select **logging > View Log Files**. Diagnostics displays a list of links to the log files that are available for viewing.
- 4** Select the link for `<diagnostics_server_install_dir>/log/server-alerting.log` to view the log messages for the Diagnostics Server.

Viewing Threshold Violations

From the Alert Events view and the Alert Rules view, you can request that Diagnostics display a detail view that depicts the threshold violation for a selected entity.

When you right-click on a row, Diagnostics displays a pop-up menu that contains the menu option **View Threshold Violation**. When you click this option, Diagnostics displays the detail view that is appropriate for the entity type depicted in the selected row. The detail view contains the selected entity in the detail table, and the graph contains the charted metrics that triggered the alert.

5

Using the Standard Detail Views

This chapter provides a detailed description of the detail views that are included in the Standard view group. 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 contributes to the observed application performance. For a high-level introduction to the Diagnostics views, see “Introducing Diagnostics Views” on page 17.

This chapter describes:	On page:
About the Standard Detail Views	98
Using the Hosts View	98
Using the Load View	103
Using the Probes View	109
Using the Probes View	109
Using the Server Requests View	115
Using the Transactions View	126
Using the Call Profile View	130
Using the Layer View	140
Using the Probes View	109

About the Standard Detail Views

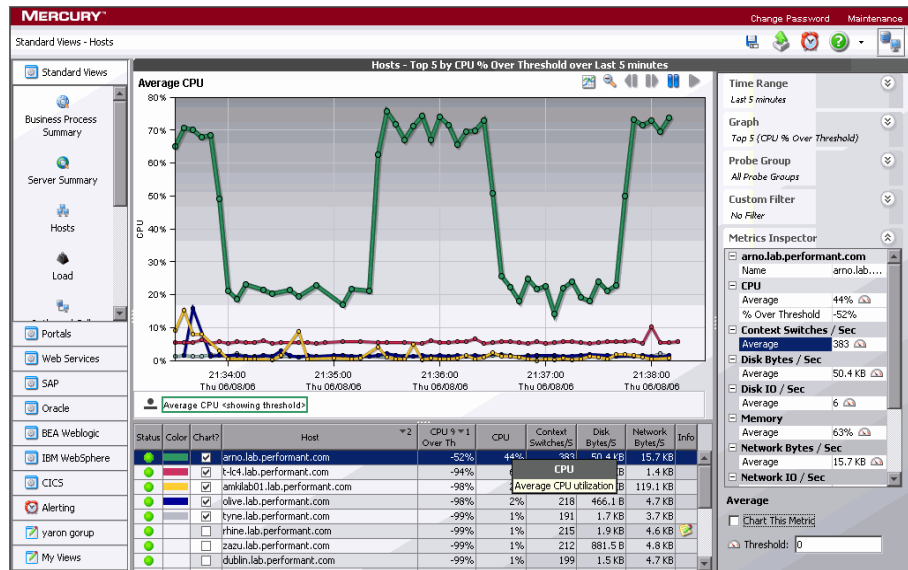
The standard detail views are predefined and installed with Diagnostics to be the primary tools that you use for analyzing the performance of your applications. To access the predefined standard detail views, open the **Standard Views** group on the View bar and click on the detail view that you want Diagnostics to display.

Using the Hosts View

The **Hosts** view contains performance metrics for the machines that host the probes and the applications they are monitoring. By default, the Hosts View graph presents trend lines that represent the CPU utilization percentage for each host. The Hosts view has the format of a detail view. For information about the layout and controls in the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

Description of the Hosts View

The following image is an example of the Hosts view:



Graph

By default in Mercury Business Availability Center and Diagnostics standalone, the Host View graph charts the CPU utilization metrics for the five hosts that have the highest utilization percentage during the previous five minutes. By default in LoadRunner and Performance Center, the Host View graph charts the CPU utilization metrics for the entire load testing scenario.

Diagnostics displays the CPU utilization for each host, using a trend line. The x-axis of the graph shows actual chronological time. The y-axis of the graph shows the percent utilization.

Detail Table

Diagnostics lists the hosts that pertain to the context shown in the bread crumbs in the detail table of the Hosts view. The metrics for each host that are reported in the table are aggregated and reported based on the time period specified in the **Time Range** view filter.

Metrics Inspector

The Metrics Inspector in the Hosts view lists the system metrics for the host in the selected row of the detail table. For information on configuring system metrics, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

Accessing the Hosts View

There are several ways to access the Hosts view within Diagnostics.

To access the Hosts view using the View bar:

- 1 Open the **Standard Views** group on the View bar.



- 2 Click **Hosts** in the Standard Views group.

Diagnostics displays the Hosts view with the default settings where the five hosts that have the highest CPU utilization during the previous five minutes are charted on the graph.

To access the Hosts view from a dashboard view:

- 1 Open a dashboard view that contains a monitoring version of the Hosts view, such as Server Summary.
- 2 Double-click the monitoring version of the Hosts view.

Diagnostics displays the Hosts view with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Hosts view was accessed from the dashboard view.

To access the Hosts view from the Status view:

Instructions for drilling down to the Hosts view for a particular host in the Status view can be found in “Drilling Down to a Host” on page 161.

Customizing the Hosts View

You can use the standard Diagnostics view controls to adjust the amount and type of data that Diagnostics displays in the Hosts view. For more information about the ways you can control how performance metrics are presented in this view, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Hosts View

Using the information displayed in the Hosts view, you can get an immediate understanding of the performance of your application on the hosts that are being monitored. If the metrics displayed in the Hosts view raises any concerns, you can drill down to find out more information from a view that depicts performance metrics at a lower level. For example, from the Hosts view, you can drill down to the probes on a given host, and then into the server requests captured by one of the probes.

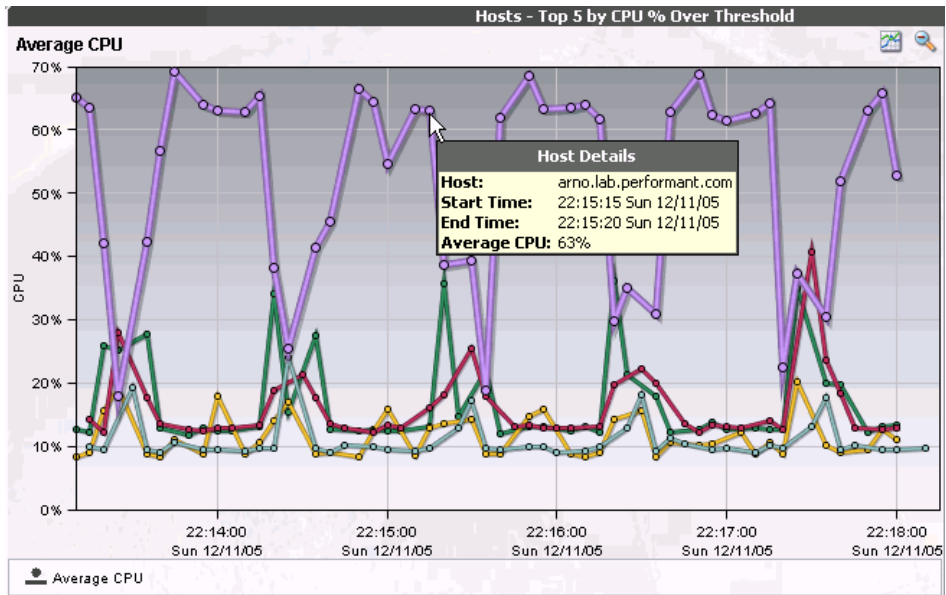
Displaying Host Details from the Charted Metrics

You can get additional details about a host whose metrics are charted in the graph by viewing the tooltips that Diagnostics displays for each charted trend line.

- ▶ When you hold your mouse pointer over a point on a charted trend line that is between two of the round nodes on the trend line, Diagnostics displays a **Host Details** tooltip.

The **Host Details** tooltip contains the name of the host whose performance is represented by the selected trend line in the graph.

- ▶ When you hold your mouse pointer over one of the nodes on the trend line for a charted metric, Diagnostics displays additional information in the **Host Details** tooltip, as shown in the following example:



The additional information shows the information that was used to plot the selected node on the trend line. The **Host Details** tooltip displays the following information:

- **Host.** The name of the host whose performance metric is represented by the selected trend line.
- **Start Time.** The start time for the aggregation period represented by the selected data point.
- **End Time.** The end time for the aggregation period represented by the selected data point.
- **Average CPU.** By default, the average CPU utilization is displayed for the period between the start time and end time. The actual metric that is displayed in the tooltip will depend on the metric that is represented by the trend line you selected.

Drilling Down from the Host in the Detail Table

You can drill down from a host in the detail table by:

- ▶ double-clicking the row for the host
- ▶ right-clicking the host's row
- ▶ double-clicking the trend line
- ▶ right-clicking the trend line
- ▶ right-clicking a row in the detail table, and selecting **View Probes** from the menu

When you drill down from a host, Diagnostics displays the Probes view with the probes that were running on the selected host during the specified time range. For more information on the Probes view, see “Using the Probes View” on page 109.

Maintaining an Alert Rule or Comments

When you right-click a row in the detail table, Diagnostics displays a menu for the selected host. This menu includes options for creating or maintaining alert notification rules and for creating or maintaining comments. For information on alert notification rules, see Chapter 4, “Alert Notification.” For information on entity comments, see “Maintaining Entity Comments” on page 79.

Using the Load View

The Load view contains the performance metrics for the Diagnostics layers where processing has taken place in your application. The Load view presents a breakdown of the load across the layers using a stacked area graph. The Load view has the format of a detail view.

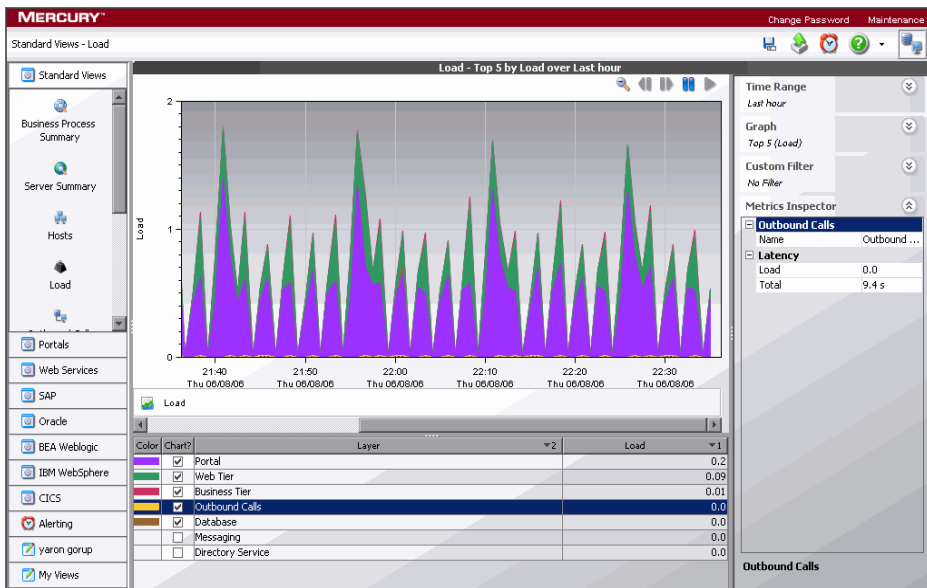
For information about Diagnostics layers, see the *Mercury Diagnostics Installation and Configuration Guide*. For information about the layout and controls in the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

The *load* in Diagnostics is determined based on a combination of your application's performance characteristics. Load is calculated to provide you with a powerful and concise view of how your application is performing. The characteristics that are used to determine the load for each Diagnostics layer are:

- ▶ The relative ratio of the amount of processing time spent in various layers over time.
- ▶ The relative amount of traffic on the monitored system over time.

Description of the Load View

By default, Diagnostics charts the load for the five layers that have the highest load values during the previous five minutes. Diagnostics depicts the load for each layer, using a stacked area graph as shown in the following example:



Graph

By default, the x-axis of the graph shows actual chronological time in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the scale for the calculated load values.

Detail Table

The detail table in the Load view lists all of the layers that pertain to the context shown in the bread crumbs displayed at the top of the view. The metrics reported in the table are filtered based on the time period specified in the **Time Range** list.

Metrics Inspector

The **Metrics Inspector** in the Load view lists the metrics for the selected row of the detail table.

Accessing the Load View

You can access the Load view from the **Standard Views** group on the View bar, by drilling down from a probe listed in the Probes view (in the Status view), and from a dashboard view that contains a monitoring version of the Load view.

To access the Load view using the View bar:

- 1 Open the **Standard Views** group on the View bar.
- 2 Click **Load** in the **Standard Views** group.



Diagnostics displays the Load view with the default settings so that the five layers that have the highest load during the previous five minutes are charted in the graph.

To access the Load view from a dashboard view:

- 1 Open a dashboard view that contains a monitoring version of the Load view.
- 2 Double-click the monitoring version of the Load view.

Diagnostics displays the Load view with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Load view was accessed from the dashboard view.

To access the Load View from other detail views:

- ▶ Instructions for drilling down to the Load view for a particular probe in the Probes view can be found in “Drilling Down from a Probe in the Detail Table” on page 114.
- ▶ Instructions for drilling down to the Load view for a particular probe in the Status view can be found in “Drilling Down to a Probe” on page 160.

Customizing the Load View

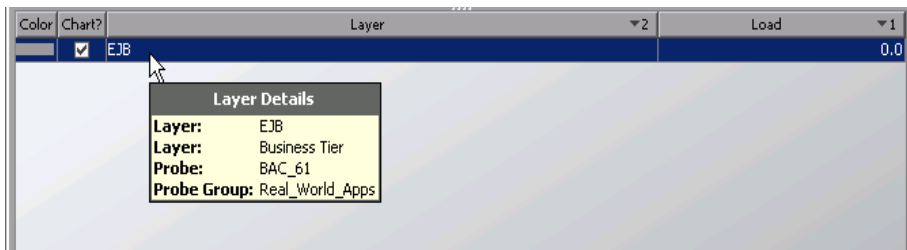
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Load view. For more information about the ways you can control how performance metrics are presented in this view, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Load View

Using the information contained in the Load view, you can get an immediate understanding of the performance of your application in the layers that are being monitored. If the information displayed in the Load view raises any concerns, you can drill down to the sub-layers.

Displaying Layer Details from the Detail Table

You can view more information about a layer listed in the detail table by holding the mouse pointer over the layer’s name in the **Layer** column. Diagnostics displays the **Layer Details** tooltip, as shown in the image below:

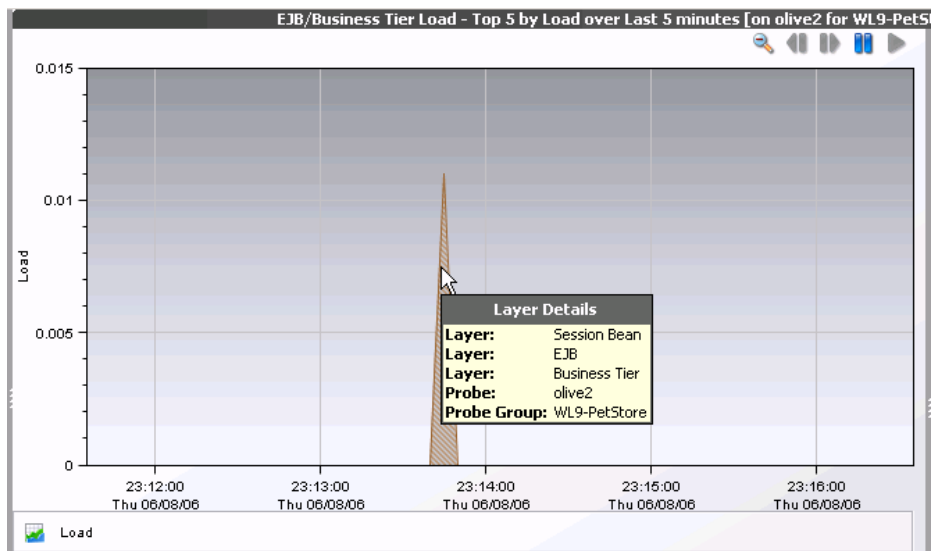


The information included in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Load view. In the example above, the Load view was accessed by drilling down from a probe in the Probes view through a layer in a higher-level Load view. The tooltip displays the following information:

- ▶ **Layer.** The name of the layer. This should be the name of the layer in the **Layer** column for the selected row in the detail table for the Load view.
- ▶ **Layer.** The name of the parent layer that was drilled into. This should be the name in the **Layer** column in the detail table of the next higher-level view listed in the bread crumbs.
- ▶ **Probe.** The name of the probe that captured the load. This value is included in the tooltip only when you access the Load view by drilling down from a probe in the Probes view or Status view.
- ▶ **Probe Group.** The name of the probe group that the probe was assigned to when it was installed. This value is included in the tooltip only when you access the Load view by drilling down from a probe in the Probes view or Status view.

Displaying Layer Details from the Charted Metrics

You can view more information about a layer charted in the graph by holding the mouse pointer over the top edge of a stacked area until the **Layer Details** tooltip is displayed, as in the following example:



The information that is included in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Load view. The example above is a Load view that was accessed by drilling down from two higher-level Load views. The tooltip displays the following information:

- ▶ **Layer.** The name of the layer.
- ▶ **Layer.** The name of the parent layer from which you drilled down.
- ▶ **Layer.** The name of the layer from which you drilled down to arrive at the parent layer.
- ▶ **Start Time.** The start time for the aggregation period represented by the selected data point.
- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Load.** The load calculated for the processing in this layer.

Drilling Down to a Layer in the Detail Table

You can drill down to a layer listed in the detail table by double-clicking or right-clicking the layer's row.

When you double-click a row in the detail table, Diagnostics displays the Load view with the sub-layers for the selected layer. If there are no sub-layers defined for the selected layer, double-clicking the row does not do anything.

When you right-click a row in the detail table, Diagnostics displays a menu with **View Load** as the only option. When you click **View Load**, Diagnostics displays the Load view with the sub-layers for the selected layer. If there are no sub-layers defined for the selected layer, the **View Load** menu option is disabled.

View Load displays a breakdown of the load for each sub-layer of the selected layer.

Using the Probes View

The Probes view displays performance metrics for the probes that are monitoring your applications. By default, the Probes View graph presents the VM heap usage for each probe using trend lines. The Probes view has the format of a detail view. For information about the layout and controls in the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

Note: Diagnostics continues to display information about a probe or probe group for as long as the data remains valid. If, for example, a probe disconnects, it disappears from view after several minutes. A probe will continue to appear in the table for as long as a week, while Diagnostics continues to report on the probe's availability.

Description of the Probes View

The following image is an example of the Probes view:



Graph

By default, the Probes View graph displays the heap usage for the five probes that have the highest average heap usage during the previous five minutes. Diagnostics displays the heap usage for each probe using a trend line.

By default, the x-axis of the graph shows actual chronological time in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the heap usage amount in megabytes (mb) or other related units, if more appropriate.

Detail Table

The detail table in the Probes view lists all of the probes that pertain to the context shown in the bread crumbs displayed at the top of the view. If you navigated to the Probes view from the View bar, the Probes view shows all of the probes. If you navigated to the Probes view from the Hosts view, the Probes view lists only the probes that were installed on the selected host. The metrics reported in the table are filtered based on the time period specified in the **Time Range** list and the probe group specified in the **Probe Group** list in the view filters.


Metrics Inspector

The Metrics Inspector in the Probes view lists the metrics for the selected row of the detail table. When the five-minute view is displayed, you can also see the status for those metrics.

Accessing the Probes View

You can access the Probes view from the **Standard Views** group on the View bar, by drilling down from a host listed in the Probes table in the Status view, and from a dashboard view that contains a monitoring version of the Probes view.

To access the Probes view using the View bar:

- 1 Open the **Standard Views** group on the View bar.
- 2  Click **Probes** in the **Standard Views** group.

The Probes view is displayed with the default settings so that the five probes that have the highest heap usage during the previous five minutes are displayed on the graph.

To access the Probes view from a dashboard view:

- 1** Open a dashboard view that contains a monitoring version of the Probes view.
- 2** Double-click the monitoring version of the Probes view.

The Probes view is displayed with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Probes view was accessed from the dashboard view.

To access the Probes view from the Status view:

Instructions for drilling down to the Probes view for a particular host in the Status view can be found in “Drilling Down to a Host” on page 161.

Customizing the Probes View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Probes view. For more information about the ways you can control how performance metrics are presented in this view, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Probes View

Using the information displayed in the Probes view, you can get an immediate understanding of the performance of your application on the probes that are being monitored. If the information displayed in the Probes view raises any concerns, you can drill down to find out more information from a more detailed view of the underlying performance metrics.

Displaying Probe Details from the Detail Table

You can view details for a probe listed in the detail table by holding your mouse pointer over that probe in the **Probe** column until the **Probe Details** tooltip is displayed:

Status	Color	Chart?	Probe	VM Heap Used	Latency	Count/s
●	■	<input checked="" type="checkbox"/>	Troy2	212.5 MB	2.4 ms	0.6
●	■	<input checked="" type="checkbox"/>	Athens2	204.2 MB		
●	■	<input checked="" type="checkbox"/>	Golem2	192.2 MB	8.3 ms	0.4
●	■	<input checked="" type="checkbox"/>	WL61_Porta	189.1 MB	408.5 ms	1.9
●	■	<input checked="" type="checkbox"/>	Budapest2	188.2 MB	8 ms	1.1
●	■	<input type="checkbox"/>	Athens1	185.9 MB	6.7 ms	0.6
●	■	<input type="checkbox"/>	Oder1	164.2 MB	15.7 ms	1.8
●	■	<input type="checkbox"/>	Tagus1	151.7 MB	9.9 ms	0.9
●	■	<input type="checkbox"/>	Dublin2	150.5 MB	12.8 ms	0.9

Probe Details

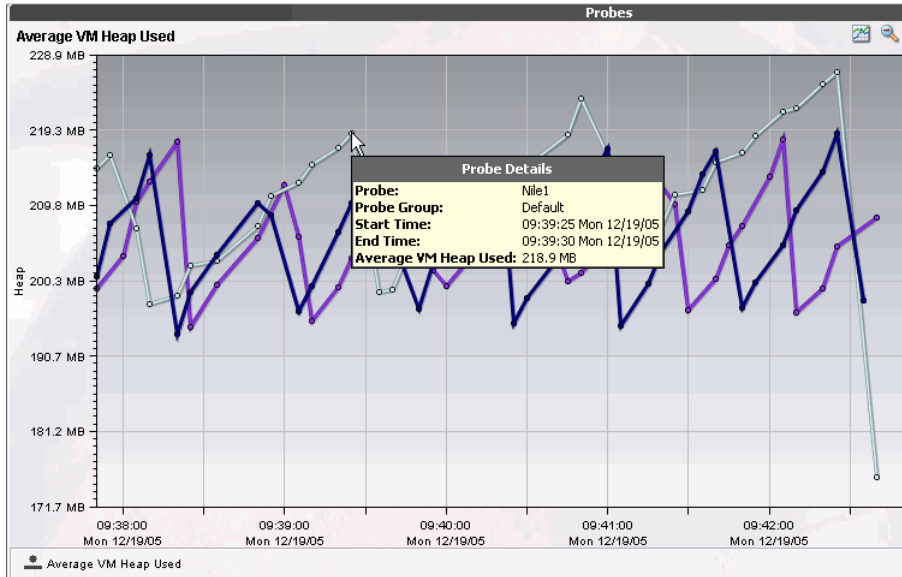
Probe: Athens2
Probe Group: Default
Host: athens.lab.performant.com
Probe Type: Java

The **Probe Details** tooltip displays the following information:

- **Probe.** The name of the probe whose metrics are represented.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- **Host.** The host on which the probe is running.
- **Probe Type.** J2EE, .NET, Oracle, or SAP_R3.

Displaying Probe Details from the Charted Metrics

You can view more information about a probe displayed in the graph by holding the mouse pointer over one of the nodes on the trend line for a charted metric until the **Probe Details** tooltip is displayed.



The **Probe Details** tooltip displays the following information:

- **Probe.** The name of the probe whose metrics are represented by the selected trend line in the graph.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- **Start Time.** The start time for the aggregation period represented by the selected data point.

- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Average VM Heap Used.** The average heap usage for the probe for the period between the start time and end time.

Note: The actual metric that is displayed in the tooltip depends on the metric that is represented by the selected trend line.

Drilling Down from a Probe in the Detail Table

You can drill down from a probe listed in the detail table by double-clicking or right-clicking the probe's row.

When you double-click a row in the detail table, Diagnostics displays the Server Requests view, with the server requests that are being run on the selected probe.

When you right-click a row in the detail table, Diagnostics displays a menu with the following options:

- ▶ **View Server Requests.** This is the same option that you get if you double-click the row. For information on the Server Requests view, see "Using the Server Requests View" on page 115.
- ▶ **View Probe Summary.** The Probe Summary is a dashboard view that is made up of detail views. For information on dashboard views, see "Introducing Diagnostics Dashboard Views" on page 29.
- ▶ **View Portal Components.** For information on the Portal Components view, see "Using the Probes View" on page 109.
- ▶ **View Load.** For information on the Load view, see "Using the Load View" on page 103.
- ▶ **Open Mercury Diagnostics Profiler.** For information on the Mercury Diagnostics Profilers, see Part III, "Using the Mercury Diagnostics Profiler for J2EE" or Part IV, "Using the Mercury Diagnostics Profiler for .NET."
- ▶ **Add/Edit Comment.** For information on entity comments, see "Maintaining Entity Comments" on page 79.

- ▶ **Add/Edit Alert Rule.** For information on alert notification rules, see Chapter 4, “Alert Notification.”

Using the Server Requests View

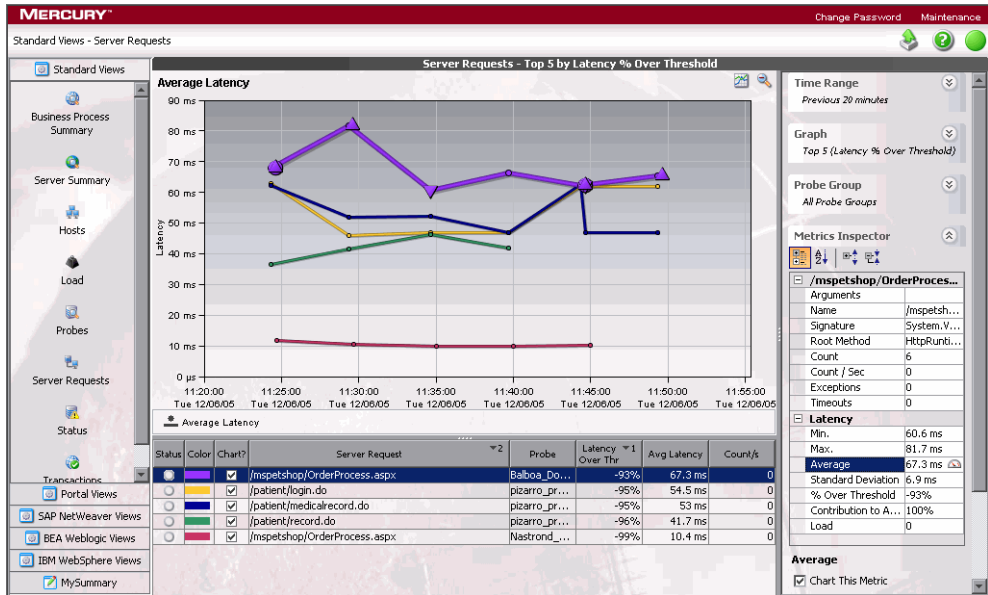
The Server Requests view displays the performance metrics for the monitored server requests in your application. The Server Requests view has the format of a detail view. For information about the layout and controls in the detail view, see Chapter 3, “Detail Views: Layout and Controls.”

Description of the Server Requests View

When you access the Server Requests view from the View bar or by drilling down from the Probes view, the view is presented in a trend format.

By default, the Server Requests view in the trend format displays the average latency for the five server requests that have the highest percent over threshold values during the previous five minutes. The metrics are displayed on the graph using trend lines.

The following image is an example of the Server Requests view in the trend format:



Accessing the Server Requests View

You can access the Server Requests view from the **Standard Views** group on the View bar, by drilling down from the entities of some of the other detail views, or from the dashboard views that include a monitoring version of the Status view.

To access the Status view using the View bar:

- 1 Open the **Standard Views** group on the View bar.
- 2 Click **Server Requests** in the **Standard Views** group.

The Server Requests view is displayed with the default settings so that the average latency for the five server requests that have the highest percent over threshold values during the previous five minutes are displayed on the graph.

To access the Server Requests view from a dashboard view:

- 1 Open a dashboard view that contains a monitoring version of the Server Requests view.
- 2 Double-click the monitoring version of the Server Requests view.

The Server Requests view is displayed with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Server Requests view was accessed from the dashboard view.

To access the Server Requests view from other detail views:

You can access the Server Requests view by drilling down to the metrics reported in the Probes view, the Transactions view, and the Status view.

- ▶ To drill down to the Server Requests view for a particular transaction in the Transactions view, see “Drilling Down from a Transaction in the Detail Table” on page 130.
- ▶ To drill down to the Server Requests view for a particular probe in the Probes view, see “Drilling Down from a Probe in the Detail Table” on page 114.
- ▶ To drill down to the Server Requests view for a particular probe in the Status view, see “Drilling Down from the Status View” on page 159.

Customizing the Server Requests View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Server Requests view. For more information about the ways you can control how performance metrics are presented in this view, see “Detail Views: Layout and Controls” on page 37.

Displaying CPU Time for Server Requests

By default, CPU time for server requests is not displayed on the Server Requests view. Tracking CPU time for all of the server requests significantly decreases the scalability and increases the amount of memory used by the Diagnostics Server.

There may be, however, times when you want to be able to see the CPU time for each server request. In these cases, you can turn on the reporting of CPU time in the Server Requests view by using the **track.fragment.cpu** property in `<diagnostics_server_install_dir>/etc/server.properties`.

Setting the value of the **track.fragment.cpu** property to **true** causes Diagnostics to report the CPU time for each server request in the Server Request view. The value of the CPU time for each server request is displayed in the Metrics Inspector. If you would like to have the CPU column displayed in the detail table, you can modify the detail table as described in “Customizing the Detail Table” on page 74.

Setting the value of the **track.fragment.cpu** property to **false** causes Diagnostics to stop reporting the CPU time for each server request in the Server Request view.

Interpreting the Server Requests View

Using the information displayed in the Server Requests view, you can get an immediate understanding of the performance of the server requests that are being run in your applications. If the information displayed in the Server Requests view raises any concerns, you can drill down to find out more information from a more detailed view of the underlying performance metrics.

Displaying Server Request Details from the Detail Table

You can view more information about a server request listed in the detail table by holding the mouse pointer over that server request in the **Server Request** column until the **Server Request Details** tooltip is displayed, as shown in the following example:

The screenshot shows the Mercury Diagnostics interface. A table lists server requests with columns for Status, Color, Chart?, Server Request, Probe, Latency Over The, Latency, and Count/s. A tooltip titled "Server Request Details" is displayed over the selected row, showing the following information:

Property	Value
Server Request:	/estore/control/commitorder
Probe:	rhine
Probe Group:	Default
Root Method:	MainServlet.doGet()
Method Signature:	void doGet(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse)
Package:	com.sun.j2ee.blueprints.petstore.control.web

The background table shows the following data for the selected row:

Status	Color	Chart?	Server Request	Probe	Latency Over The	Latency	Count/s
●	■	☑	/sampleportal/sample.portal	WL81_Por...	122%	2.2 s	0.3
●	■	☑	/CallChainWebApp/CallChain	WL81_Cro...	-10%	901.4 ms	0
●	■	☑	/estore/control/commitorder	rhine	-26%	743 ms	0.2
●	■	☑	/estore/control/commitorder	order2	-30%	704.7 ms	0.1

On the right side, the Metrics Inspector shows the following performance metrics:

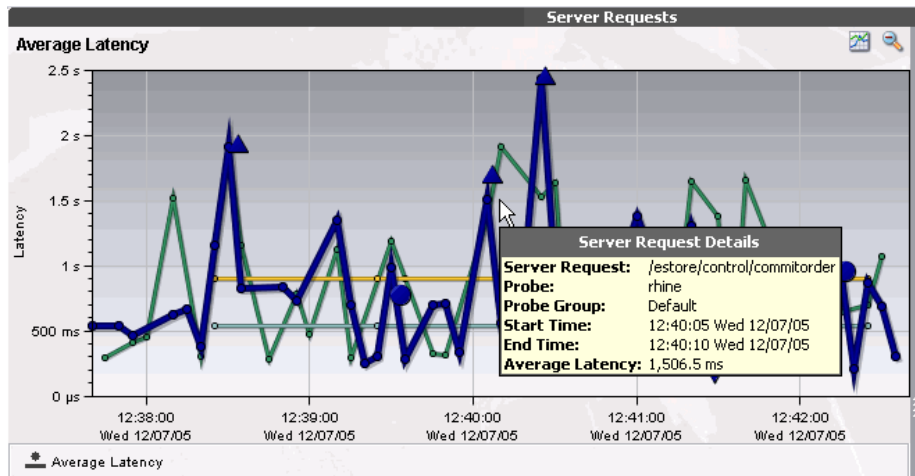
Metric	Value
Exceptions	355
Timeouts	0
Latency	2.2 s
Min.	756 ms
Max.	8.7 s
Average	2.2 s
Standard Deviation	1.4 s
% Over Threshold	122%
Contribution to A...	99%
Load	0.7

The **Server Request Details** tooltip displays the following information:

- **Server Request.** The name of the selected server request.
- **Probe.** The name of the probe that captured the server request. This should be the same name that is displayed in the **Probe** column of the Server Requests table.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- **Root Method.** The method that originated from the server request. This may be a portion of a URL or, in the case of an RMI call, may be a class and method representing the name of the server request itself.
- **Method Signature.** The method that was called as a result of the server request.
- **Package.** The name of the package that contains the class from which the method was called.

Displaying Server Request Details from the Charted Metrics

You can view more information about a server request listed in the graph by holding the mouse pointer over one of the nodes on the trend line for a charted metric until the **Server Request Details** tooltip is displayed, as shown in the following example:



The **Server Request Details** tooltip displays the following information:

- ▶ **Server Request.** The name of the selected server request.
- ▶ **Probe.** The name of the probe that captured the server request.
- ▶ **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- ▶ **Start Time.** The start time for the aggregation period represented by the selected data point.
- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Average Latency.** The average latency for the server request during the aggregation period.

Note: The actual metric that is displayed in the tooltip depends on the metric that is represented by the trend line you selected.

Drilling Down from a Server Request in the Detail Table

You can drill down from a server request listed in the detail table by double-clicking or right-clicking the server request's row.

When you double-click a row in the detail table, Diagnostics displays the Layers view for the selected server request.

When you right-click a row in the detail table, Diagnostics displays a menu for the selected server request with the following options:

- ▶ **View Layers.** Diagnostics displays the Layers view for the selected server request. The **View Layers** option displays a breakdown of the latency contribution for each layer in the server request. For more information on the Layers view, see "Using the Layer View" on page 140.
- ▶ **View Application Versions.** This option is relevant for WebLogic version 9 or later.
- ▶ **View Outbound Calls.** For more information, see "Using the Probes View," on page 109.

- ▶ **Open Mercury Diagnostics Profiler.** For information on the Mercury Diagnostics Profilers, see Part III, “Using the Mercury Diagnostics Profiler for J2EE” or Part IV, “Using the Mercury Diagnostics Profiler for .NET.”
- ▶ **Create/Edit/Delete Alert Rule.** For information on alert notification rules, see Chapter 4, “Alert Notification.”
- ▶ **Create/Edit/Delete Comment.** For information on entity comments, see “Maintaining Entity Comments” on page 79.

Drilling Down to an Instance Tree for a Server Request

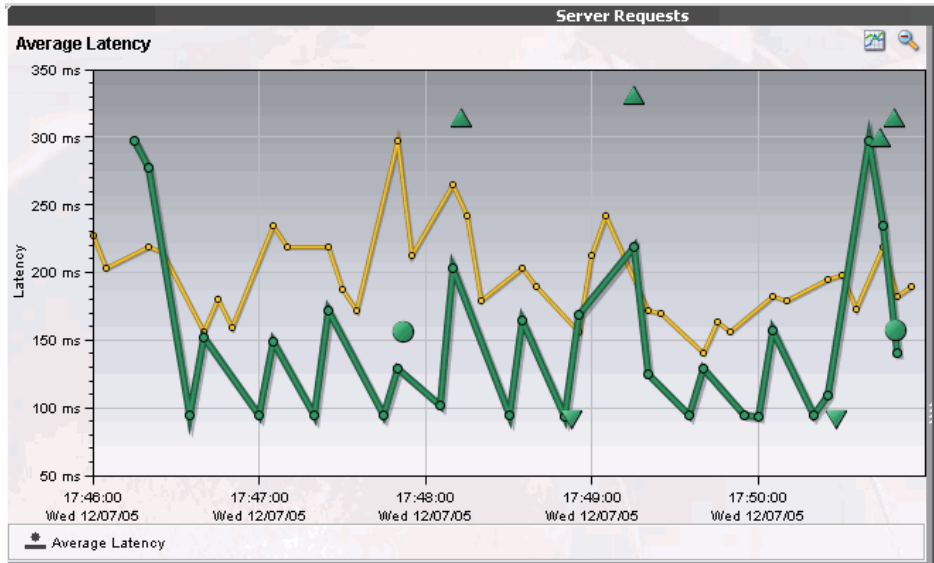
You can drill down from the graph in the Server Requests view to see a call profile for particular periods of time. The call profile is presented in the Call Profile view as an instance tree that depicts the method calls and their latency in a graph and in a table. For more information on the Call Profile view, see “Using the Call Profile View” on page 130.

When you select a metric displayed in the Server Request view, by either clicking the metric in the graph or by selecting the row for the server request from the detail table, the trend line for the metric is highlighted by a thicker line and larger points.

In addition, special icons are displayed to mark the points at which instance trees have been created for significant method calls that were made during the execution of the server request.

Note: The instance trees are available only on the Min, Max, and Average Latency metrics.

In the following example, the green server request trend line has been selected in the graph of the Server Requests view:



The circle and triangle icons that appear above, below, and on top of the selected trend line are the markers that indicate where instance trees for the server request have been saved to help you understand the performance of your applications.

About Instance Trees

For a detailed explanation of instance trees and how Diagnostics captures them, see Chapter 9, “Instance Trees.”

About Instance Tree Markers in the Server Requests Graph

Instance tree markers are displayed for the selected service request trend line. The markers are placed on the graph to indicate the total latency and the end time for the server request for which the instance tree was created.

Four instance tree marker icons are used to represent the four different types of instance trees that are created for a server request:



► Maximum Instance Tree.

The maximum instance tree for a server request represents an instance tree with the largest latency for the time period. This depicts one of the worst-performing invocations of the server request and resulting root method. This is one of the primary tools used for diagnosing the root cause of poor application performance.



► Minimum Instance Tree.

The minimum instance tree for a server request represents an instance tree with the smallest latency for the time period. This depicts one of the best-performing invocations of the selected server request and resulting root method, and can be useful for comparison to instances that perform poorly.



► Average Instance Tree.

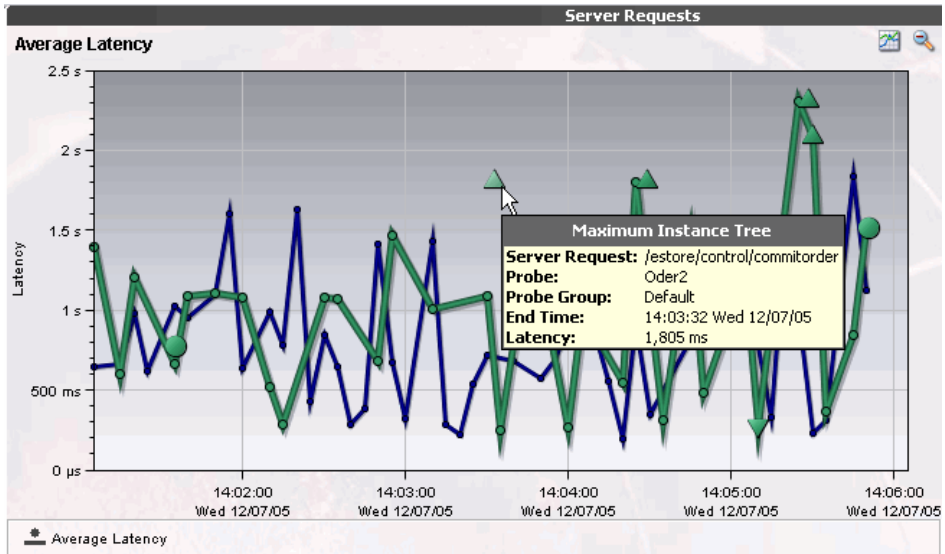
The average instance tree for a server request represents an instance tree that represents the average latency for the time period. This depicts a typical invocation of the selected server request and resulting root method call, and can be useful for comparison to instances that perform poorly.



► Cross VM Instance Tree.

The cross VM instance tree for a server request represents an instance tree that includes a call to another server request via a technology such as RMI or WebServices.

When you hold the mouse pointer over one of the instance tree markers on the graph, a tooltip identifies the type of instance tree marker, along with information about the server request for which the instance tree was created. The following image shows an example of the tooltip for a Maximum Instance Tree.



The instance tree marker tooltip displays the following information:

- ▶ **Title.** The type of instance tree marker that has been selected.
- ▶ **Server Request.** The name of the server request selected in the **Server Request** column.
- ▶ **Probe.** The name of the probe that captured the server request.
- ▶ **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- ▶ **End Time.** The end time for the server request invocation represented by the selected instance tree.
- ▶ **Latency.** The latency for the server request depicted in the instance tree.

Drilling Down from the Instance Tree

There are two ways to drill down from the instance trees that are represented by the instance tree markers on the graph of the Server Request view:

- ▶ Click the instance tree marker. Diagnostics displays the Call Profile view for the selected instance in the Diagnostics UI.
- ▶ Right-click the instance tree marker. Diagnostics opens a new window to display the Call Profile View. This is useful when you want to compare the call profiles for several different server requests or for the same server requests at different times. See Chapter 9, “Instance Trees” for instructions on understanding and using the call profiles to do performance analysis.

Note: It is possible that the instance tree that is displayed for an instance tree marker is different than the one that was anticipated. This is because the instance tree on the Diagnostics Server may have been replaced since the last update to the UI. So, even though a specific tree was selected, another tree that matches the type of the selected instance tree is displayed

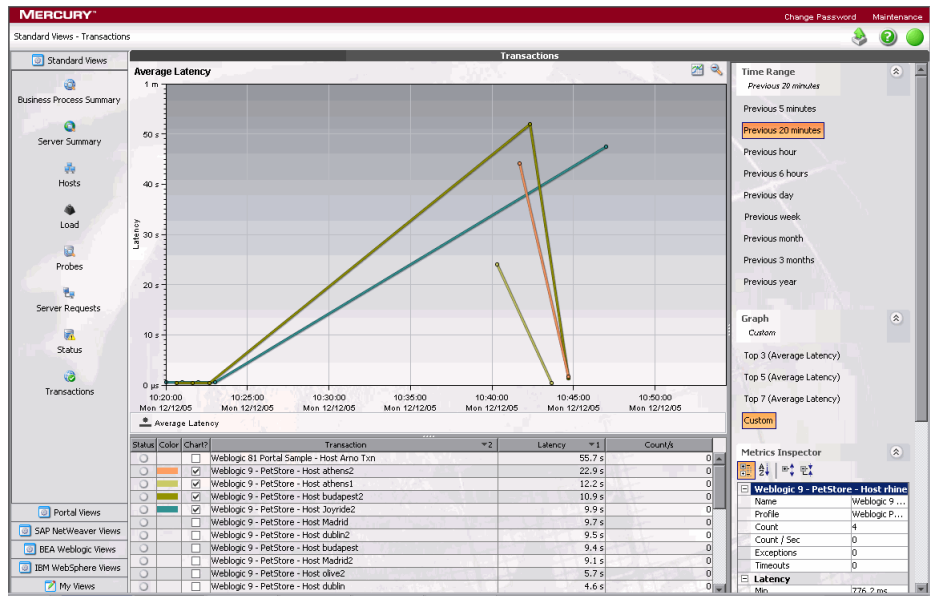
For more information on the Call Profile view, see “Using the Call Profile View” on page 130.

Using the Transactions View

The Transactions view displays performance metrics for the transactions that are being executed by your applications. This view has the format of a detail view. For information about the layout and controls in the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

Description of the Transactions View

The following image is an example of the Transactions view:



Graph

By default, the Transactions View graph displays the five transactions that have the highest average latency during the previous hour. Diagnostics displays the average latency for each transaction using a trend line.

By default, the x-axis of the graph shows the actual chronological time in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the average latency in seconds and milliseconds.

Detail Table

The detail table in the Transactions view lists all of the transactions that pertain to the context shown in the bread crumbs displayed at the top of the view. If you navigated to the Transactions view from the View bar, the Transactions view shows all of the transactions. The metrics reported in the table are filtered based on the time period specified in the **Time Range** list, and the probe group specified in the **Probe Group** list in the view filters.

Metrics Inspector

The Metrics Inspector in the Transactions view lists the metrics for the selected row of the detail table. For more information, see “Working with the Metrics Inspector” on page 48.

Accessing the Transaction View

You can access the Transactions view from the Standard Views group on the View bar, and from a dashboard view that contains a monitoring version of the Transactions view.

To access the Transactions view using the View bar:

- 1 Open the **Standard Views** group on the View bar.
- 2 Click **Transactions** in the **Standard Views** group.



The Transactions view is displayed with the default settings so that the five transactions that have the highest latency during the previous hour are displayed on the graph.

To access the Transactions view from a dashboard view:

- 1 Open a dashboard view that contains a monitoring version of the Transactions view.
- 2 Double-click the monitoring version of the Transactions view.

The Transactions view is displayed with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Transactions view was accessed from the dashboard view.

Customizing the Transactions View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Transactions view. For more information, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Transactions View

Using the information displayed in the Transactions view, you can get an immediate understanding of the performance of your application for the business transactions that are being monitored. If the information displayed in the Transactions view raises any concerns, you can drill down to find out more information from a more detailed view of the underlying performance metrics.

Displaying Transaction Details from the Detail Table

You can view details for a transaction listed in the detail table by holding your mouse pointer over that transaction in the **Transaction** column. The **Transaction Details** tooltip is displayed, as shown in the following example:

Status	Color	Chart?	Transaction	Latency	Count/s
<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	Weblogic 81 Portal Sample - Host Arno Txn	51.4 s	0
<input type="checkbox"/>		<input checked="" type="checkbox"/>	Weblogic 9 - PetStore - Host oder2	2.3 s	0
<input type="checkbox"/>		<input checked="" type="checkbox"/>	Weblogic 9 - PetStore - Host oder	2.2 s	0
<input type="checkbox"/>		<input checked="" type="checkbox"/>	Weblogic 9 - PetStore - Host Joyride	1.7 s	0
<input type="checkbox"/>		<input checked="" type="checkbox"/>	Weblogic 9 - PetStore - Host rhine	1.2 s	0
<input type="checkbox"/>		<input type="checkbox"/>	Cross VM Call Chain Txn - tyne Probl	958.8 ms	0
<input type="checkbox"/>		<input type="checkbox"/>	Weblogic 9 - PetStore - Host olive2	885.2 ms	0
<input type="checkbox"/>		<input type="checkbox"/>	Weblogic 9 - PetStore - Host dublin2	846.7 ms	0
<input type="checkbox"/>		<input type="checkbox"/>	Weblogic 9 - PetStore - Host tagus2	741 ms	0
<input type="checkbox"/>		<input type="checkbox"/>	Weblogic 9 - PetStore - Host tagus	740.7 ms	0

Transaction Details

Transaction: Weblogic 9 - PetStore - Host oder

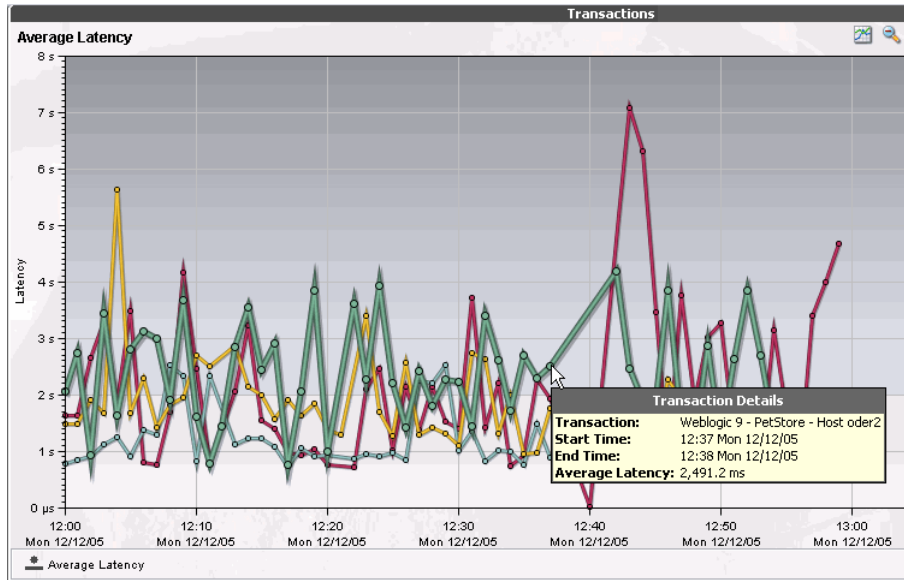
Profile: Weblogic PetStore

The **Transaction Details** tooltip displays the following information:

- ▶ **Transaction.** The name of the transaction whose metrics are represented by the selected trend line in the graph.
- ▶ **Profile.** The type of application server and the application.

Displaying Transaction Details from the Charted Metrics

You can view more information about a transaction displayed in the graph by holding the mouse pointer over one of the nodes on the trend line for a charted metric until the **Transaction Details** tooltip is displayed.



The **Transaction Details** tooltip displays the following information:

- ▶ **Transaction.** The name of the transaction whose metrics are represented by the selected trend line in the graph.
- ▶ **Start Time.** The start time for the aggregation period represented by the selected data point.
- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Average Latency.** The average latency for this transaction for the period between the start time and end time.

The actual metric that is displayed in the tooltip tip will depend on the metric that is represented by the trend line you selected.

Drilling Down from a Transaction in the Detail Table

You can drill down from a transaction listed in the detail table by double-clicking or right-clicking the transaction's row.

When you double-click a row in the detail table, Diagnostics displays the Server Requests view with the server requests that are being executed as part of the selected transaction.

When you right-click a row in the detail table, Diagnostics displays a menu for the selected transaction with the following options:

► **View Server Requests**

This is the same view that you see if you double-click the selected row. For information on the Server Requests view, see “Using the Server Requests View” on page 115.

► **View Layers**

For information on the Layers view, see “Using the Layer View” on page 140.

► **View Portal Components**

For information on the Portal Components view, see “Using the Probes View” on page 109.

Using the Call Profile View

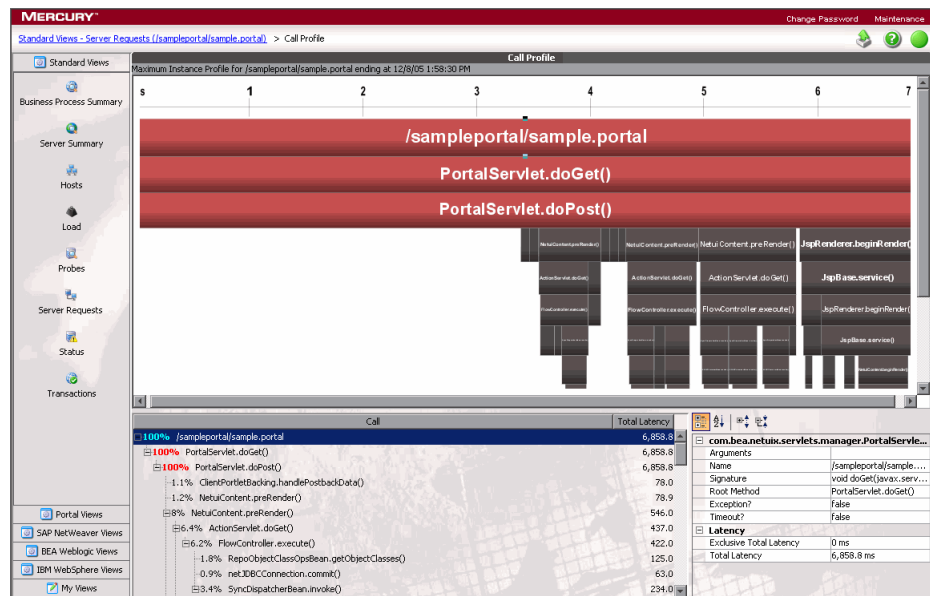
The Call Profile view displays the method calls and their latency for a particular instance of a monitored server request in your application. The Call Profile view displays the metrics in a graphical call profile, and in a table as a call tree. For an explanation of instance trees, see Chapter 9, “Instance Trees.”

The Call Profile view shares many features with the other detail views, but is different enough to be considered a unique view type.

Note: You may not save a Call Profile view as a custom view. When you use the Call Profile view, you must configure the view to customize the displayed information each time you access the view.

Description of the Call Profile View

The following image is an example of the Call Profile view:

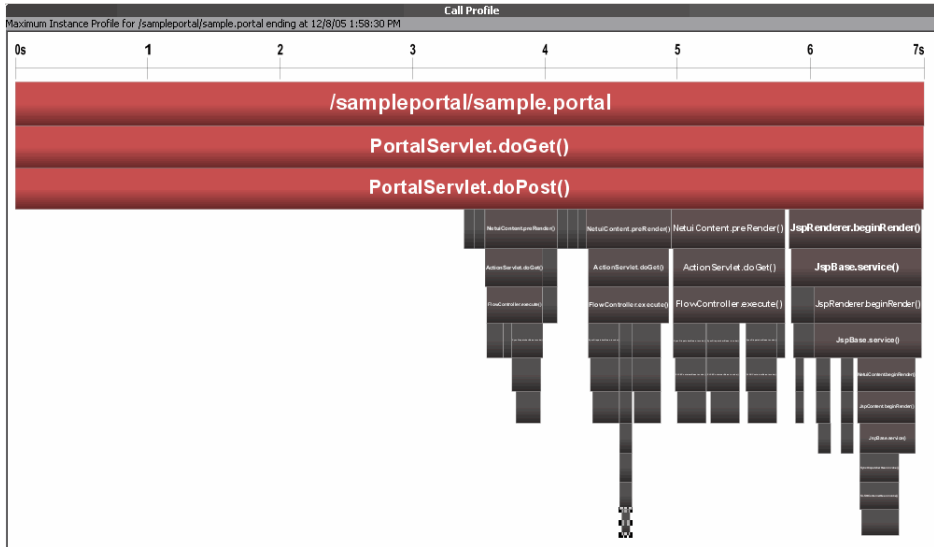


The Call Profile view is made up of three areas in addition to the View bar that is part of all Diagnostics views:

- Call Profile Graph
- Call Tree Table
- Metrics Inspector

Description of Call Profile Graph

The Call Profile graph displays the method calls that make up a server request in a graphical format, highlighting the latency of each call, the time when each call took place, and the call stack for each call.



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 on when they took place, and sequence of the calls relative to each other. The legend across the top of the instance profile denotes the amount of time, in seconds, since the server request was started.

The vertical axis on the instance 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. Calls made at deeper levels of the call stack are shown at the lower levels of the profile.

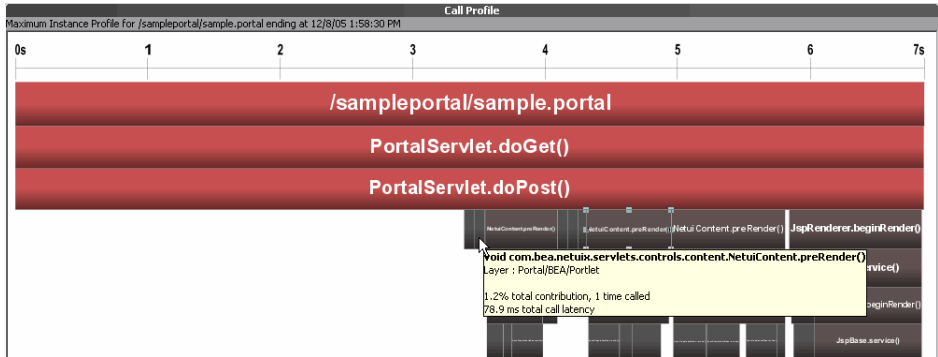
Each box in the instance profile represents a call where the left edge of the box is the start of the method call and the right edge is the return from the call. The length of the box indicates the duration of the call execution. The position of the call box along the horizontal axis indicates the actual time that the call started and ended. The call boxes that appear directly beneath a parent call box are the child calls that are invoked by the parent.

The gaps between the call boxes on a layer of the instance profile indicate one of the following processing conditions:

- ▶ The processing during the gap was taking place in code that is local to the parent at the previous higher level, and not in child calls in a lower layer.
- ▶ The processing during the gap was taking place in a child call that was not instrumented or included in a capture plan for the run.

The call boxes are colored to emphasize the critical path calls. The calls that are part of a path through the profile that has the highest latency are colored red. Call path components that are not part of a critical high-latency path are colored grey. Note that for a call profile showing a cross-VM call tree, each "hop" will be colored differently to help visually distinguish the calls that occurred on each tier.

If the duration of a call is very short or if the call appears further down in the call stack, the name of the method that is represented by the call box can become too small to read. You can view the details for a particular call by holding your mouse pointer over the call box until a tooltip is displayed. You can view additional information for a call by clicking the call box. This causes the selection in the tree table to move to that call, and prompts the Metrics Inspector to show the details for the call you clicked.



The tooltip displays the following call details:

- The method call
- The Diagnostics layer where the call occurred
- The percentage contribution to the total latency represented by the call
- The total latency of the call

Description of the Call Tree Table

The Call Tree table appears directly below the Call Profile graph. This table presents the information from the Call Profile graph in a tabular format.

Call	Total Latency	Total CPU
100% /wps/portal	19,422.0	18,031.2
100% Servlet.doGet()	19,422.0	
0.3% Servlet.determineContentType()	63.0	
3.8% FullEntitlementsEngine.getCheckEntitlements()	735.0	
2.6% RoleManager.loadRolesForPrincipal()	500.0	
1.4% WSJdbcConnection.prepareStatement()	265.0	
1.4% WSJdbcConnection.prepareStatement()	265.0	
1.4% b.prepareStatement()	265.0	
1.4% c.prepareStatement()	265.0	
0.7% WSJdbcConnection.prepareStatement()	141.0	
0.7% WSJdbcConnection.prepareStatement()	141.0	
0.7% b.prepareStatement()	141.0	
0.7% c.prepareStatement()	141.0	
0.3% ResourceManager.getSharedChildNodesBySuperType()	63.0	
0.3% ResourceManagerDataAccess.getSharedResourcesByParentOIDsSuper	63.0	
0.3% ResourceManager.getSharedChildNodesBySuperType()	62.0	

The first row in the table contains the root of the call stack, which is the server request that you drilled down to when you requested that the Call Profile view be displayed for a server request. The child rows in the tree are the method calls that were made as a result of the server call. The method calls that are parents in the call stack have the expand/collapse controls in front of them so that you can display the parents and children as required.

The table contains the following columns:

- ▶ **Call.** The server request call or method call whose performance metrics are reported in the row. Preceding the call name is the percentage contribution of the method call to the total latency of the service request.
- ▶ **Total Latency.** The total latency for the call. The time is reported in milliseconds.

Note: The total latency for a parent call includes not only the sum of the latency of each of its children, but also the latency for the processing that the method did on its own.

Note: When you click a row call in the Call Tree table, the corresponding box is selected in the Call Profile graph, and the metrics for the selected call are displayed in the Metrics Inspector.

Description of the Metrics Inspector

The Metrics Inspector lists the metrics for the call that is selected in the Call Tree table and the Call Profile. For more information about the Metrics Inspector, see “Working with the Metrics Inspector” on page 48.

Note: The Metrics Inspector in the Call Profile view does not enable setting thresholds or charting other metrics.

Accessing the Call Profile View

You can access the Call Profile view by drilling down from server requests in the Server Request view.

To access the Call Profile view from the Server Request View, see “Using the Server Requests View” on page 115.

Note: It is possible that the instance tree that is displayed for an instance tree marker on the Server Request view is different than the one that was anticipated. This is because the instance tree on the Diagnostics Server may have been replaced since the last update to the UI. So, even though a specific tree was selected, another tree that matches the type of the selected instance tree is displayed.

Interpreting the Call Profile View

Using the information displayed in the Call Profile view, you can get an immediate understanding of the method calls that are being invoked as a result of the server requests in your application, including their latency and percent contribution to the total latency.

Analyzing Performance with a Call Profile Graph

The Call Profile graph enables 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.

Note: When you click a call box in the Call Profile graph, the corresponding row is selected in the Call Tree table, and the metrics for the selected call are displayed in the Metrics Inspector.

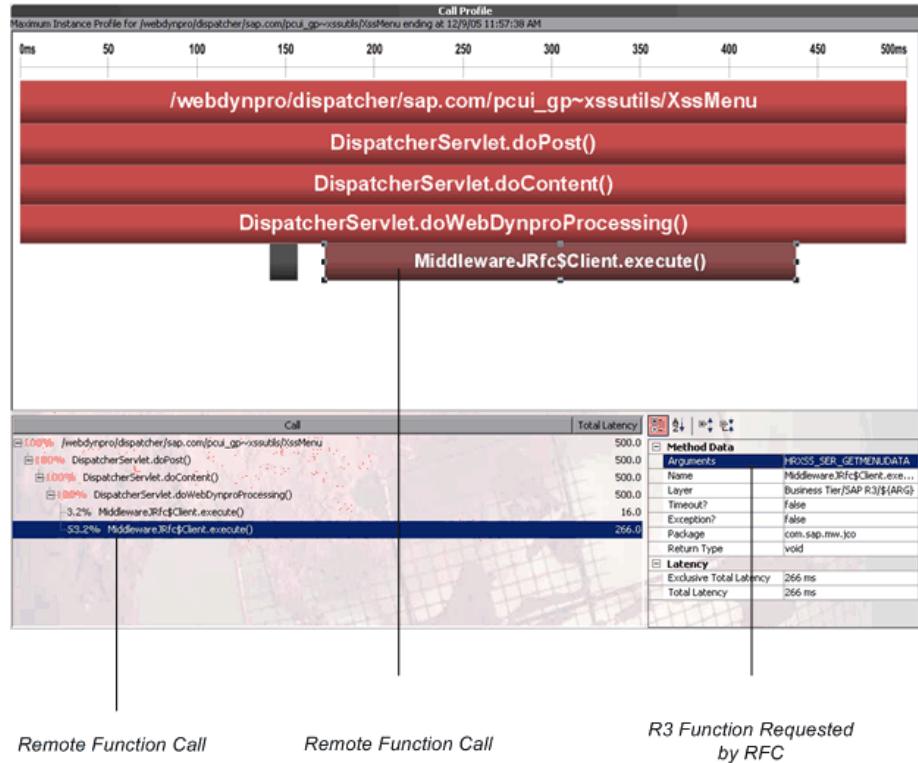
Comparing Call Profile Graphs

The Call Profile view enables you to compare two or more call profiles.

When you right click an instance tree marker, the **Open Call Profile in New Window** menu option is displayed. Selecting this option causes the Call Profile view to be displayed in a new window without the **View Bar**. You may then return to the original window and navigate to select another instance tree to display in the Call Profile view. Once the new tree is displayed, you can compare it with the tree that is displayed in the alternate window.

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 Call Profile view, as shown in the following screen image:



When a JAVA method executes an RFC, it specifies the R3 function as an argument 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 Metrics Inspector. The **Arguments** entry under the **Method Data** heading the Metrics Inspector contains the R3 function that was being executed by the selected RFC.

Analyzing Remote Method Invocation (RMI)

When analyzing RMI in the Call Profile view, 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 table with just the latency for its own processing.

Analyzing Life Cycle Methods for Portlets

Life cycle methods for portlets are identified by the name of the method (beginRender, endRender, and so on), and the portlet on which the method was invoked. The portlet name is a combination of its title and label.

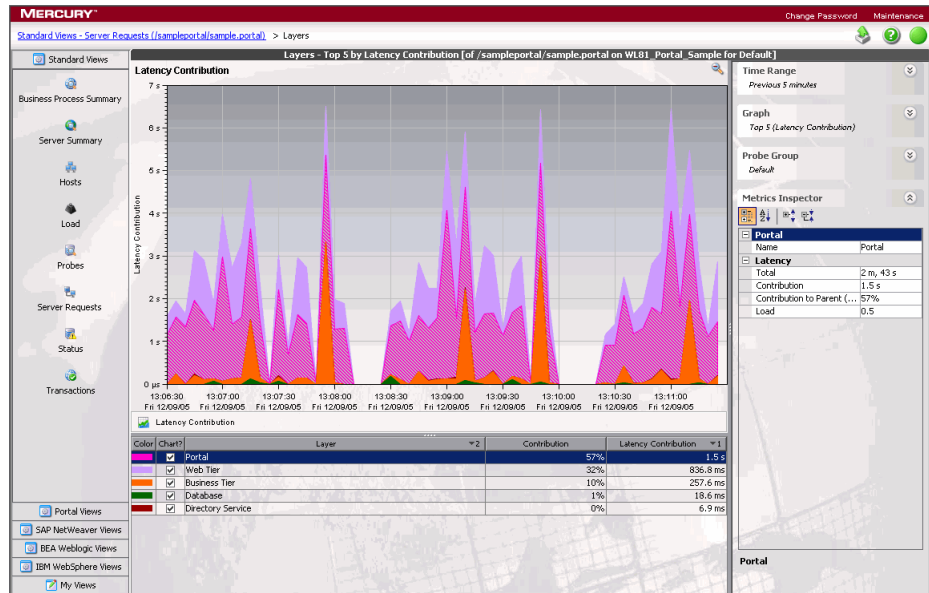
Using the Layer View

The Layers view displays the performance metrics for the Diagnostics layers where processing has taken place in your application. The Layers view presents a breakdown of the processing across the layers using a stacked area graph. The Layers view has the format of a detail view. For information about the layout and controls in the detail views, see Chapter 3, “Detail Views: Layout and Controls.”

Occasionally, due to design decisions, a class in your application that does not directly implement a J2EE component may contain J2EE functionality that you would like to monitor. Likewise, you may want to monitor a particular class that is of special interest to you in a custom layer. In situations like these, where you want to monitor specific classes in specific ways, Diagnostics enables you to define custom layers. You must configure the instrumentation of the probe when you want Diagnostics to monitor and report on custom classes or packages. For instructions on configuring the instrumentation of the probes, see the *Mercury Diagnostics Installation and Configuration Guide*.

Description of the Layers View

By default, Diagnostics displays the latency contribution for the five layers that have the highest percent contribution values during the previous five minutes in the Layers view. Diagnostics displays the latency contribution for each layer using a stacked area graph, as shown in the following example:



Layers View Graph

By default, the x-axis of the graph shows the actual chronological time in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the total latency contribution in seconds.

Detail Table

The detail table in the Layers view lists all of the layers that pertain to the context shown in the bread crumbs displayed at the top of the view. The metrics reported in the table are filtered based on the time period specified in the **Time Range** list.

Metrics Inspector

The Metrics Inspector in the Layers view lists the metrics for the selected row of the detail table.

Accessing the Layers View

The Layers view cannot be accessed directly from a view group on the View bar or from a standard dashboard view because layers are always displayed in the context of a transaction or server request. You can access the Layers view by drilling down to the metrics reported in the Transactions view or Server Requests view.

- ▶ To drill down to the Layers view for a particular transaction in the Transactions view, see “Drilling Down from a Transaction in the Detail Table” on page 130.
- ▶ To drill down to the Layers view for a particular server request in the Server Requests view, see “Drilling Down from a Server Request in the Detail Table” on page 120.

Customizing the Layers View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Layers view. For more information, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Layers View

Using the information displayed in the Layers view, you can get an immediate understanding of the performance of your application in the layers that are being monitored. If the information displayed in the Layers view raises any concerns, you can drill down to find out more information from a more detailed view of the underlying performance metrics.

Displaying Layer Details from the Detail Table

You can view more information about a layer listed in the detail table by holding the mouse pointer over that layer in the **Layer** column until the **Layer Details** tooltip is displayed, as shown below:

Color	Chart?	Layer	Contribution	Latency Contribution
	<input checked="" type="checkbox"/>	Portal	57%	1.3 s
	<input checked="" type="checkbox"/>	Web Tier	32%	706.3 ms
	<input checked="" type="checkbox"/>	Business Tier	10%	226.2 ms
	<input checked="" type="checkbox"/>	Database	0%	4.5 ms
	<input checked="" type="checkbox"/>	Directory Service	0%	2.3 ms

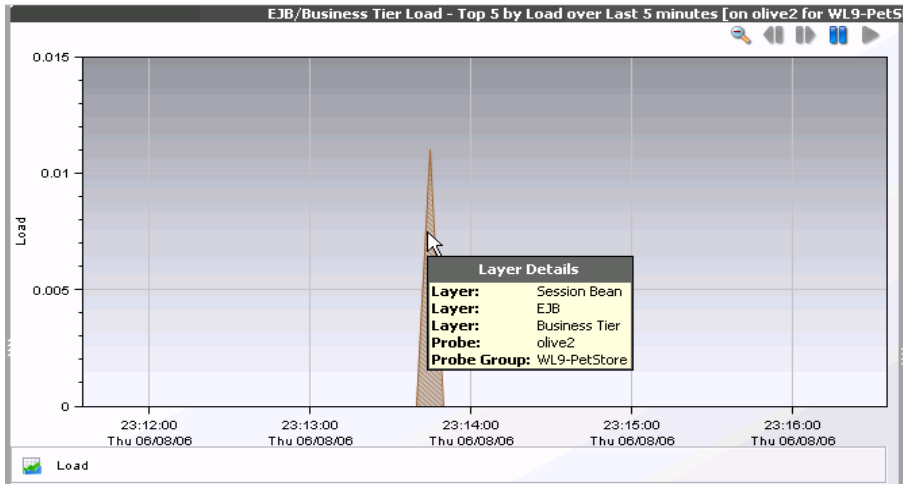
Layer Details	
Layer:	Web Tier
Server Request:	/sampleportal/sample.portal
Probe:	WLS1_Portal_Sample
Probe Group:	Default

The information displayed in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Layers view. In the example above, the Layers view was accessed by drilling down in the Server Request view. The **Layer Details** tooltip displays the following information:

- **Layer.** The name of the layer selected in the **Layer** column.
- **Server Request.** The name of the server request for which the layers are being displayed.
- **Probe.** The name of the probe that captured the server request.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.

Displaying Layer Details from Charted Metrics

You can view more information about a layer displayed in the graph by holding the mouse pointer over the top edge of a stacked area until the **Layer Details** tooltip is displayed, as shown in the following example:



The information displayed in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Layers view. In the example above, the Layers view was accessed by drilling down in the Server Request view. The tooltip displays the following information:

- ▶ **Layer.** The name of the layer.
- ▶ **Server Request.** This is the name of the server request for which the layers are being displayed.
- ▶ **Probe.** The name of the probe that captured the server request.
- ▶ **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- ▶ **Start Time.** The start time for the aggregation period represented by the selected data point.

- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Latency Contribution-** The amount of time, in milliseconds, that the processing in this layer contributed to the overall latency.

Drilling Down from a Layer in the Detail Table

You can drill down from a layer listed in the detail table by double-clicking or right-clicking the layer's row.

When you double-click a row in the detail table, Diagnostics displays the Layers view with the sub-layers for the selected layer. If there are no sub-layers defined for the selected layer, double-clicking the row will display a detail view, with a note in the table indicating that no data is available.

When you right-click a row in the detail table, Diagnostics displays a menu for the selected layer, with the following options:

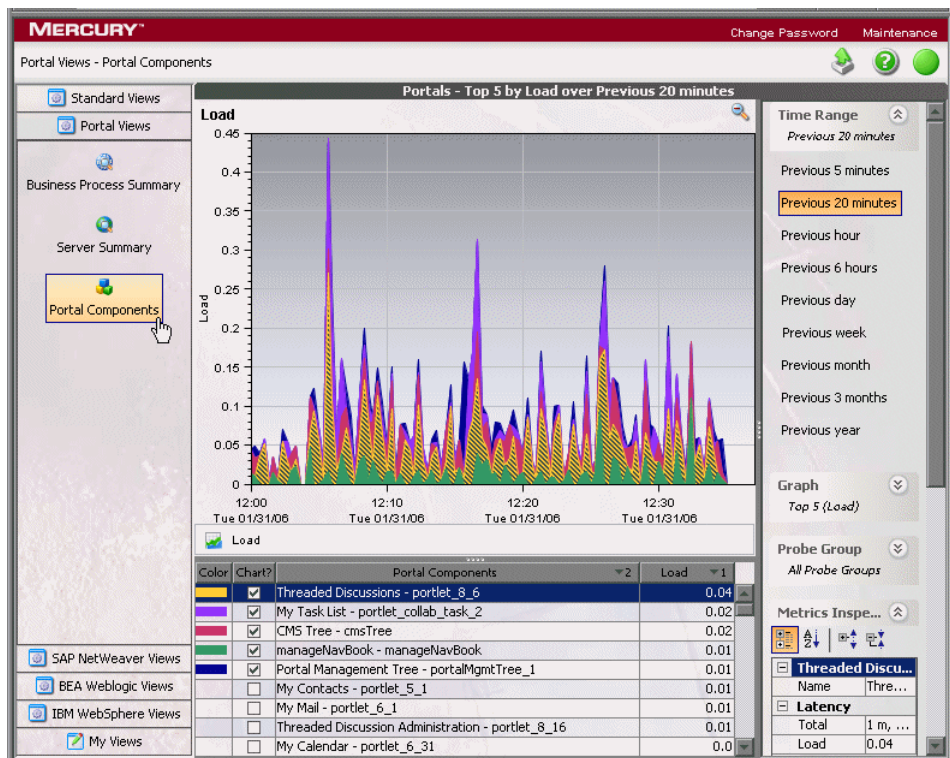
- ▶ **View Layers.** Diagnostics displays a breakdown of the latency contribution for each sub-layer of the selected layer. If there are no sub-layers defined for the selected layer, the **View Layers** menu option is disabled.
- ▶ **Open Mercury Diagnostics Profiler.** For information on the Mercury Diagnostics Profilers, see Part III, "Using the Mercury Diagnostics Profiler for J2EE" or Part IV, "Using the Mercury Diagnostics Profiler for .NET."

Using the Portal Components View

The Portal Components view displays the load performance metrics for the portal components that are being executed by your applications. This view has the format of a detail view. For information about the layout and controls in the detail views see Chapter 3, “Detail Views: Layout and Controls.”

Description of the Portal Components View

The following image is an example of the Portal Components view:



Graph

By default, the graph in the Portal Components view displays metrics for the top five portal components with respect to their load values.

By default, the x-axis of the graph shows actual chronological time in hours, minutes, and seconds (hh:mm:ss). The y-axis of the graph shows the calculated load values.

Detail Table

The detail table in the Portal Components view lists all of the portal components that pertain to the context shown in the bread crumbs displayed at the top of the view. If you navigated to the Portal Components view from the View bar, the Portal Components view shows all of the portal components. The metrics reported in the table are filtered based on the time period specified in the **Time Range** list and the probe group specified in the **Probe Group** list in the view filters.

Metrics Inspector

The **Metrics Inspector** in the Portal Components view lists the metrics for the selected row of the detail table.

Accessing the Portal Components View

You can access the Portal Components view from the **Portal Views** group on the View bar, and from a dashboard view that contains a monitoring version of the Portal Components view.

To access the Portal Components view using the View bar:

- 1 Open the **Portal Views** group on the View bar.
- 2 Click **Portal Components** in the **Portal Views** group.



The Portal Components view is displayed with the default settings so that the five portal components that have the highest load during the previous hour are charted on the graph.

To access the Portal Components view from a dashboard view:

- 1 Open a dashboard view that contains a monitoring version of the Portal Components view.
- 2 Double-click the monitoring version of the Portal Components view.

The Portal Components view is displayed with the same metrics that were displayed in the monitoring version. The bread crumb trail indicates that the Portal Components view was accessed from the dashboard view.

To access the Portal Components view from other detail views:

Instructions for drilling down to the Portal Components view for a particular transaction in the Transaction view can be found in “Drilling Down from a Transaction in the Detail Table” on page 130.

Customizing the Portal Components View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Portal Components view. For more information about the ways you can control how performance metrics are presented in this view, see “Detail Views: Layout and Controls” on page 37.

Interpreting the Portal Components View

Using the information displayed in the Portal Components view, you can get an immediate understanding of the performance of your application on the portal components that are being monitored. If the information displayed in the Portal Components view raises any concerns, you can drill down to find out more information from a more detailed view of the underlying performance metrics.

Displaying Portal Components Details from the Detail Table

You can view details for a portal component listed in the detail table by holding your mouse pointer over the portal component in the **Portal Components** column. The **Layer Details** tooltip is displayed as shown in the following example:

Color	Chart?	Portal Components	Load
	<input checked="" type="checkbox"/>	CMS Tree - cmsTree	0.1
	<input checked="" type="checkbox"/>	Content Management - cmsPortlet_1	0.1
	<input checked="" type="checkbox"/>	manageNavBook - manageNavBook	0
	<input checked="" type="checkbox"/>	manageContentBook - manageContentBook	0
	<input checked="" type="checkbox"/>	Main - main_1	0
	<input type="checkbox"/>	Threaded Discussions - portlet_8_6	0
	<input type="checkbox"/>	WS Tree - wsTree	0
	<input type="checkbox"/>	RSS News Feed - portlet1_2	0
	<input type="checkbox"/>	My Calendar - portlet_6_31	0

Layer Details

Layer: manageNavBook - manageNavBook

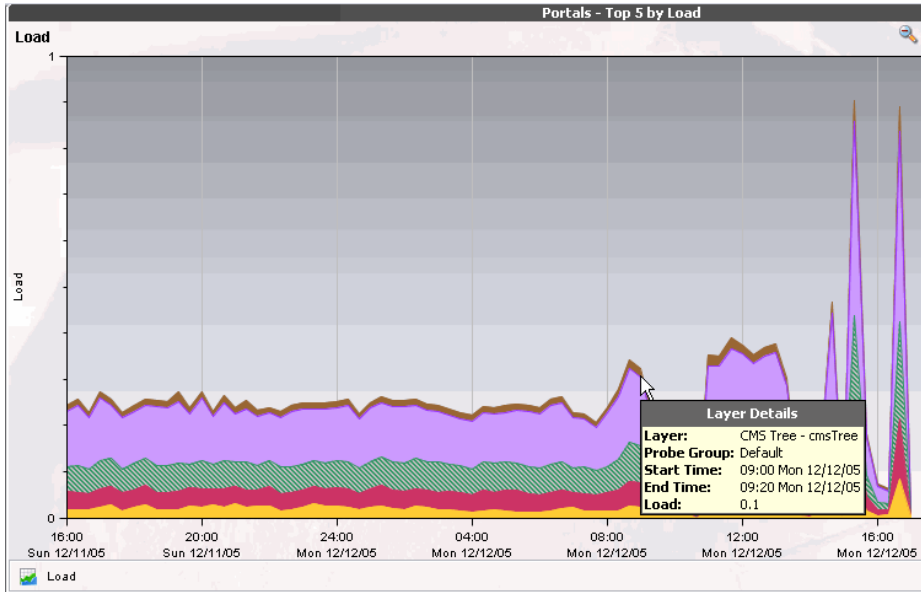
Probe Group: Default

The **Layer Details** tooltip displays the following information:

- **Layer.** The name of the layer that represents the portal component.
- **Probe Group.** The name of the probe group to which the portal component belongs.

Displaying Portal Components Details from the Charted Metrics

You can view more information about a portal component charted in the graph by holding the mouse pointer over the top edge of a stacked area for a charted metric until the **Layer Details** tooltip is displayed.



The **Layer Details** tooltip displays the following information:

- ▶ **Layer.** The name of the layer that represents the portal component.
- ▶ **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- ▶ **Start Time.** The start time for the aggregation period represented by the selected data point.
- ▶ **End Time.** The end time for the aggregation period represented by the selected data point.
- ▶ **Load.** The load calculated for the processing in this layer.

Drilling Down from a Portal Component in the Detail Table

You can drill down from a portal component listed in the detail table by double-clicking or right-clicking the portal component's row.

When you double-click a row in the detail table, Diagnostics displays the Life Cycle Methods view, with the methods that are being executed for the selected portal component. You can also display the Life Cycle Methods view by right clicking a row in the detail table and selecting **View Life Cycle Methods**.

6

Using the Status View

This chapter describes the Status view and how to use the controls and features of this view to analyze and understand the performance characteristics of your applications.

This chapter describes:	On page:
About the Predefined Status View	154
Accessing the Status View	155
Customizing the Status View	156
Interpreting the Status View	157
Drilling Down from the Status View	159

About the Predefined Status View

The *Status* view is a table of the probe groups that you defined when you installed the probes. There are two tables for each listed probe group. The first table is the *Probe* table, which lists all of the probes in the probe group, and the other table is the *Host* table, which lists the hosts for each of the probes in the probe group. The Status view presents a status for each probe group, for each probe in the probe table, and for each host in the Host table.

The following example of a Status view describes the features and controls of the view.

The screenshot shows a 'Status over Previous 5 minutes' view. It is a hierarchical table with the following structure:

- Probe Group Headers:**
 - JavaRealProbes (Status: S=1)
 - DotnetRealProbes (Status: S=1)
 - SyntheticProbe42 (Status: S=1)
- Probe Table Headers (under DotnetRealProbes):**
 - CallChain.NET (VM Heap Used: 94.8 MB, Latency: 1.5 s, Count/5: 1.9, Timeouts: 0, Exceptions: 0)
 - MSPetShop.NET (VM Heap Used: 94.8 MB, Latency: 48.6 ms, Count/5: 2.8, Timeouts: 0, Exceptions: 183)
- Host Table Headers (under DotnetRealProbes):**
 - ABSENT.mercury.co.il (CPU: 15%, Disk Bytes/5: 28.4 KB, Network Bytes/5: 275.4 KB)
- Probe Table Headers (under SyntheticProbe42):**
 - SyntheticProbe42 (VM Heap Used: 157.2 ms, Count/5: 17.6, Timeouts: 0, Exceptions: 0)
- Host Table Headers (under SyntheticProbe42):**
 - (Empty header row)

Labels in the image point to:

- Probe Group Status:** The 'S=1' indicator next to the probe group name.
- Probe Group Headers:** The top-level group names.
- Host Status:** The 'S=1' indicator next to the host name.
- Probe Status:** The 'S=1' indicator next to the probe name.
- Host:** The host name in the Host table.
- Probe Table Headers:** The headers for the probe table (VM Heap Used, Latency, Count/5, Timeouts, Exceptions).
- Host Table Headers:** The headers for the host table (CPU, Disk Bytes/5, Network Bytes/5).

Probe Group

The top level in the Status view table is the probe group. When you installed each probe, you assigned it to a probe group.

At the top of the Status view are the probe Group headers, which apply to each probe group listed in the view.

By default, the first column in each probe group entry in the table contains the probe group's status, which indicates how the performance of all probes in that probe group compares to the performance thresholds.

The Probe and Hosts tables for each probe group are listed below the name and status of the group.

Probe Table

The first table under the probe group entry is the Probe table. The Probe table has its own set of headers, and contains an entry for each probe in the probe group. By default, each entry in the Probe table contains the status of the probe, along with several metrics that help you understand the performance characteristics of that probe.

Host Table

The second table under the probe group entry is the Host table. The Host table has its own set of headers, and contains an entry for each host in the probe group. By default, each entry in the Host table contains the status of the host, along with several metrics that help you understand the performance characteristics of the host.

Accessing the Status View

You can access the Status view from the View bar or from any of the predefined dashboard views such as the Business Process Summary and the Server Summary.

To access the Status view from a dashboard view:

Open the dashboard view. The Status view appears as one of the views on the predefined dashboard view.

When the Status view appears on a dashboard view, the navigation and control features of the Status view work just as they do when it is displayed as a separate view.

To access the Status view using the View bar:



Click **Status View** in the **Standard Views** group on the View bar.

The Status view is displayed.

Customizing the Status View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Status view. For more information, see Chapter 3, “Detail Views: Layout and Controls.”

In addition to the standard Diagnostics view controls, there are controls unique to the Status view that you can use to customize the view of the performance metrics.

You can collapse a probe group entry to hide the tables of the group:

- ▣
 - by clicking the collapse button next to the name of the probe group.
 - by double-clicking the probe group’s name.

You can expand a probe group entry so show the tables of the group:

- ▣
 - by clicking the expand button for the probe group.
 - by double-clicking on the collapsed probe group.

Customizing the Status View Tables

The table headers in the Status view contain controls that allow you to specify which columns should appear in the tables, and the order in which they should appear. You can also specify which columns to use to sort the rows in the table. For more information on customizing the data display in the Status view, see “Using Table Header Controls” on page 32.

Interpreting the Status View

The Status view provides you with high-level performance metrics for the probes and probe hosts in each of the probe groups. Using the information displayed in the view, you can gain an immediate understanding of the performance of your applications. If the information displayed in the Status view raises any concerns, you can drill down to find out additional information from a more detailed view of the underlying performance metrics.

Note: The status is calculated on data received in the last five minutes only.

Investigating Alert Conditions



The **Status** column of the detail table contains a status indicator that tells you, at a glance, how the listed entity is doing compared to the thresholds that you set for its metrics and the metrics of its child entities. For information on setting thresholds, see “Setting Metric Thresholds” on page 53.

The color of the status indicator tells you the status of the entity. If you hold the mouse pointer over the status indicator in a row of the table, a tooltip is displayed to provide a verbal description of the statuses in the following list:

Status Indicator Color	Description
Green	Good - The component is performing within defined thresholds.
Yellow	Warning - The component is occasionally but not consistently exceeding defined thresholds. If the metric that has been exceeding the defined threshold is displayed in the detail table, the cell for the metric is outlined in yellow as well.
Red	Critical - The component is consistently exceeding defined thresholds. If the metric that has been exceeding the defined threshold is displayed in the detail table, the cell for the metric is outlined in red as well.
Grey	No status information available. Either no data has been received for the metric or no threshold has been set.

Displaying Probe Details

You can view the configuration summary of a probe listed in the Probe table by holding the mouse pointer over the probe's name in the **Probe** column until the **Probe Details** tooltip is displayed, as shown in the following example:

The screenshot shows a window titled "Status over Previous 5 minutes" with a "Probe Group" dropdown set to "JavaRealProbes". Below is a table of probes with columns for Status, Probe, VM Heap Used, Latency, Count/S, Timeouts, and Exceptions. A tooltip titled "Probe Details" is shown over the "AbsentPetstore" probe, displaying its configuration: Probe: AbsentPetstore, Probe Group: JavaRealProbes, Host: ABSENT.mercury.co.il, and Probe Type: Java. The tooltip also shows a summary table with columns for CPU, Disk Bytes/S, and Network Bytes/S.

Status	Probe	VM Heap Used	Latency	Count/S	Timeouts	Exceptions
●	AbsentPetstore	186.7 MB	3.3 ms	185.4	0	0
●	QConLakers	49.8 MB	15.3 ms	2.8	0	0
●	Raptor1Medrec	162.7 MB	386.3 ms	36.8	0	0
●	Raptor1Portal	303.8 MB	255.1 ms	37.5	0	573

CPU	Disk Bytes/S	Network Bytes/S
75%	60.7 KB	6.0 MB
14%	28.2 KB	274.9 KB
2%	13.4 KB	14.0 KB

The **Probe Details** tooltip displays the following information:

- **Probe.** The name of the selected probe.
- **Probe Group.** The name of the probe group to which the selected probe was assigned when it was installed.
- **Host.** The host name or IP address of the host for the probe. This host is also listed in the Host table in the same probe group.
- **Probe Type.** The type of probe.

Drilling Down from the Status View

From the Status view, you can drill down to the probes, hosts, and layers to view more detailed performance metrics.

Drilling Down to a Probe Group

You can drill down to a probe group listed in the Status view by right-clicking the probe group name. A menu opens with the following options:

- **View Probe Group Summary.** The Probe Group Summary is a dashboard view made up of detail views. For information on dashboard views, see “Introducing Diagnostics Dashboard Views” on page 29.
- **View Probes.** For information on the Probes view, see “Using the Probes View” on page 109.
- **View Hosts.** For information on the Hosts view, see “Using the Hosts View” on page 98.
- **View Load.** For information on the Load view, see “Using the Load View” on page 103.
- **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one.
- **Create/Edit/Delete Comment.** Create a comment, or edit or delete an existing one.

Drilling Down to a Probe

You can drill down to a probe listed in the Status view by double-clicking or right-clicking that probe's row.

When you double-click a row in the Probe table, Diagnostics displays the Probes view for the selected probe.

When you right-click a row in the Probe table, a menu opens for the selected probe with the following options:

- ▶ **View Probes.** Displays the activity of the probe for the last 5-minute period.
- ▶ **View Server Requests.** For information on the Server Requests view, see “Using the Server Requests View” on page 115.
- ▶ **View Probe Summary.** The Probe Summary is a dashboard view made up of detail views. For information on dashboard views, see “Introducing Diagnostics Dashboard Views” on page 29.
- ▶ **View Portal Components.** For information on the Portal Components view, see “Using the Probes View” on page 109
- ▶ **View Load.** For information on the Load view, see “Using the Load View” on page 103.
- ▶ **Open Mercury Diagnostics Profiler.** The appropriate Mercury Profiler opens. That is, if the probe is a .NET Probe the .NET Profiler opens; if the probe is a J2EE Probe the J2EE Profiler opens.

For information on the Mercury Diagnostics Profilers, see Part III, “Using the Mercury Diagnostics Profiler for J2EE” or Part IV, “Using the Mercury Diagnostics Profiler for .NET.”

- ▶ **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one.
- ▶ **Create/Edit/Delete Comment.** Create a comment, or edit or delete an existing one.

Drilling Down to a Host

You can drill down to a host listed in the Status view by double-clicking or right-clicking the host row.

When you double-click a row in the Host table, Diagnostics displays the Hosts view for the selected host. For more information on the Hosts view, See “Using the Hosts View” on page 98.

When you right-click a row in the Host table, a menu opens with the following:

- ▶ **View Hosts.** For information on the Hosts view, see “Using the Hosts View” on page 98.
- ▶ **View Probes.** For information on the Probes view, see “Using the Probes View” on page 109.
- ▶ **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one.
- ▶ **Create/Edit/Delete Comment.** Create a comment, or edit or delete an existing one.

Status Propagation

It is important to note the way in which status propagates through the Diagnostics application. For example, a green probe will turn red if any of its server requests turn red. Similarly, a probe group will turn red if any of its probes turn red. Hosts are separate entities, however, and will only turn red if their host metrics exceed defined thresholds.

7

Using the Specialized Views

Mercury Diagnostics displays unique view groups for selected types of monitored environments.

This chapter describes:	On page:
Using the Portal Views	163
Using the Web Services Views	167
Using the SAP Views	173
Using the Oracle Views	179
Using the BEA WebLogic Views	184
Using the IBM WebSphere Views	185
Using the CICS Views	187

Using the Portal Views

The Mercury Diagnostics Portals view group displays load performance metrics for portlets that are managed in the following portals:

- ▶ BEA WebLogic Portal Server WL 8.1 SP3, SP4
- ▶ WebSphere 5.1
- ▶ SAP Enterprise Portal

To access the Portal views:

Click **Portals** in the view bar (the left panel of the screen) and then click the Portal view that you want to display.

The default views contained in the Portals view group are described in the following sections:

- ▶ “Business Process Summary View (Mercury Business Availability Center or Standalone Mode),” below
- ▶ “LR / PC Summary View (LoadRunner or Performance Center)” on page 165
- ▶ “Server Summary View (Portals)” on page 165
- ▶ “Portal Components View” on page 166

Business Process Summary View (Mercury Business Availability Center or Standalone Mode)

If you are using Diagnostics with Mercury Business Availability Center or in standalone mode, the Portals view group contains the Business Process Summary view.

The Portals Business Process Summary view is a dashboard view that contains the following views:

- ▶ **Status.** A monitoring version of the standard Diagnostics Status view. For more information about the Status view, see Chapter 6, “Using the Status View.”
- ▶ **Transactions.** A monitoring version of the standard Diagnostics Transactions view. For more information about the Transactions view, see “Using the Transactions View” on page 126.
- ▶ **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 166.

LR / PC Summary View (LoadRunner or Performance Center)

If you are using Diagnostics with LoadRunner or Performance Center, the Portals view group contains the LR / PC Summary view.

The Portals LR / PC Summary view is a dashboard view that contains the following views:

- ▶ **Transactions.** A monitoring version of the standard Diagnostics Transactions view. For more information about the Transactions view, see “Using the Transactions View” on page 126.
- ▶ **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. For more information about the Server Requests view, see “Using the Server Requests View” on page 115.
- ▶ **Load.** A monitoring version of the standard Diagnostics Load view. For more information about the Load view, see “Using the Load View” on page 103.
- ▶ **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 166.

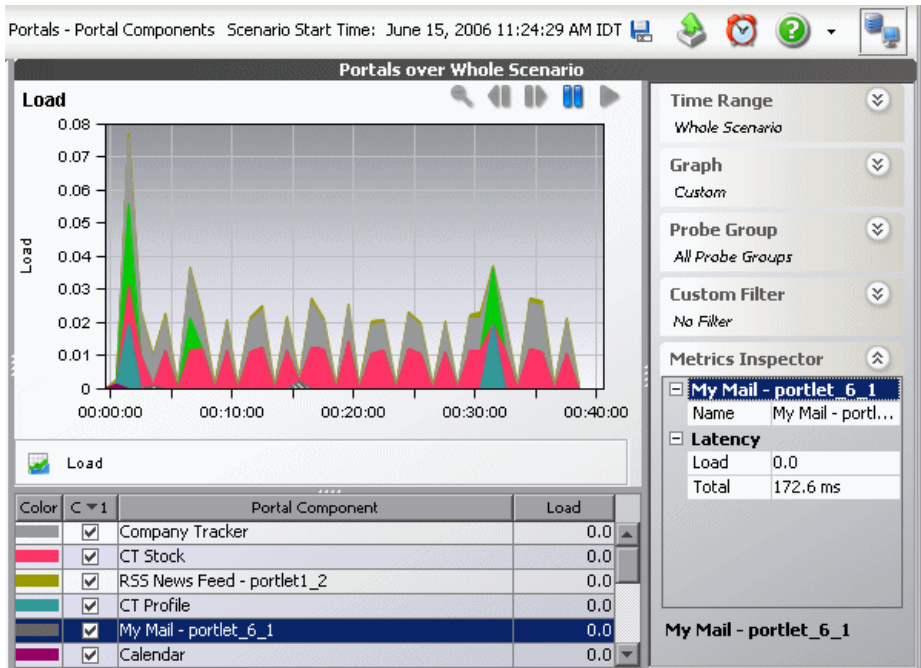
Server Summary View (Portals)

The Portals Server Summary view is a dashboard view that contains the following views:

- ▶ **Status.** A monitoring version of the standard Diagnostics Status view. For more information about the Status view, see Chapter 6, “Using the Status View.”
- ▶ **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. For more information about the Server Requests view, see “Using the Server Requests View” on page 115.
- ▶ **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View,” below.

Portal Components View

The Portal Components view displays load performance metrics for portlets (iViews in SAP) that are monitored by Diagnostics. In the following example of a screen in Diagnostics, the Portal Components view displays WebSphere and WebLogic portlets.



WebLogic portlets are displayed in the following format in the graph legend: <portlet name> - <Window label>. For example, My Mail - portlet_6_1.

WebSphere portlets and SAP iViews are displayed by their name in the graph legend.

From the portlets, you can drill down to the Life Cycle Methods view, which displays a breakdown of the load across the methods for that portlet.

Note: You can also drill down to the Portal Components view from the standard Probes view, by right clicking the relevant probe and selecting **View Portal Components**. In this case, the Portal Components view displays the specific portlets that are being monitored by that probe.

For more information about the Portal Components view, see “Using the Portal Components View” on page 146.

Using the Web Services Views

A Web service is a network component that provides services to other remote components, using the Internet for direct application-to-application interaction. A Web service is usually made up of a variety of different Web service operations that can be called by the client.

For example, a currency conversion Web service offers a variety of Web service operations, such as returning the exchange rate for a specific currency or returning the currency symbol based on the specified locale.

Mercury Diagnostics monitors *outbound* Web services calls made from within your monitored environment, and *inbound* Web services calls received and processed in your monitored environment.

For example, a client requests an exchange rate from a currency conversion Web service. From the client's point of view, that request is defined as an outbound call. The service provider receives the request, processes it, and sends a response (the exchange rate). From the service provider's point of view, the Web service request received from the client is defined as an inbound call.

Mercury Diagnostics displays information about outbound and inbound Web services calls in the Web Services view group.

Note: Diagnostics monitors only Web service operations invoked by HTTP/SOAP-based remote procedure calls (RPC).

To access the Web Services views:

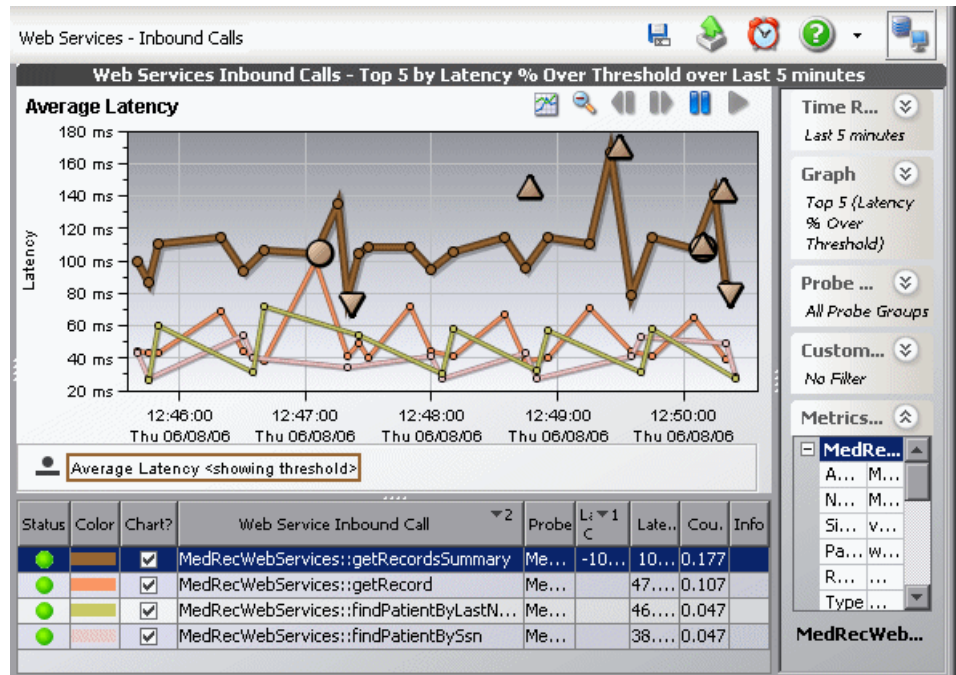
Click **Web Services** in the view bar (the left panel of the screen) and then click the Web Services view that you want to display.

The unique Mercury Diagnostics views for Web services are described in the following sections:

- “Inbound Calls,” below
- “Outbound Calls” on page 170
- “Instance Trees Displaying Web Services Correlations” on page 171

Inbound Calls

The Web Services Inbound Calls view displays information about the inbound Web services calls received and processed in your monitored environment.



Web service calls are displayed in the following format in the graph legend: **<Web-service-name>::<operation-name>**. For example, **CurrencyConversionService::ConvertToNum**.

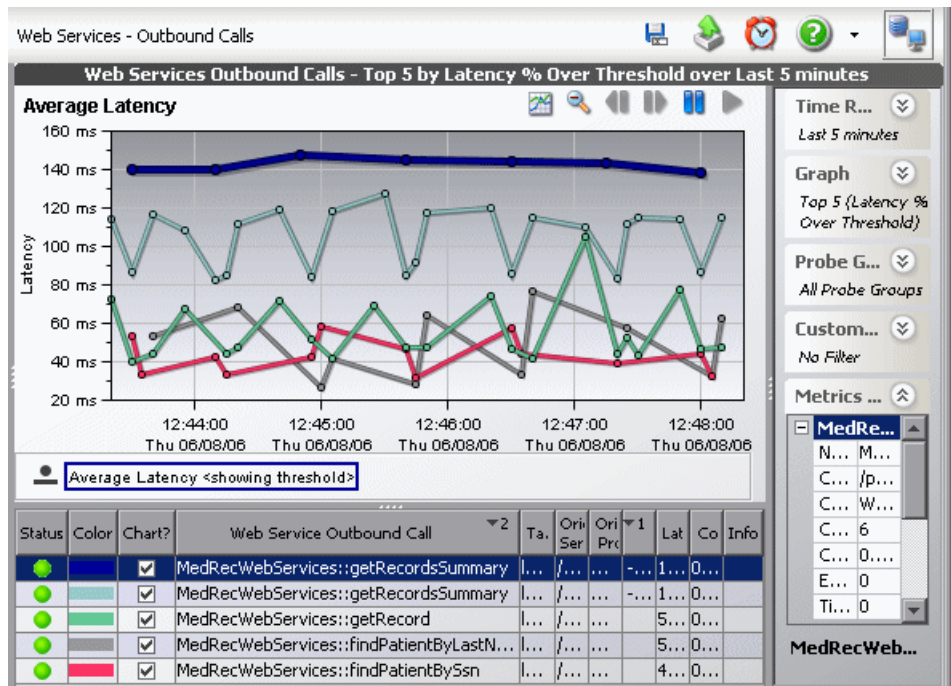
In Diagnostics, inbound Web services calls are displayed as a type of server request. You can drill down to instance trees that display the method calls and their latency for a particular instance of a server request. For more information, see “Drilling Down to an Instance Tree for a Server Request” on page 121.

You can also drill down to the sublayers of each call by right-clicking the inbound call in the graph legend and selecting **View Layers**.

Sometimes, an inbound Web services call can trigger external outbound Web services calls to another Web service. You can drill down to these outbound calls by right-clicking the inbound call in the graph legend and selecting **View Outbound Calls**.

Outbound Calls

The Web Services Outbound Calls view displays information about the outbound Web services calls made from within your monitored environment.



In Diagnostics, outbound Web services calls are displayed as remote calls within a server request.

The Outbound Calls graph legend includes the following columns:

- ▶ **Web Service Outbound call.** The name of the Web service that is being called. It is displayed in the following format: **<Web-service-name>::<operation-name>**. For example, **CurrencyConversionService::ConvertToNum**.
- ▶ **Target.** The hostname and port number (displayed as **<host:port>**) of the the Web service that is being called.
- ▶ **Originating Server Request.** The server request from which this Web Service was called.
- ▶ **Originating Probe.** The probe from which this Web Service was called.

You can drill down to the original server request displayed in the Server Requests view by right-clicking the outbound call in the graph legend and selecting **View Server Requests**.

Instance Trees Displaying Web Services Correlations

If both the outbound and inbound Web services calls for a particular Web service are monitored in your environment, you can view the correlation between outbound and inbound calls in an instance tree in the Call Profile view.

You can view Web service correlations across multiple VMs and across different platforms, such as J2EE and .NET.

To drill down to instance trees displaying Web services correlations:



Click the red X on the Server Requests view containing the outbound Web services call. The Call Profile view opens, displaying a correlation instance tree.

In the following example of an instance tree in the Call Profile view, the service request, displayed in red, contains an outbound Web services call. The inbound call for the same Web service is displayed in blue. The inbound Web service triggers a call to another external Web service, displayed in green.

The screenshot shows the Mercury Diagnostics interface. The breadcrumb navigation indicates the path: **Web Services** > **- Outbound Calls (Service::Sell)** > **Server Requests (/RemoteTraderClient/Default.aspx)** > **Call Profile**. The main window title is **Call Profile [Instance of /RemoteTraderClient/Default.aspx ending at 6/8/06 10:42:33 AM o...]**. The call tree consists of the following layers (from top to bottom):

- Red:** /RemoteTraderClient/Default.aspx
- Red:** HttpRuntime.ProcessRequestNow()
- Red:** Service.Buy()
- Red:** Remote Call to Service::Buy on probe RemoteTrader.NET on nectar.mercury.co.il
- Blue:** HttpRuntime.ProcessRequestNow()
- Blue:** Void Service.Buy()
- Blue:** TraderService.buy()
- Blue:** Remote Call to TraderService:buy on probe GreenPerfume on perfume.mercury.co.il
- Green:** WebServiceServlet.doPost()
- Green:** ServletBase.doPost()
- Green:** Dispatcher.process()
- Green:** TraderBean.buy()

At the bottom, a table lists the call details:

Call	Total Latency
100% /RemoteTraderClient/Default.aspx	3,031.1
100% HttpRuntime.ProcessRequestNow()	3,030.4
99.5% Service.Buy()	3,015.7
99.3% Remote Call to Service::Buy on probe	3,009.6

On the right side, a properties window for **System.Web.Http...** is visible, showing fields for Application, Arguments, Name, Signature, and Package.

Using the SAP Views

Mercury Diagnostics displays two different types of SAP performance data in the SAP view group:

- ▶ **SAP R/3.** Represents the ABAP stack of the SAP system.

When you install and configure the Diagnostics Collector to gather data from a SAP R/3 system, you define instances of SAP R/3 to be monitored. Each one of these instances is represented as an SAP R/3 Probe, belonging to a probe group, in the Mercury Diagnostics UI.

Mercury Diagnostics displays SAP R/3 performance data and metrics in the SAP view group.

- ▶ **NetWeaver.** Represents the SAP Web Application Server (WAS) Java stack.

The NetWeaver data is collected by the Mercury Diagnostics Probe for J2EE. Mercury Diagnostics displays SAP NetWeaver performance data and metrics in the SAP view group.

To access the SAP views:

Click **SAP** in the view bar (the left panel of the screen) and then click the SAP view that you want to display.

The default views contained in the SAP view group, are described in the following sections:

- ▶ “NetWeaver Summary,” below
- ▶ “R3 Summary” on page 175
- ▶ “NetWeaver Requests” on page 176
- ▶ “NetWeaver Threads” on page 177
- ▶ “R3 Probes” on page 178
- ▶ “R3 Server Requests” on page 179

NetWeaver Summary

The NetWeaver Summary view is a dashboard view that contains a summary of the NetWeaver data displayed by Diagnostics.

The NetWeaver Summary dashboard view contains the following views:

- ▶ **Status.** A monitoring version of the standard Diagnostics Status view. All Diagnostics probe groups, probes, and hosts are displayed in this view, and not only those that are SAP-related. For more information about the Status view, see Chapter 6, “Using the Status View.”
- ▶ **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. All Diagnostics server requests are displayed in this view, and not only those that are SAP-related. For more information about the Server Requests view, see “Using the Server Requests View” on page 115.
- ▶ **Portals.** A monitoring version of the standard Diagnostics Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 166.
- ▶ **Probes.** A monitoring version of the standard Diagnostics Probes view. All Diagnostics probes are displayed in this view, and not only those that are SAP-related. For more information about the Probes view, see “Using the Probes View” on page 109.
- ▶ **NetWeaver Threads.** A monitoring version of the NetWeaver Threads view. For more information, see “NetWeaver Threads” on page 177.
- ▶ **NetWeaver Requests.** monitoring version of the NetWeaver Requests view. For more information, see “NetWeaver Requests” on page 176.

R3 Summary

The R3 Summary view is a dashboard view that contains a summary of the SAP R/3 data displayed by Diagnostics.

The R3 Summary dashboard view contains the following views:

- ▶ **Status.** A monitoring version of the standard Diagnostics Status view. All Diagnostics probe groups, probes, and hosts are displayed in this view, and not only those that are SAP-related. For more information about the Status view, see Chapter 6, “Using the Status View.”

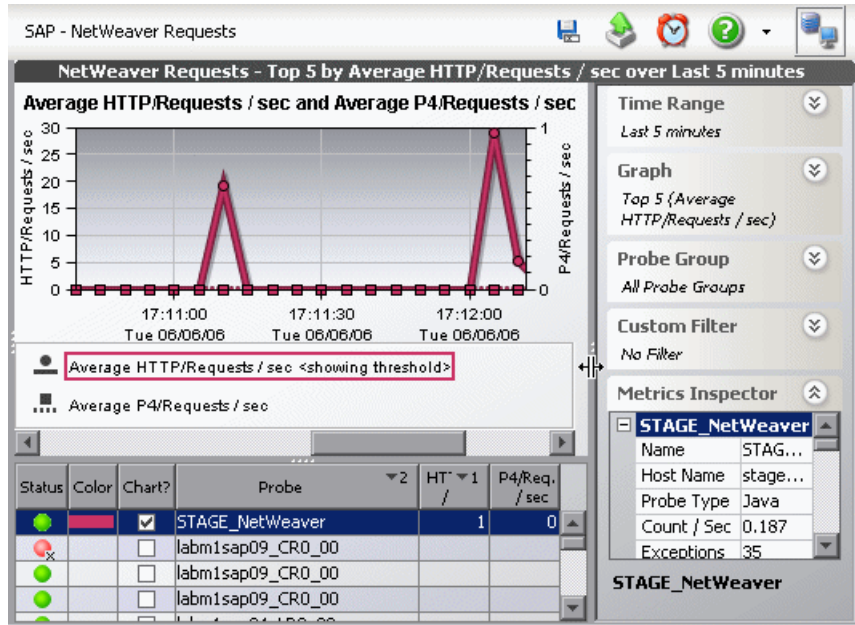
The only metrics in the table that are relevant for the SAP R/3 Probes are the **Latency** and **Count** metrics.

For the SAP R/3 hosts, the table displays CPU metrics. You can view unique performance metrics for the SAP R/3 hosts by drilling down to the Diagnostics Hosts view.

- ▶ **SAP R/3 Probes.** A monitoring version of the R3 Probes view. For more information, see “R3 Probes” on page 178.
- ▶ **SAP R/3 Server Requests.** A monitoring version of the R3 Server Requests view. For more information, see “R3 Server Requests” on page 179.

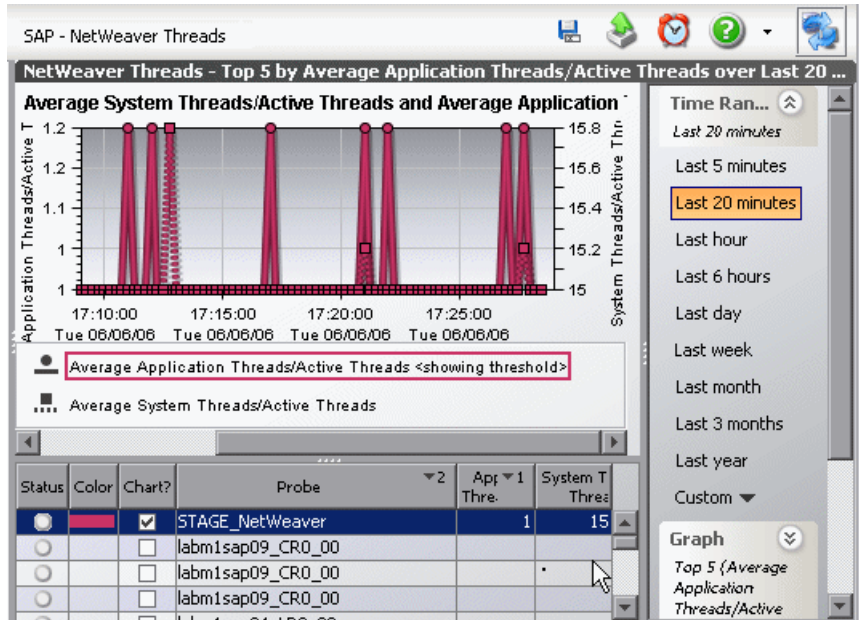
NetWeaver Requests

The NetWeaver Requests view displays the average HTTP and P4 requests per second running on the SAP Web Application Server Java stack.



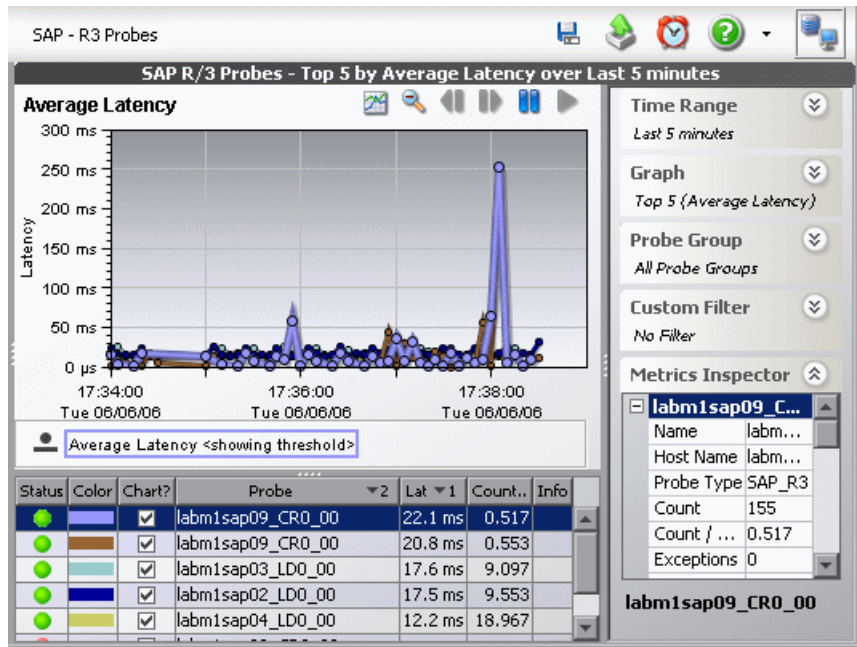
NetWeaver Threads

The NetWeaver Threads view displays information about the active threads running on the SAP Web Application Server Java stack.



R3 Probes

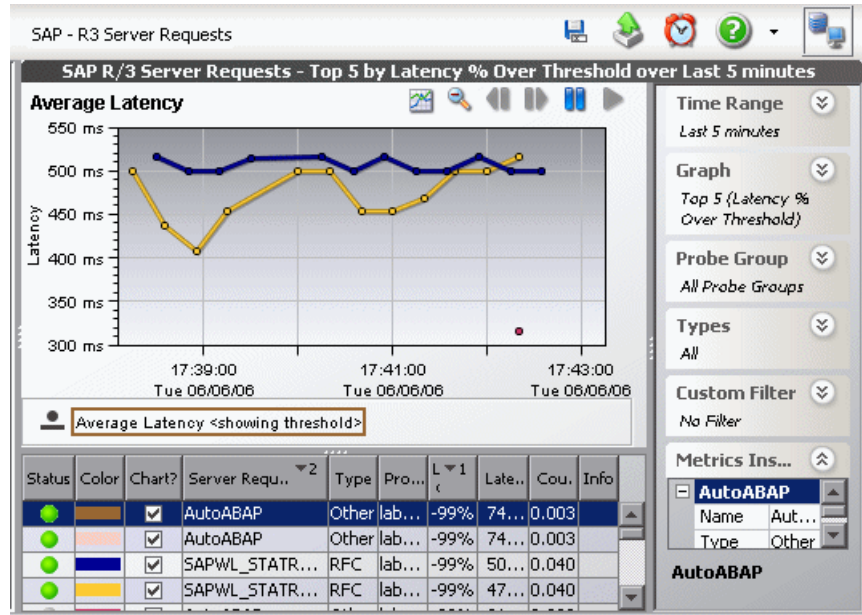
The R3 Probes view displays unique metrics for the SAP R/3 instances.



The default metric displayed in the view for SAP R/3 Probes is **Average Latency**.

R3 Server Requests

The R3 Server Requests view displays information about the server requests running on the ABAP stack of the SAP system.



Using the Oracle Views

When you install and configure the Diagnostics Collector to gather data from an Oracle 10g Database, you define instances of Oracle 10g to be monitored. Each one of these instances is represented as an Oracle Probe, belonging to a probe group, in the Mercury Diagnostics UI.

Mercury Diagnostics displays Oracle 10g performance data and metrics in the Oracle view group.

To access the Oracle views:

Click **Oracle** in the view bar (the left panel of the screen) and then click the Oracle view that you want to display.

The default views contained in the Oracle view group are described in the following sections:

- “Oracle Summary View,” below
- “Oracle Probes View” on page 181
- “Oracle Wait Time View” on page 183

Oracle Summary View

The Oracle Summary view is a dashboard view that contains a summary of the Oracle 10g data displayed by Diagnostics.

The Summary dashboard view contains the following views:

- **Status view.** A table displaying the Oracle probes contained in each probe group and the hosts for those probes.



The status indicator in the status column indicates how each component is performing based on the thresholds set for that component. There are no other metrics in the table that are relevant for the Oracle Probes.

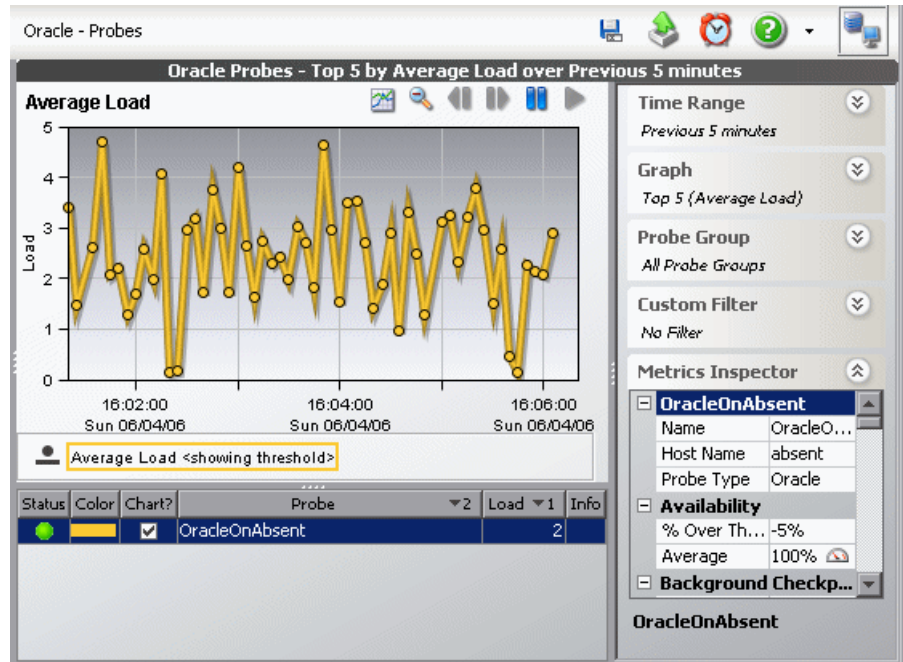
For the Oracle hosts, the table displays CPU metrics. You can view unique performance metrics for the Oracle hosts by drilling down to the Diagnostics Hosts view.

Note: All Diagnostics probe groups are displayed in this view, but within each probe group, only Oracle Probes are displayed. Host information for other non-Oracle probes also appears in this view.

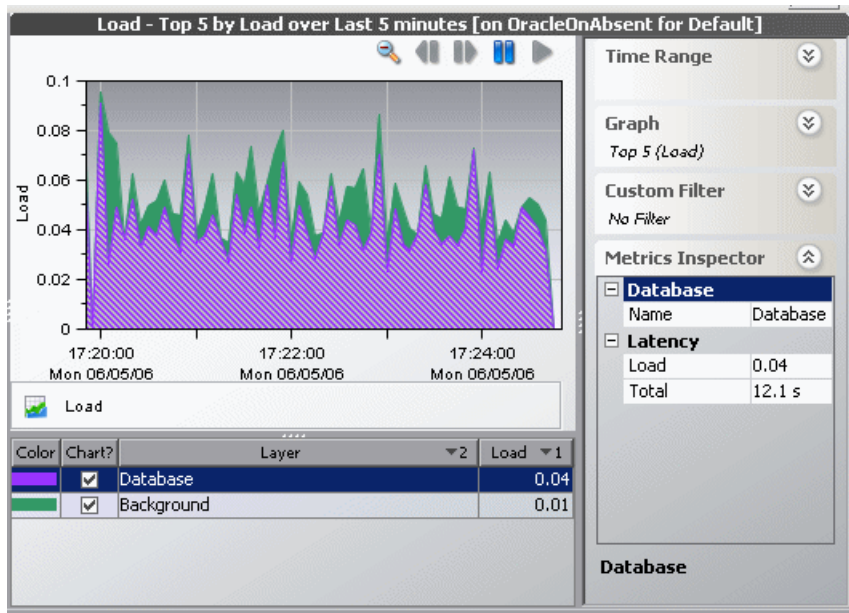
- **Oracle Probes.** A monitoring version of the Oracle Probe view. For more information, see “Oracle Probes View,” below.
- **Wait Time.** A monitoring version of the Oracle Wait Time view. For more information, see “Oracle Wait Time View” on page 183.

Oracle Probes View

The Oracle Probes view displays unique metrics for the Oracle 10g Database instances.



From the Oracle Probe view, you can drill down further to the Load view, by double-clicking the probe in the graph legend.

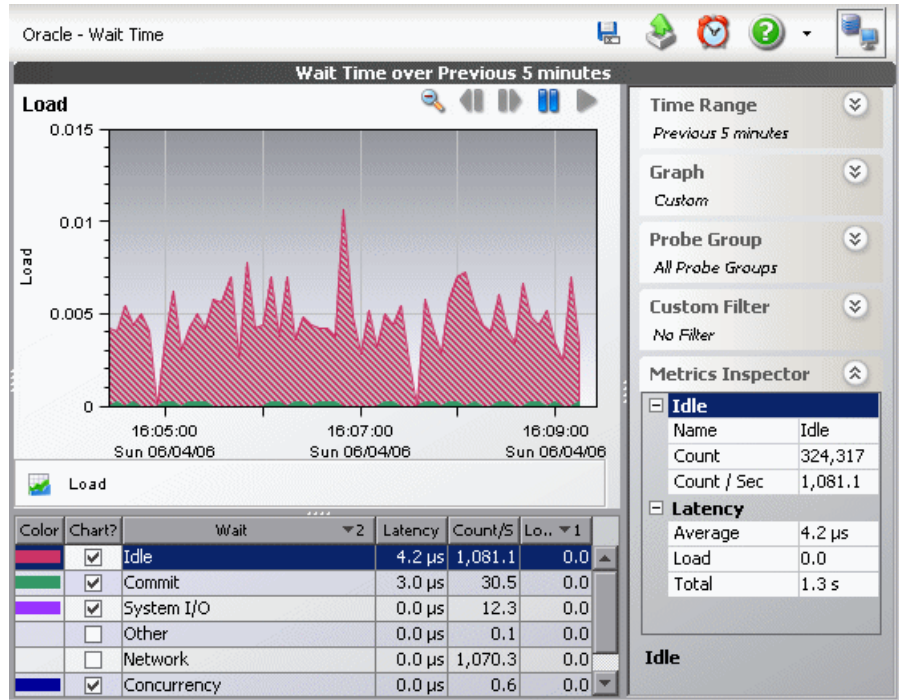


The layers in the Oracle Probe Load view represent the actions that take place on the specific Oracle 10g instance. Layer names and calculations in this view are based on the Oracle 10g time model.

You can drill down to more layers, by double-clicking the **Database** layer in the Load graph legend.

Oracle Wait Time View

The Oracle Wait Time view displays *wait event classes* for the Oracle 10g instances.



Wait event classes represent specific types of events that Oracle has to wait for to continue processing. You can drill down to further wait event statistics by double-clicking one of the wait events classes in the graph legend.

Using the BEA WebLogic Views

Mercury Diagnostics displays BEA WebLogic application server metrics in the BEA WebLogic view groups.

To access the BEA WebLogic views:

Click **BEA WebLogic** in the view bar (the left panel of the screen) and then click the WebLogic view that you want to display.

The BEA WebLogic view group contains the following views:

BEA WebLogic Summary View

The Summary view is a dashboard view containing the standard Diagnostics Status view along with monitoring versions of the standard Diagnostics Server Requests view and all views displayed in the BEA WebLogic view group.

For more information about the BEA WebLogic views, see the descriptions that follow, in this section. For more information about the Status view, see Chapter 6, “Using the Status View.” For more information about the Server Requests view, see “Using the Server Requests View” on page 115.

EJB Pooled Resource Contention

This view displays the following default metrics:

- ▶ **EJB-Pool Current Waiters.** Current number of threads that have waited for a lock on a bean.
- ▶ **EJB-Pool Get Timeouts.** Current number of threads that have timed out waiting for a lock on a bean.

JDBC Connection Status

This view displays the following default metrics:

- ▶ **JDBC Active Connections.** Current total number of active connections.
- ▶ **JDBC Current Capacity.** Current capacity of this connection pool.

JDBC Resource Contention

This view displays the following default metrics:

- ▶ **JDBC Requests Waiting for Connection.** Current total number waiting for a connection.
- ▶ **JDBC Wait Seconds High.** Number of seconds the longest waiter for a connection waited.

Server Threads

This view displays the following default metrics:

- ▶ **Execute Queues Requests.** Number of requests, which have been processed by this queue.
- ▶ **Execute Queues Pending Requests.** Number of requests waiting in the server's default execute queue.

Using the IBM WebSphere Views

Mercury Diagnostics displays IBM WebSphere application server metrics in the IBM WebSphere view groups.

To access the IBM WebSphere views:

Click **IBM WebSphere** in the view bar (the left panel of the screen) and then click the WebSphere view that you want to display.

The IBM WebSphere view group contains the following views:

IBM WebSphere Summary View

The Summary view is a dashboard view containing the standard Diagnostics Status view along with monitoring versions of the standard Diagnostics Server Requests view and selected views displayed in the IBM WebSphere view group.

For more information about the IBM WebSphere views, see the descriptions that follow, in this section. For more information about the Status view, see Chapter 6, “Using the Status View.” For more information about the Server Requests view, see “Using the Server Requests View” on page 115.

JDBC Connection Status

This view displays the following default metrics:

- **JDBC Free Pool Size.** Number of free connections in the pool.
- **JDBC Connections in Use.** Number of connection objects in use for a particular connection pool (applicable to 5.0 DataSource only).

JDBC Resource Contention

This view displays the following default metrics:

- **JDBC Concurrent Waiters.** Average number of threads that are concurrently waiting for a connection.
- **JDBC Avg. Wait Time.** Average waiting time in milliseconds until a connection is granted.

MQ Connection Stats

This view displays the following default metrics:

- **JMS Connection Use Time.** Average use time of the connection to the JMS server.
- **JMS Connection Wait Time.** Average wait time of the connection to the JMS server.

MQ Message Driven Bean Stats

This view displays the following default metrics:

- **MDB Message Session Wait Time.** Average time to obtain a ServerSession from the pool (message-driven beans).
- **MDB Message Count.** Number of messages delivered to the bean on Message method (message-driven beans).

MQ Queue Stats

The default metric displayed in this view is **Depth of all Queues**. This is the number of messages currently on the QueuePoint.

Server Threads

This view displays the following default metrics:

- ▶ **Threads Percent Maxed.** Average percent of the time that all threads are in use.
- ▶ **Threads Pool Size.** Average number of threads in pool.

Servlet Session Management

This view displays the following default metrics:

- ▶ **SM Session Life Time.** Average session lifetime in milliseconds.
- ▶ **SM Invalidated Via Timeout.** Number of sessions that were invalidated by timeout.

Using the CICS Views

CICS (Customer Information Control System) is a transaction server that runs primarily on IBM mainframe systems under z/OS.

CICS Transaction Gateway is a J2EE connector that handles communication between a WebSphere application server and a CICS system.

Mercury Diagnostics monitors communication that takes place between the WebSphere application Server and the CICS Transaction Gateway, and displays these metrics in the CICS view groups.

To access the CICS views:

Click **CICS** in the view bar (the left panel of the screen) and then click the CICS view that you want to display.

The CICS view group contains the following views:

CICS Summary

The Summary view is a dashboard view containing monitoring versions of the standard Diagnostics Server Requests view and all the views displayed in the CICS view group.

For more information about the CICS views, see the descriptions that follow, in this section. For more information about the Server Requests view, see “Using the Server Requests View” on page 115.

J2C Connection Pool

The default metric displayed in this view is **J2C Pool Size**. This is the average number of managed connections in the pool.

J2C Connections and Faults

The default metric displayed in this view is **J2C Connections Allocated**. This is the total number of times that a managed connection is allocated to a client (the total is maintained across the pool, not per connection).

J2C Load

The J2C Load view displays a breakdown of the load across the different J2C classes.

J2C Usage Time

The default metric displayed in this view is **J2C Usage**. This is the average time in milliseconds that connections are in use.

J2C Wait Time

The default metric displayed in this view is **J2C Wait Time**. This is the average waiting time in milliseconds until a connection is granted.

8

Customizing Diagnostics Views

This chapter provides instructions for creating, saving, and organizing custom Diagnostics views.

This chapter describes:	On page:
About Mercury Diagnostics Custom Views	190
Understanding Diagnostics Custom View Retention	190
Creating View Groups	191
Saving a Customized View	192
Creating a New View	194
Renaming a View	199
Deleting a View	200
Modifying a Custom View	201
Sharing Custom Views	201
Upgrade of Custom Views From Previous Versions	202

About Mercury Diagnostics Custom Views

Diagnostics provides many different ways for you to customize the views so that you can filter and focus the information displayed as you analyze performance issues. For information on how to customize the information displayed in the views, see Chapter 2, “Introducing Diagnostics Views.”

When you have customized a view, you may want to save the changes so that you can reuse the custom view when you use Diagnostics in the future. The controls in the View bar enable you to save, retrieve, and revise your customized views. Instructions for using the View bar controls to maintain your custom views are provided in the following sections.

Understanding Diagnostics Custom View Retention

When you make modifications to the way information is displayed in a detail view or a dashboard view, Diagnostics retains the customizations in different ways depending on whether the view was a custom view or a standard view and depending on how you navigated to the view.

Customized Default Views

When you customize a view that you selected from a view group that was installed with the product, the changes are retained and displayed each time you access the view as long as you do not shut down Diagnostics completely. Once you shut down Diagnostics, the view is displayed in its default configuration the next time it is accessed.

Customized Drilldown Views

When you customize a view that you accessed by drilling down from another view, your customizations are retained only as long as you navigate within the bread crumbs. If you drill down on the same path again, the customizations will be retained as long as you did not navigate to another path in the meantime.

Customization of Custom Views

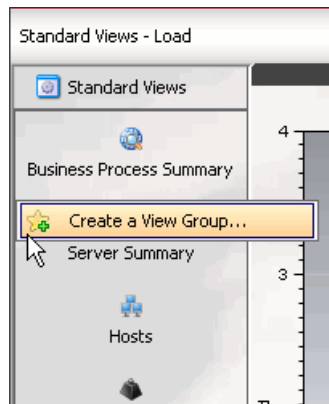
When you customize a view that you selected from a view group that you created, the customizations are retained when you access the view again, regardless of whether the Diagnostics has been shut down. Customizations to your custom views are automatically saved.

Creating View Groups

Custom views must be saved in a custom view group, such as **My Views**, which is installed with the product, or in a custom view group that you created. Custom views cannot be stored in the Standard views, Portals, SAP NetWeaver, BEA WebLogic, or IBM WebSphere view groups installed with the product.

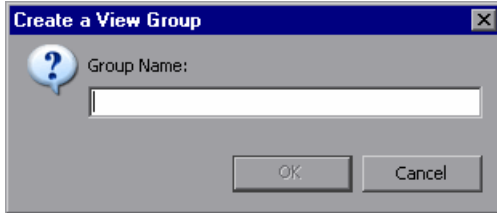
To create a view group:

- 1 Right-click the View bar inside any view group, and select **Create a View Group**.



Note: Be sure to hold the pointer over the View bar, and not over the icon for a view.

The Create a View Group dialog box opens.



- 2 Type a name for the view group, and click **OK**.

The new view group is created and opens in the View bar.

Saving a Customized View

As you work with a view, you may find that the settings you made to customize the view have been particularly useful in helping you to understand an aspect of your application's performance. You can save the customized view exactly as it is displayed. Diagnostics saved the view in one of your custom view groups so that you can use it to analyze the performance of your application in the future. You can save customized versions of views that you opened from the Standard Views group, and you can save new versions of customized views that you previously stored in one of your view groups.

Note: If you are using Diagnostics with either Mercury Business Availability Center or Performance Center, the customized views are saved for the Mercury Business Availability Center or Performance Center user who created the view.

If you are using Diagnostics with LoadRunner, the views are stored on the Diagnostics Server for user **admin**.

There are two ways to save a custom view once you have configured the view in the desired manner: using the pop-up menu in a custom view group and using the Save this View button in the toolbar.

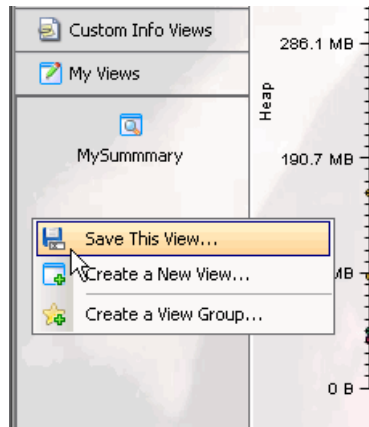
To save a view using the pop-up menu in a custom view group:

- 1 Open the custom view group where you want to store the saved view.

Note: Custom views can only be saved in the view group, My Views, or in a view group that you have created.

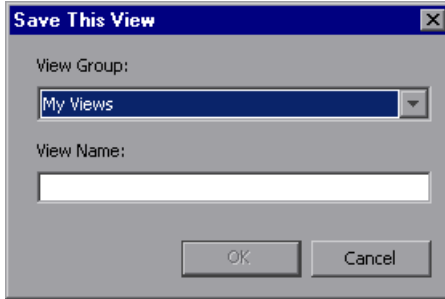


- 2 Right-click the View bar inside the custom view group, and select **Save This View**, or click the **Save This View** button.



Note: Be sure to hold the pointer over the View bar, and not over the icon for a view.

The Save This View dialog box opens.



- 3** The **View Group** list lists all the view groups that are eligible to contain custom views.
 - a** Select a view group from the list.
 - b** Type a name for the view, and click **OK**.

The view that is currently displayed is saved, and an icon for the new view is displayed in the custom view group.

Creating a New View

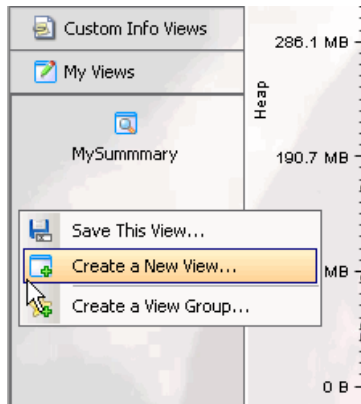
You may assemble various combinations of one or more existing views to create a completely new view. By creating a new view, you can put the standard views and custom views that you use regularly together in one view, allowing you to see the performance metrics side-by-side and to drill down to the underlying metrics just as you can in any other view in Diagnostics.

To create a view:

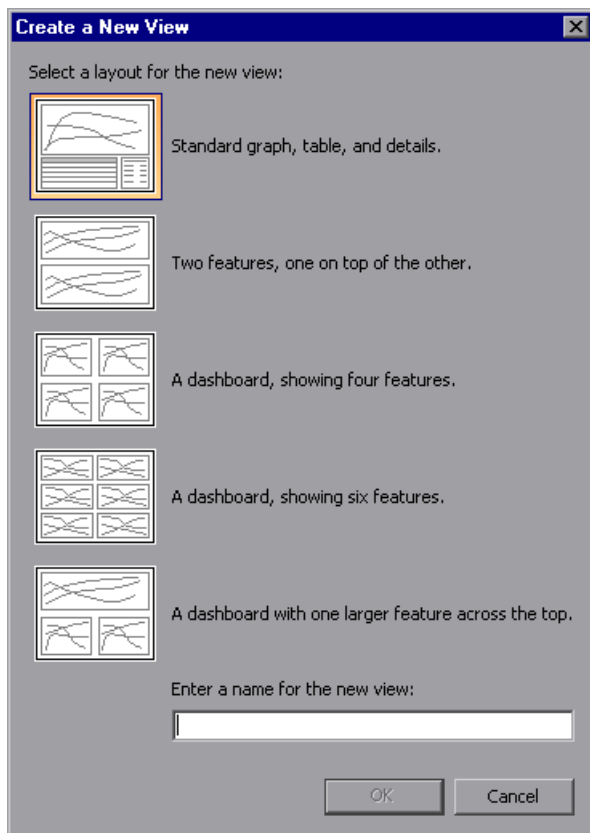
- 1 Click the custom view group in the view bar where you want to create a new view.

Note: Custom views can only be created in the view group, My Views, or in a view group that you have created.

- 2 Right-click the View bar inside the custom view group, and select **Create a New View**.

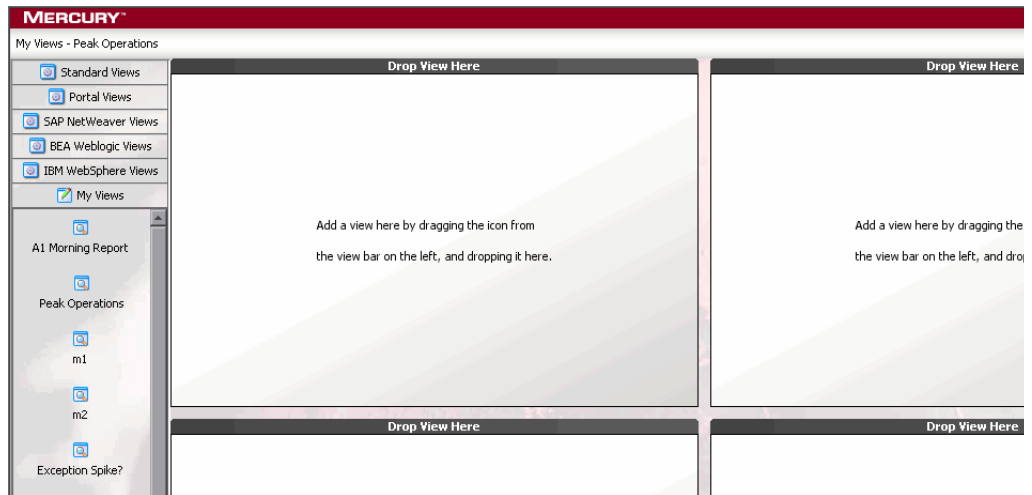


The Create a New View dialog box opens.



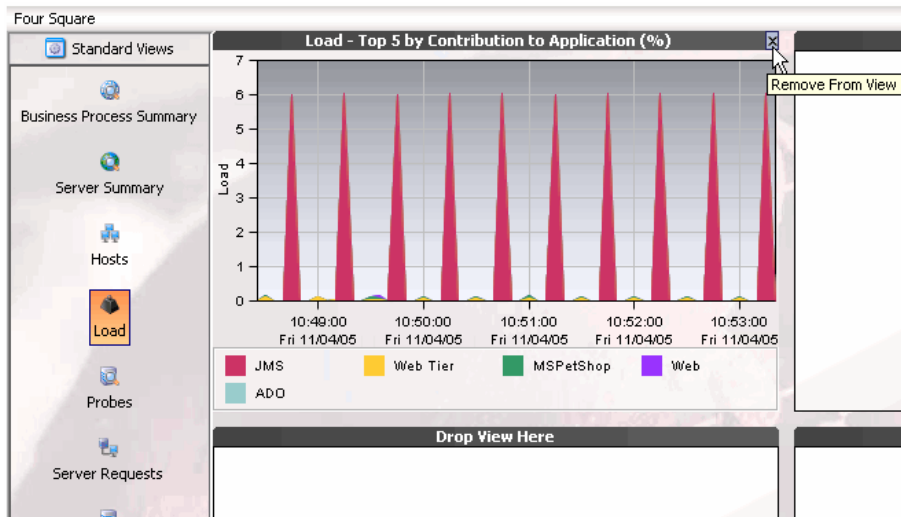
- 3 Click the icon for the desired layout of the new view. The selected icon is highlighted. Enter a name for the new view, and click **OK**.

An icon for the new view is added to the view group, and the selected view layout is displayed on your screen with **Drop View Here** placeholders for each of the views that make up the new view. The example below shows the view layout for the four-feature dashboard.



- 4 Select the views that you want to appear in the custom view group.
 - To add a view, select it from the view groups in the View bar, and drag it into the **Drop View Here** placeholders.
 - To remove a view, click the **Remove from View** button in the upper right corner of the view, as shown below.

The view is deleted, and the empty **Drop View Here** placeholder is displayed once more.



The changes that you make to the new view are automatically saved as you make them.

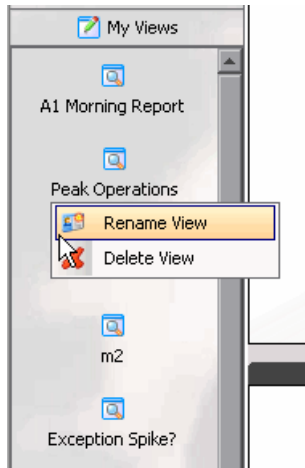
Renaming a View

You can rename a view that you have saved in a custom view group.

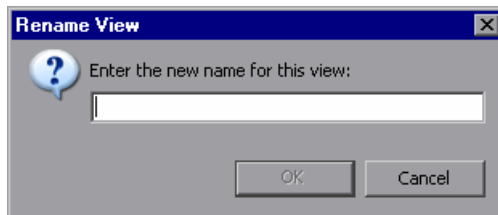
Note: You can only rename views in the view group My Views or in a view group that you have created.

To rename a view:

- 1 Click the view group in the View bar that contains the view that you want to rename.
- 2 Right-click the icon for the view that you want to rename and select **Rename View**.



The Rename View dialog box opens.



- 3 Enter a new name for the view, and click **OK**.

The new name for the view appears under its icon in the View bar.

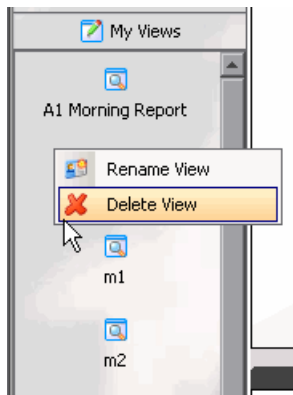
Deleting a View

You can delete a view that you have saved in a custom view group.

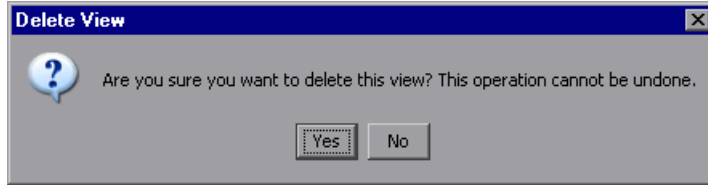
Note: You can only delete views from the view group My Views or in a view group that you have created.

To delete a view:

- 1 Click the view group in the View bar that contains the view that you want to delete.
- 2 Right-click the icon for the view that you want to delete, and select **Delete View**.



Diagnostics displays the Delete View warning message.



- 3 Click **Yes** to delete the view.

The selected view is deleted and no longer appears in the view group.

If the view that you deleted was displayed on the active Diagnostics screen, then the view is replaced with the default view that is displayed when Diagnostics opens.

Modifying a Custom View

When you are working with a custom view that you have saved, any changes you make to the view are automatically saved to the custom view. The next time you open the custom view, the changes you made the last time you opened the view are used to display the performance metrics.

Sharing Custom Views

Diagnostics provides a way for you to share the custom views that you have created, with other users.

When you create a custom view, the view is stored on the Diagnostics Server in Commander mode in the directory `<diagnostics_server_install_dir>/storage/userdata/<customer name>/<user name>`. Views are stored as XML files, and are named based on the view group and the view name. For example, the Employee Benefits Overview view, in the Employee Application view group, is stored as **Employee Application - Employee Benefits Overview.xml**.

To share a view with another user, copy the XML file to that user's directory.

Note: The two views are completely independent, and changes to one will not be reflected in the other.

Upgrade of Custom Views From Previous Versions

When you open the custom views that were created in Diagnostics 4.0 or 4.1 for the first time in Diagnostics 4.2, Diagnostics upgrades the view for any changes that are necessary because of changes to the functionality of Diagnostics. When Diagnostics changes your custom views it issues a message to let you know that your custom view has been modified.

For instructions on upgrading the Diagnostics data when moving from Diagnostics 4.0 or 4.1 to 4.2, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

For instructions on using individual custom views from a backup or from another user see “Sharing Custom Views” on page 201.

9

Instance Trees

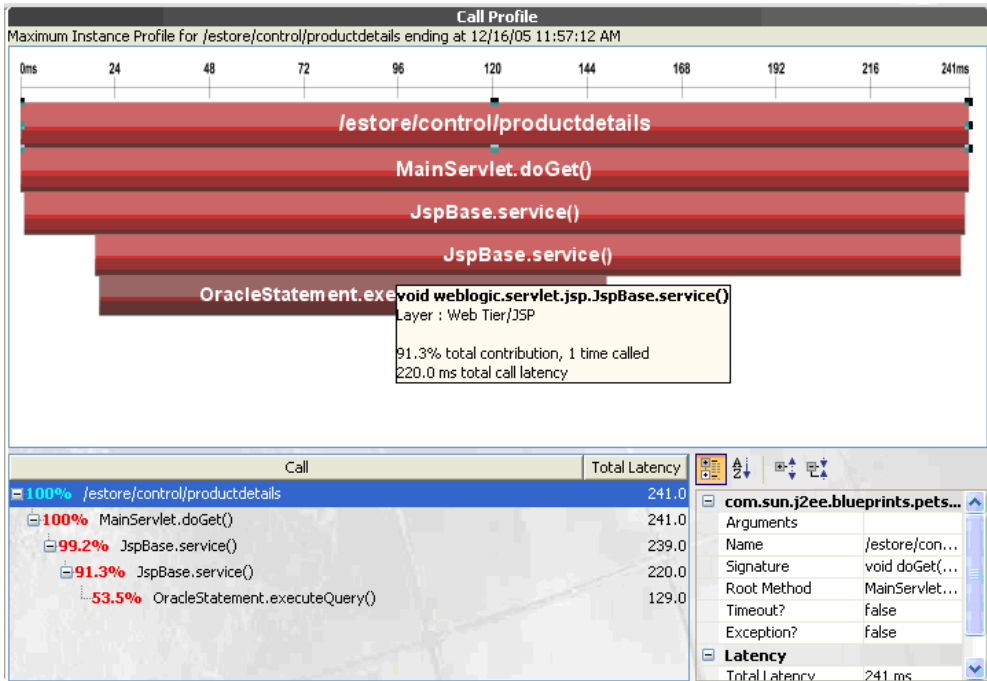
This chapter provides a detailed description about instance trees and how they are used.

This chapter describes:	On page:
Introducing Instance Trees	204
Solving Problems with Instance Trees	205
Cross-VM Trees	209
When Instance Trees Are Not Sufficient	211
Aggregate Trees	212

Introducing Instance Trees

Mercury Diagnostic probes work by adding instrumentation before and after important methods in your application to provide a simplified program trace (with elapsed times) for performance optimization and problem diagnosis.

This information is presented in a call profile visualization such as the following:



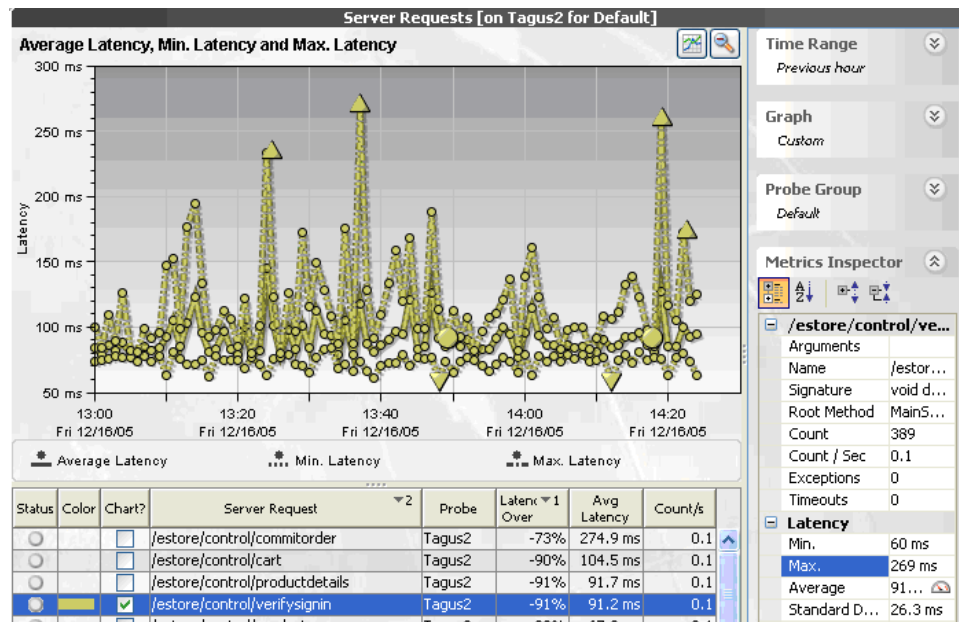
Due to storage considerations, however, Diagnostics does not record every captured trace.

Instead, at 5-minute intervals, Diagnostics uses an algorithm to select the most interesting instance trees for each server request. The selected instance trees are stored to disk and the trees that were not selected are discarded. For each server request on each probe, Diagnostics saves an instance tree for:

- ▶ the *fastest* (or minimum) invocation of that server request
- ▶ the *slowest* (or maximum) invocation of that server request
- ▶ the *second slowest* invocation of that server request
- ▶ a very close approximation of the *median* instance tree (in other words, a tree for a server request invocation that took an average amount of time)
- ▶ when the server request makes external calls (across two or more virtual machines) such as RMI, a *randomly selected end-to-end complete trace* of the entire cross-VM tree, including the trees of the remote systems (as long as the remote system was also instrumented).

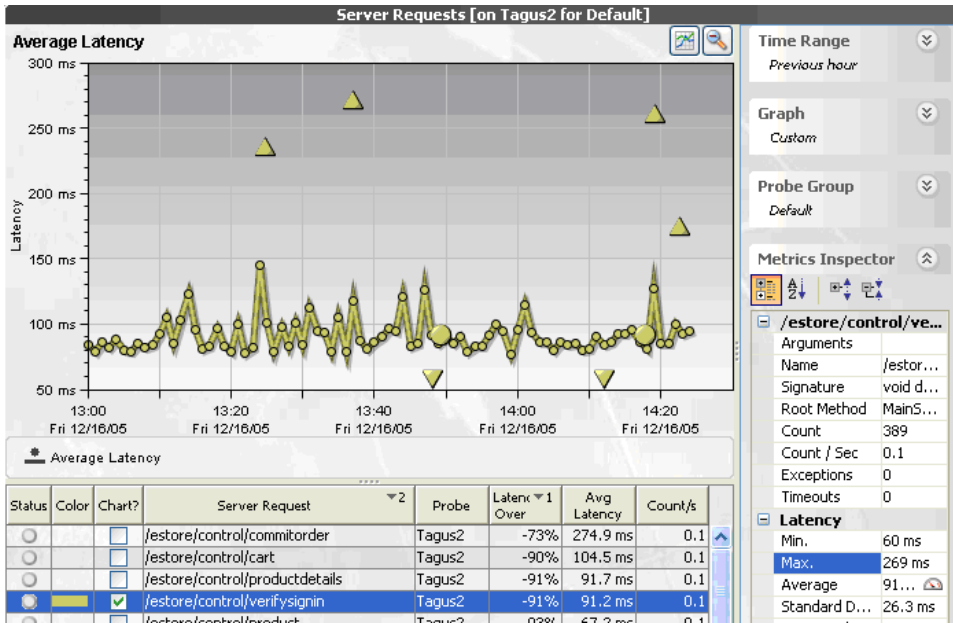
Solving Problems with Instance Trees

To allow detailed analysis of your application's performance, you can display the average, minimum, maximum, and standard deviation response times on the same graph. The following image is an example of the Server Requests view:



You can see that the response time for the `/estore/control/verifysignin` server request generally varies between about 60 ms and 140 ms, and sometimes spikes above 250 ms.

The following image is less noisy, without the minimum and maximum trends:

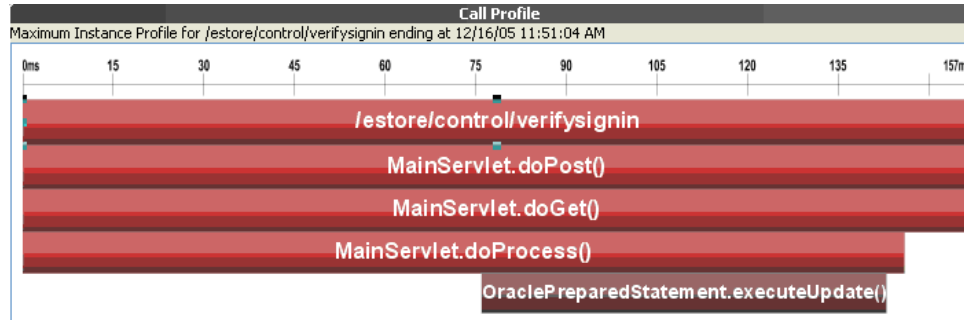


You can see small icons where the fastest, slowest, and median instance trees have been recorded:

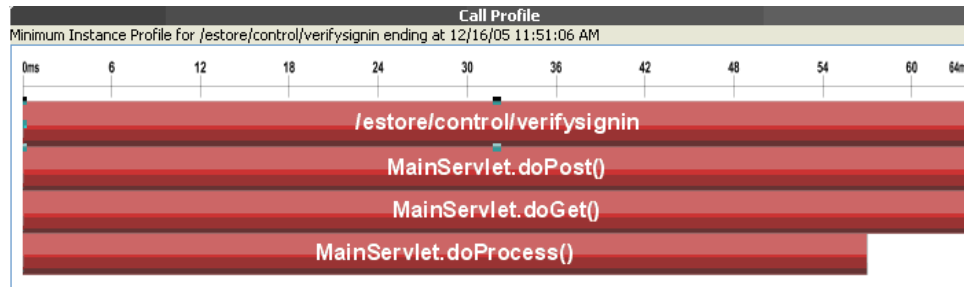
	A median instance tree
	A slow instance tree
	A fast instance tree

To understand why `/estore/control/verifysignin` is sometimes 3 times slower than average (or 5 times slower than the best case), compare the instance trees against one another:

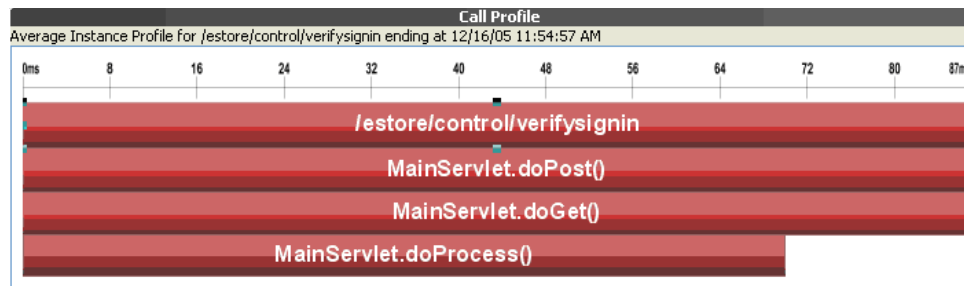
Maximum (slowest) call:



Minimum (or fastest) call:



Average (or median) call:



In the above example, in the slow case, the database (**OraclePreparedStatement.executeUpdate**) is slowing down the server request. Note that there is not much difference between the fastest and average calls in terms of application behavior.

Clearly, it is not always true that the average profile will look the same as the minimum profile. For example, consider a server request that queries information out of an EJB. There are at least three basic execution time profiles for this operation:

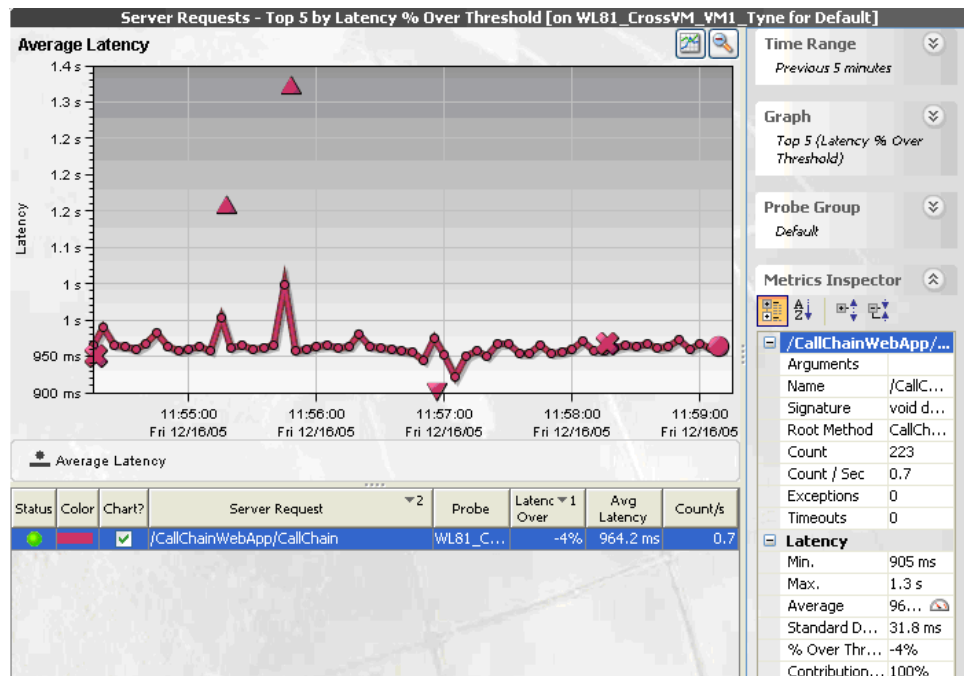
- ▶ The EJB is cached in the application server memory and the retrieval is nearly instantaneous (minimum profile).
- ▶ The EJB must be retrieved from the database (depending upon usage statistics, this could be the average case).
- ▶ The server request must wait a significant amount of time for a pooled database connection prior to requesting the EJB data (maximum profile).

Cross-VM Trees

Cross-VM trees are collected to help you quickly understand average application behavior when multiple virtual machines are involved.

The goal is to help you triage the problem to a particular virtual machine so that you can examine the situation in more detail. This can be done, for example, by pulling up application JMX metrics, or looking at system metrics such as CPU utilization or disk I/O on the problematic machine.

The following image is an example of a trend for a server request that makes cross-VM calls:





The cross sign, indicates a Cross-VM call tree. The color of the cross corresponds to the color of the server request trend line to which it relates. When you click it, a complete cross-system call profile opens:



The red nodes are on the origination system, and the blue nodes are on the child system (**tyne.lab.performant.com**).

The above call tree indicates that over half of the elapsed time for this server request is due to the child system (in this case, **tyne.lab.performant.com**).

When Instance Trees Are Not Sufficient

For some applications, three types of instance trees are not sufficient to distinguish possible application behaviors. For example, a common enterprise pattern is to have a single point of entry into a J2EE application (a "controller" servlet) that dispatches between completely different commands (with completely different performance behavior).

A URL for such a server request may appear inside Diagnostics as follows:

```
/controller
```

where the actual URLs used by customers to access the system may look like this:

```
/controller?action=checkout&userId=43a893c43&itemId=889fe
```

```
/controller?action=browseItem&userId=43a893c43&itemId=889fe
```

```
/controller?action=listCategory&userId=43a893c43&categoryId=438
```

In this case, a maximum, minimum, and average tree for **/controller** are clearly insufficient to diagnose application problems—the profile will be extremely different for each type of action processed.

For these cases, slightly customize the probe instrumentation for the **/controller** servlet in order to pick up the equivalent of the 'action' parameter as part of the Diagnostics server request. This is a minor change that Mercury Customer Support can help you make for your application.

In this case, you would see three distinct server requests inside Diagnostics:

```
/controller?action=checkout
```

```
/controller?action=browseItem
```

```
/controller?action=listCategory
```

Each of these three server requests would have minimum, maximum, and average instance trees to assist in your problem diagnosis.

Aggregate Trees

Previous versions of Diagnostics used aggregate trees instead of instance trees. In fact, the Mercury Diagnostics Profiler still supports both types of trees. This section discusses the benefits of instance trees over aggregate trees.

About Instance Trees

An instance tree call profile is a visual representation of what your application actually did in response to a particular request. Instance trees have exact start times (such as 5:09:33 PM on Tuesday), as do the methods within them.

To handle the large number of instance trees that can be generated for a heavily used server request, Diagnostics selects the most interesting instance trees to keep.

About Aggregate Trees

An aggregate tree is an amalgamation of all of the instance trees that have occurred during that 5-minute period. This ensures that no data is lost.

As nothing is ever thrown away (it is all averaged together) this overcomes the potential for losing some interesting data.

Resolving Problems with Instance and Aggregate Trees

Aggregate trees amalgamate data, but the visualized trees do not represent what actually occurred in the system.

For example, an aggregate tree will show a cache being both hit and missed (when in reality, for any particular invocation, only one of the two will occur). This can be confusing and even lead to misunderstandings when reports are passed on to others.

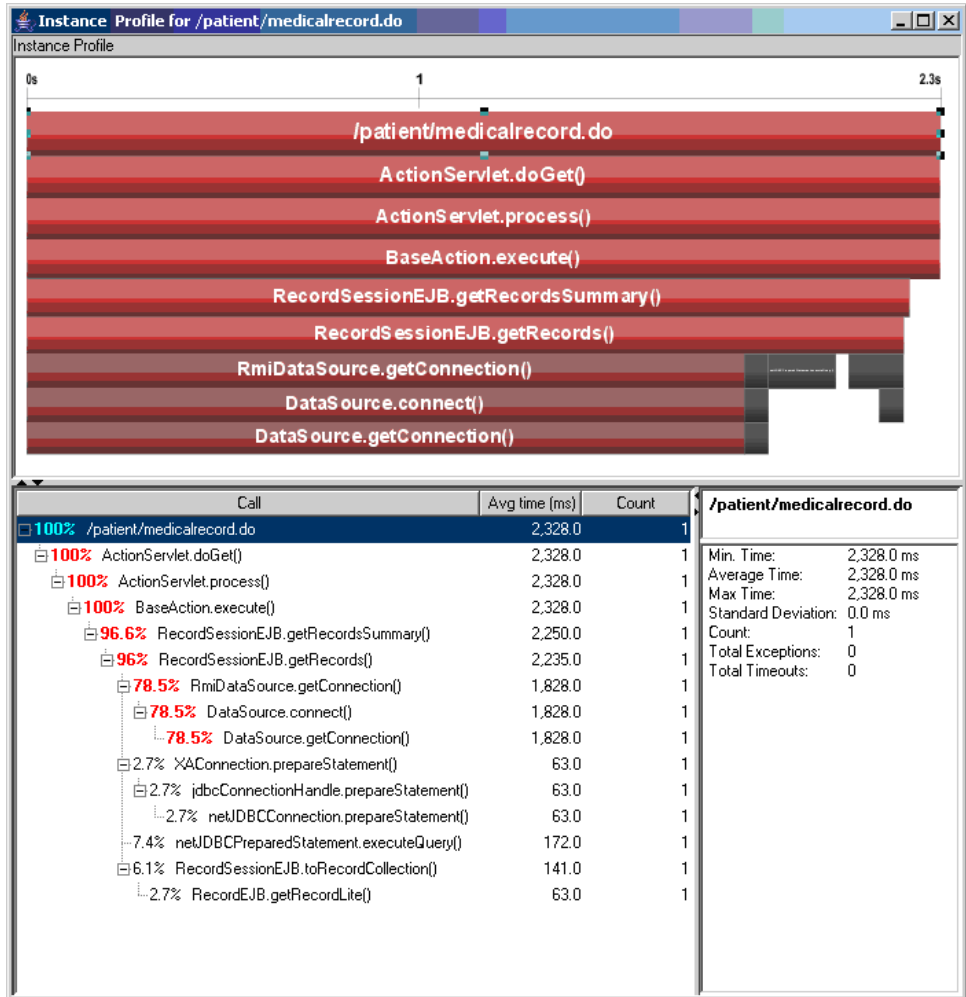
Another disadvantage is that while aggregate trees do well for general performance tuning (such as what occurs during a load test), they tend to hide information about relatively infrequent slow cases. Unfortunately, it is often these cases that cause problems for the customer.

The following examples from the Diagnostics Profiler (which has both instance and aggregate trees) demonstrate this problem:

First, consider the following two 'slowest' instance trees recorded on the medical records J2EE application.

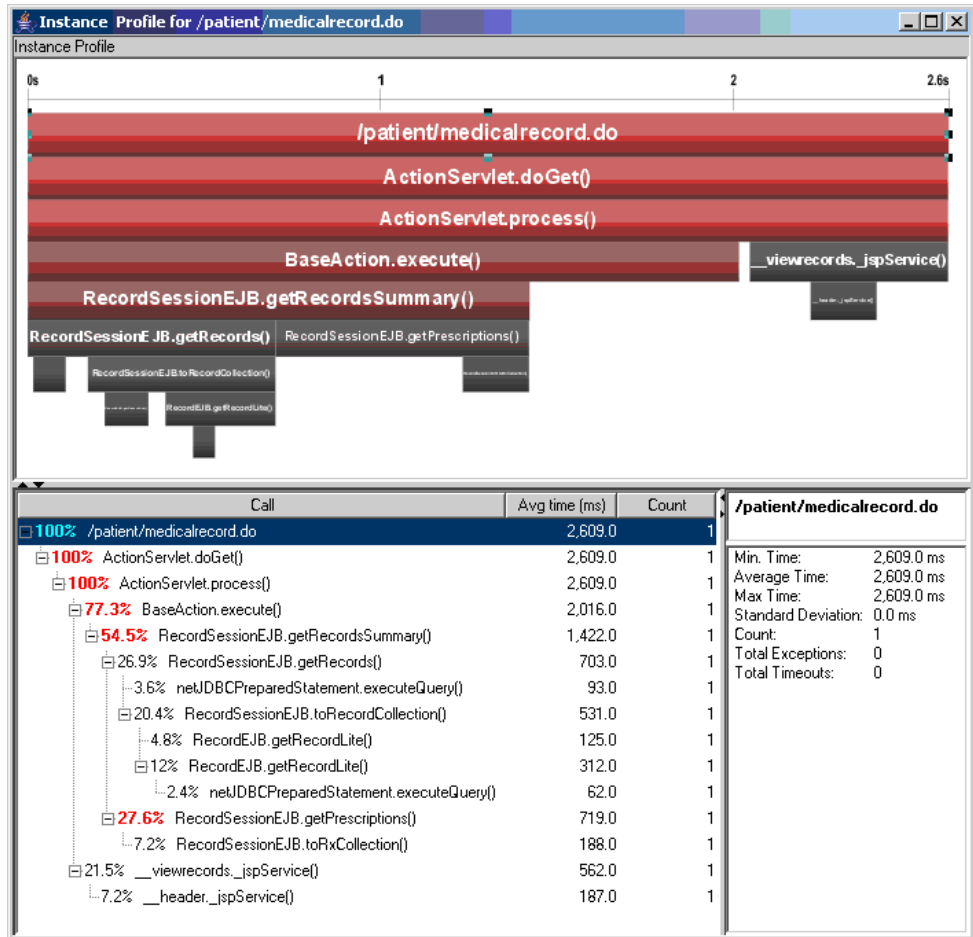
Example Instance Tree #1

The following slowest instance tree quite clearly shows that the majority of the 2.3 seconds spent responding to the `/patient/medicalrecord.do` server request was spent waiting for a JDBC connection to the database. (Is the application server's database connection pool too small for this workload?)



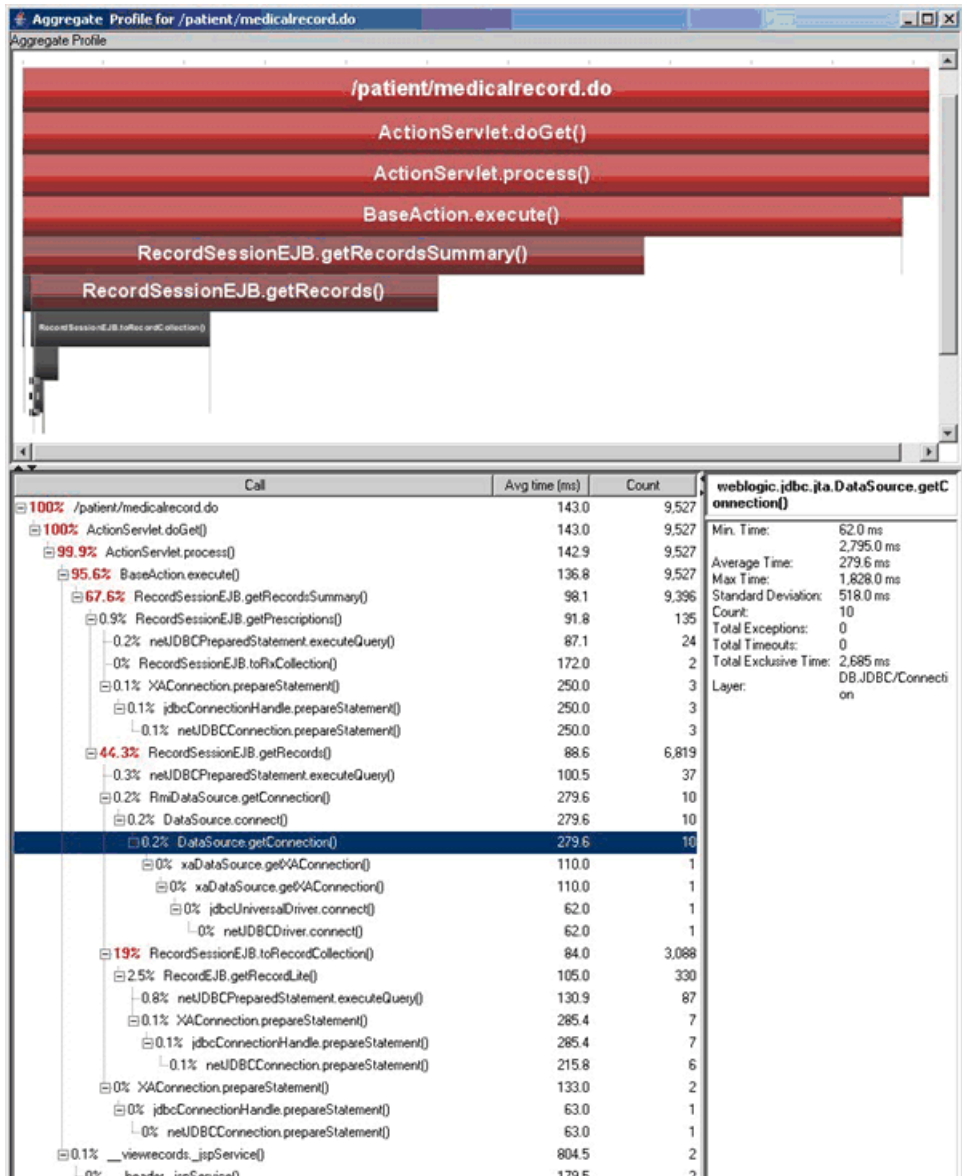
Example Instance Tree #2

This second slow instance tree from the same application shows a different call path: time spent reading records and prescriptions from the database. (Is the database overloaded? Are the SQL queries run by these methods optimal?)



Aggregate Tree Example

The instance trees in the previous two examples show distinctly different problems for the `/patient/medicalrecord.do` server request. The following aggregate tree was produced:



Several things can be deduced from this example:

- ▶ The aggregate tree is much larger. As it combines multiple possible execution paths into a single tree, it is more difficult to understand what the application is doing.
- ▶ If you want to improve the overall performance of your application, the aggregate tree does indicate that you should concentrate on the common case (that the database connection was quickly retrieved from the pool).
- ▶ The aggregate tree does not identify that occasionally this server request stalls because it cannot obtain a database connection from the pool. **DataSource.getConnection()** contributes only 0.2%, approximately the same as other methods that have never caused a problem (such as **RecordEJB.getRecordLite**).

Part III

Using the Mercury Diagnostics Profiler for J2EE

10

Using the J2EE Diagnostics Profiler

This chapter provides an overview of the processing of the Mercury Diagnostics Profiler for J2EE, descriptions of the screens, and instructions for using some of the global user interface controls.

This chapter describes:	On page:
Accessing the Mercury Diagnostics Profiler for J2EE	222
Mercury Diagnostics Profiler for J2EE Processing	223
Common J2EE Diagnostics Profiler Tab Navigation and Display Controls	225

Note: The Mercury Diagnostics Profiler for J2EE operates in an unlicensed mode with load restrictions until the probe is able to connect to a Diagnostics Server that has been properly licensed. In unlicensed mode, the Profiler is limited to capturing data from 5 concurrent threads.

For more information about licensing, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

If you installed the probe from the Mercury Web site and you want to use it with a Diagnostics Server, contact Mercury Support.

Accessing the Mercury Diagnostics Profiler for J2EE

The J2EE Diagnostics Profiler is installed as part of the Mercury Diagnostics Probe for J2EE (J2EE Probe).

Once you have installed and configured the J2EE Probe and you have started the application that is being monitored, you can access the J2EE Diagnostics Profiler from your browser and view Diagnostics data. You can also access the J2EE Diagnostics Profiler by drilling down from the Mercury Diagnostics screens.

To view Diagnostics data from the J2EE Diagnostics Profiler:

- 1 In your browser, go to the J2EE Diagnostics Profiler URL:
http://<probe_host>:<probeport>/profiler.

The probes are assigned to the first available port beginning at 35000.

Note: You can find the port that a particular probe is using in the probe's **probe.log** file located in **<probe_install_directory>/log/<probe_id> directory**. In the **probe.log** file, find the line that begins with the words **webserver listening on**, for example: **webserver listening on 0.0.0.0:35003**. The port is the number after the colon, in this example 35003.

- 2 Type your username and password.

You are prompted to enter a username and password. The default username is **admin**. The default password is **admin**. You may be prompted again to enter a username and password. Re-enter the same details.

For more details about usernames and passwords, see the *Mercury Diagnostics Installation and Configuration Guide*.

To drill down to the Diagnostics Profiler from Mercury Diagnostics:

- From any Status screen in Mercury Diagnostics, right-click the probe entity and select **Open Mercury Diagnostics Profiler** from the menu.
- Alternatively, in the standard Probes view in Mercury Diagnostics, right-click the probe entity in the detail table and select **Open Mercury Diagnostics Profiler** from the menu.

If the Profiler fails to open when performing the drill down, ensure that you have set a default browser within your operating system.

Mercury Diagnostics Profiler for J2EE Processing

This section describes the way in which the J2EE Probe monitors your application and how this data is displayed in the J2EE Diagnostics Profiler.

Monitoring Method Latency and Call Stacks

The Mercury Diagnostics Probe for J2EE (J2EE Probe) monitors your application and keeps track of the metrics for all of the instrumented methods that your application calls. As it is monitoring, it captures the call stack for the three slowest instances of each server request. It also captures a call stack representing the aggregated latency of all call instances for a type of service request.

When a new server request instance is encountered that is slower than one of the currently captured instances for the server request, it replaces one of the previously captured instances.

The J2EE Diagnostics Profiler displays metrics for all of the instrumented methods. You can drill down to the instances of the methods that were included in one of the four server request call stacks that were captured when you accessed the J2EE Diagnostics Profiler user interface. While you are analyzing the information displayed on the various tabs of the J2EE Diagnostics Profiler, you are working with the methods and call stacks captured from the time that the user interface was started. In the meantime, to minimize performance impacts, the J2EE Probe continues to monitor your application, capture method metrics, and capture call stacks.

For more information on monitoring method latency with the J2EE Diagnostics Profiler, see Chapter 11, “Analyzing Method Latency with J2EE Diagnostics Profiler Screens.”

Monitoring Application Memory

The J2EE Diagnostics Profiler allows you to monitor your application's memory usage using one of the following methods:

- Light Weight Memory Diagnostics
- Heap Breakdown

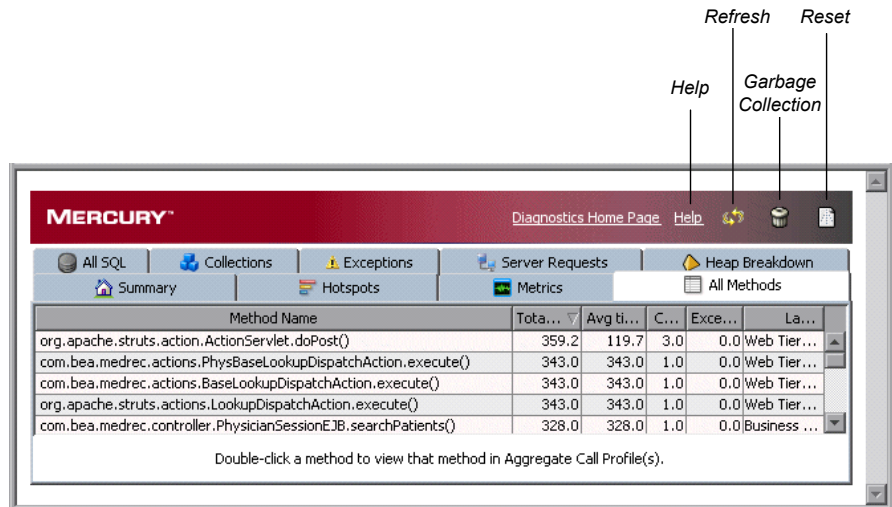
By default, both of these methods are disabled. Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you to identify objects that may be leaking.

For more information about Light Weight Memory Diagnostics, see “Analyzing Memory Using the Collections Tab” on page 257.

For more information on Heap Breakdown, see “Analyzing Memory Using the Heap Breakdown Tab” on page 262.

Common J2EE Diagnostics Profiler Tab Navigation and Display Controls

This section describes the following features and controls that are common to all of the J2EE Diagnostics Profiler tabs:



Refreshing Metrics

If you would like to work with more current performance metrics, click the **Refresh** button on the top right corner of the screen to refresh the information displayed with the latest metrics and call stacks. The system does not refresh itself automatically.



Resetting Metrics

You can force the J2EE Diagnostics Profiler to use new baselines for the calculation of instance counts, average latency, and slowest latency, and to force-drop all captured call stacks, by clicking the **Reset** button.



Note: You may want to do this after your system has warmed up so that the metrics represent processing that takes place when your application is running in a more steady state.

Garbage Collection



When you want to deallocate used memory, you can forcibly perform garbage collection inside the JVM by clicking the **Garbage Collection** button on the top right corner of the screen

Accessing Help

When you click **Help**, on the top right hand corner of the screen, you access the on-line help manual for the Mercury Diagnostics Profiler for J2EE.

11

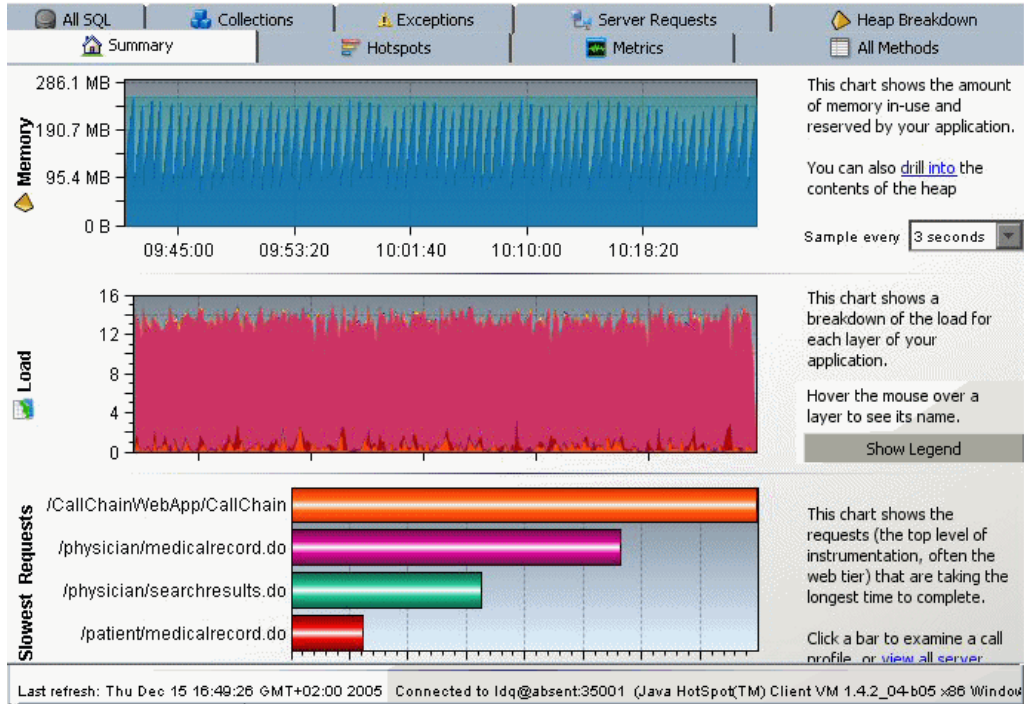
Analyzing Method Latency with J2EE Diagnostics Profiler Screens

This chapter provides a detailed description of the screens, graphs, and tables that are used to present the J2EE performance metrics for the application that is being analyzed.

This chapter describes:	On page:
Analyzing Performance Using the Summary Tab	228
Analyzing Performance Using the Hotspots Tab	232
Analyzing Performance Using the Metrics Tab	234
Analyzing Performance Using the All Methods Tab	236
Analyzing Performance Using the All SQL Tab	240
Analyzing Performance Using the Exceptions Tab	242
Analyzing Performance Using the Server Requests Tab	244
Analyzing Performance Using the Call Profile Window	247
Analyzing Performance Using the Web Services Tab	254

Analyzing Performance Using the Summary Tab

The Summary tab consists of graphs that display information about the memory in use and reserved by your application, the load for each layer of your application and the slowest requests made to your application server.

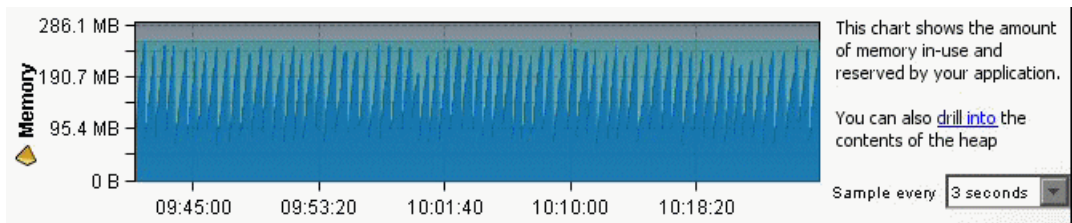


Understanding the Summary tab

The Summary tab is divided into the following sections:

Memory Graph

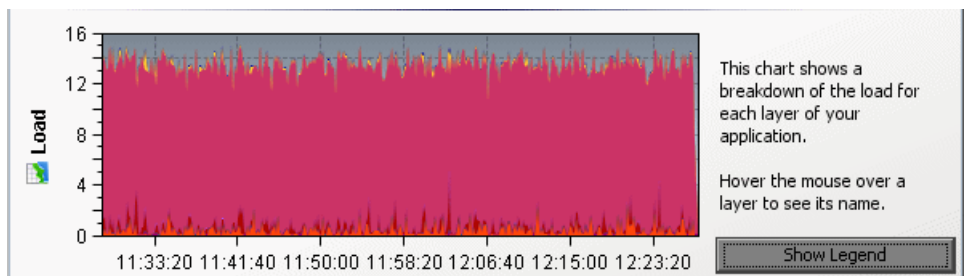
The Memory graph shows the amount of memory allocated in your application and the amount of memory reserved by your application.



You can see more details about the exact amount of allocated memory or reserved memory in your application, by holding your pointer on different sections of the graph and viewing the tooltip.

Load Graph

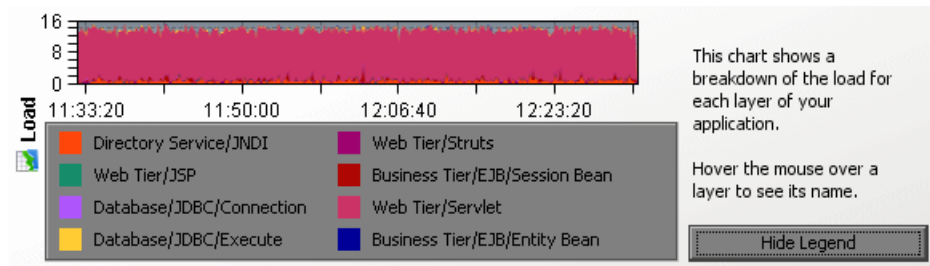
The Load graph shows the breakdown of the load for each layer of your application.



Note: The performance metrics for classes and methods are grouped into layers based upon the resources that the application invokes to perform the processing. The J2EE Diagnostics Profiler displays the layers on one level and does not split them into sublayers.

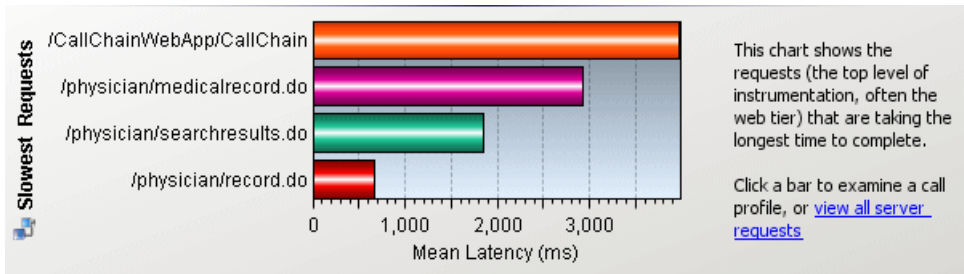
You can see the name of each layer by holding your pointer on different sections of the graph and viewing the tooltip.

To view a legend of the graph that displays the names of all the layers, click **Show Legend**.



Slowest Requests Graph

The Slowest Request graph shows the server requests that are taking the longest time to complete.



Note: To view the aggregated call profile for a server request in the Slowest Request graph, click the relevant bar. For more information about the call profile window, see “Analyzing Performance Using the Hotspots Tab” on page 232.

Information Pane

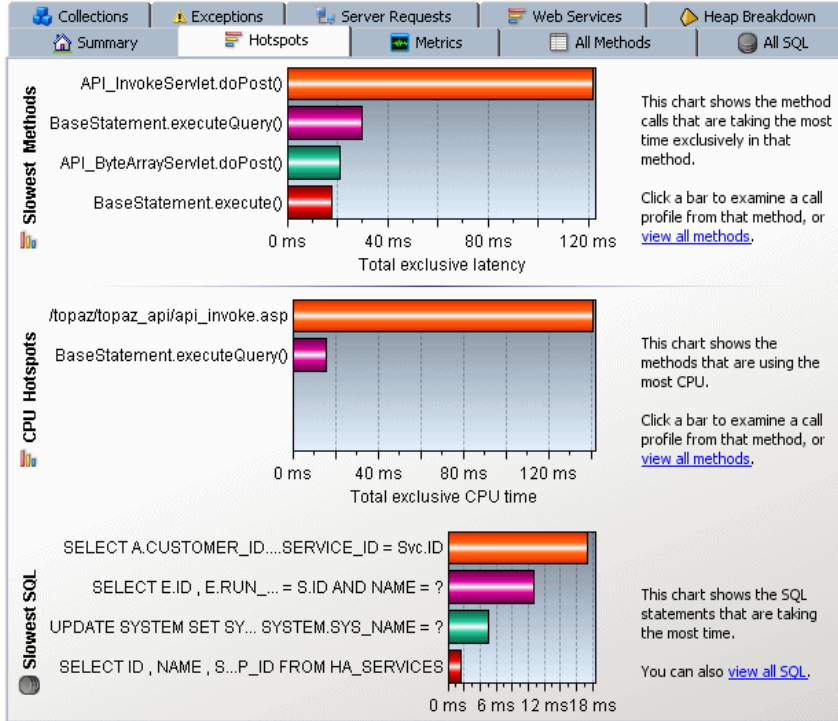
The information pane at the bottom of the window displays the following information:

- The date and time of the last time you refreshed the system.
- The JVM details.
- The Operating System details
- The probe ID.

Last refresh: Thu Dec 15 16:49:26 GMT+02:00 2005 Connected to ldq@absent:35001 (Java HotSpot(TM) Client VM 1.4.2_04-b05 x86 Windows 2003)

Analyzing Performance Using the Hotspots Tab

The Hotspots tab displays bar charts of the significant metrics that have been captured during the monitoring of your application.



Note: You can view the details of any of the graph metrics by holding your pointer over the relevant bar and viewing the tooltip.

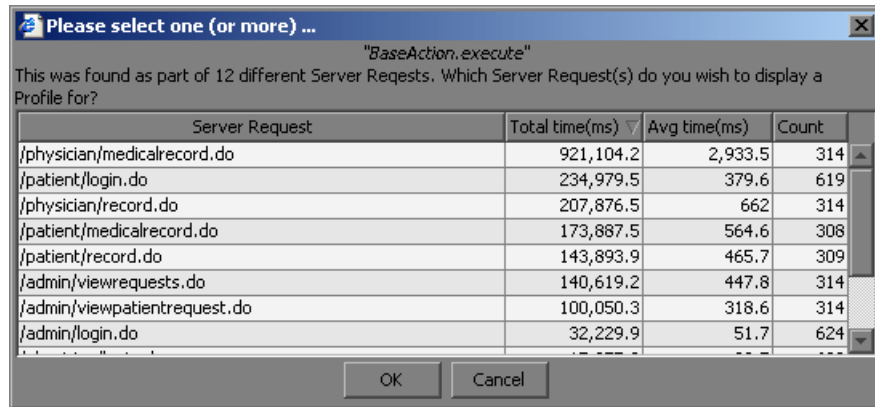
Understanding the Hotspots Tab

The Hotspots tab contains the following three sections:

Slowest Methods Graph

This chart shows the method calls that are taking the most time exclusively in that method. To view the call profile for a method call in the Slowest Methods graph, click the relevant bar. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

If the method is part of more than one server request, when you double click the method, the following dialog box opens and asks you to select the relevant server request.



Double-click the appropriate server request row to view the call profile.

CPU Hotspots

This chart shows the methods that are using the most CPU.

To view the call profile for a particular method, click the relevant bar. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

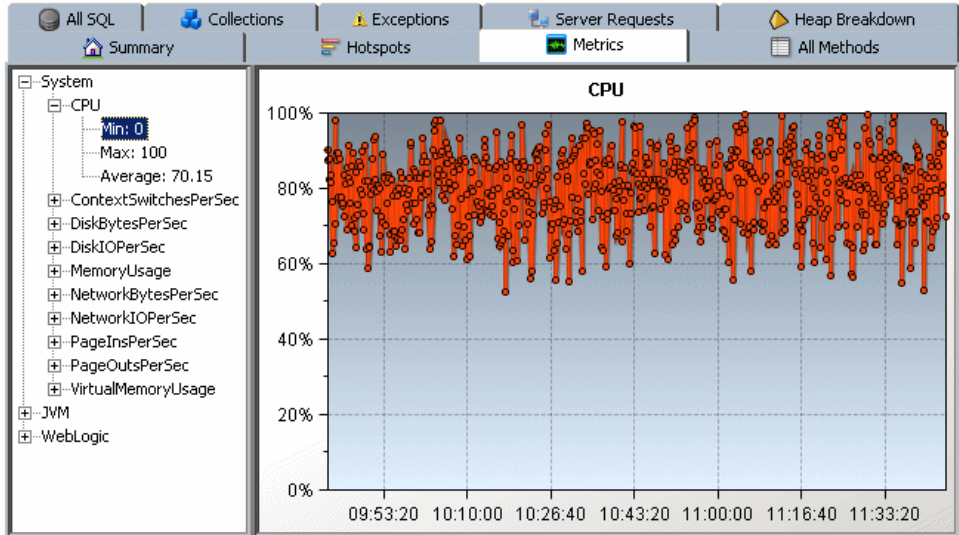
Slowest SQL Graph

This chart shows the SQL statements that are taking the most time.

To view the SQL statement details for a selected statement in the Slowest SQL graph, click the relevant bar. For more information about SQL statement details, see “Analyzing Performance Using the All SQL Tab” on page 240.

Analyzing Performance Using the Metrics Tab

The Metrics tab displays information about the Operating System, the JVM and the application server.



Understanding the Metrics Tab

The Metrics tab is divided into two panes:

- ▶ **The Tree Pane.** displays the metrics in an expandable tree.
- ▶ **The Graph Pane.** displays a graph of the selected metric from the tree pane.

The top three levels displayed in the tree are:

- ▶ **System.** metrics about the Operating System
- ▶ **JVM.** metrics about the JVM
- ▶ **<application server>.** metrics about the application server. The application servers that will be displayed are Weblogic, Websphere or SAP.

Note: When more than one probe is running on the same host, the System metrics only appear for one of the probes.

When you expand each of the top levels, the tree displays the associated metrics for each top level. As you further expand each metric, you arrive at a minimum, a maximum and an average numerical value for each metric.

When you click on a specific metric in the tree, the graph pane displays a graph representing the selected metric. You can select more than one metric to display in the graph pane using the **Control** key.

The X-axis in the graph represents time. The Y-axis in the graph represents the numerical value of the metric. Metrics are displayed for the last five minutes unless the probe is part of another Mercury Diagnostics product, in which case they are displayed for three hours.

Analyzing Performance Using the All Methods Tab

The All Methods tab lists the method calls that your application makes according to the instrumentation in the auto_detect Points file.

Method Name	Total time(ms)	Avg time(ms)	Count	Exceptions	Total CPU(ms)	Avg CPU(ms)	Layer
com.sun.j2ee.blueprints.pet...	1,596.1	66.5	24.0	0.0			Web Tier/Servlet
weblogic.servlet.jsp.JspBas...	1,135.1	126.1	9.0	0.0			Web Tier/JSP
com.sun.j2ee.blueprints.pet...	642.6	214.2	3.0	0.0			Web Tier/Servlet
com.sun.j2ee.blueprints.pet...	419.8	419.8	1.0	0.0			Web Tier/Servlet
weblogic.servlet.FileServlet...	312.5	0.9	353.0	0.0			Web Tier/Servlet
weblogic.servlet.internal.Se...	216.2	0.3	778.0	0.0	220.0	0.3	Web Tier/Session
\$Proxy0.getProfileMgr()	111.3	111.3	1.0	0.0			Business Tier/EJ...
com.sun.j2ee.blueprints.pet...	111.1	111.1	1.0	0.0			Business Tier/EJ...
\$Proxy0.handleEvent()	110.6	110.6	1.0	0.0			Business Tier/EJ...
com.sun.j2ee.blueprints.pet...	110.3	110.3	1.0	0.0		0.3	Business Tier/EJ...
oracle.jdbc.driver.OraclePre...	97.0	97.0	1.0	0.0			Database/JDBC/...
\$Proxy2.getOrderDetails()	91.7	91.7	1.0	0.0			Business Tier/EJ...
com.sun.j2ee.blueprints.cus...	91.4	91.4	1.0	0.0			Business Tier/EJ...
com.sun.j2ee.blueprints.per...	74.4	74.4	1.0	0.0			Business Tier/EJ...

Double-click a method to view that method in Aggregate Call Profile(s). All times shown are exclusive (not including time spent in profiled chil...

Understanding the All Methods Tab

The All Methods tab displays a table that contains the following columns:

- ▶ **Method name.** The name of the methods that were called. The Method name has the following syntax: **<package name>.<class name>.<method name>**.
- ▶ **Total Time.** The aggregate latency for all of the calls to the method. The total latency is shown in milliseconds.
- ▶ **Avg Time.** The average latency for all of the calls to the method. The average latency is shown in milliseconds.
- ▶ **Count.** The number of times that the method was invoked.
- ▶ **Exceptions.** The number of times that the method generated an exception.

- **Total CPU.** The total amount of CPU time that all invocations of the listed method used.

The CPU Time Java metric relies on CPU timestamping which is supported on the following platforms:

- Windows
- Solaris 8+
- AIX 5+

Use caution when enabling the collection of CPU timestamps because of the increase in Diagnostics overhead. The increased overhead is caused by an additional call for each method that is needed to collect the timestamp.

The capture of CPU Time is controlled by two properties:

- The **use.cpu.timestamps** property in the **capture.properties** file located in `<probe_install_directory>\etc` must be set to **true**
- The **cpu.timestamp.collection.method** property in `<probe_install_dir>\etc\dynamic.properties` must be set to one of the following valid values:
 - 0 - No CPU timestamping
 - 1 - CPU timestamps will be collected only for server requests
 - 2 - CPU timestamps will be collected for all methods

The default value is 1, which means that CPU times can be reported at the server request level but not at the transaction level.

If you want to turn off all CPU timestamping so that CPU time will not be gathered and reported for server requests, set the property to 0. If you want to gather CPU timestamping for all methods so that CPU time will be gathered and reported for all methods, set the property to 2.

For BPM transactions, the **cpu.timestamp.collection.method** property is ignored. CPU timestamping will always be collected for all methods for a BPM transaction.

- ▶ **Avg CPU (not enabled by default).** The CPU time that the method used during an average invocation.

The **Avg CPU** metric is not enabled by default. To enable it, set the property **use.cpu.timestamps** in the **capture.properties** file located in **<probe_install_directory>\etc** to **true**.

Note: CPU time is supported on Windows, Solaris 8 and above, and AIX 5 and above.

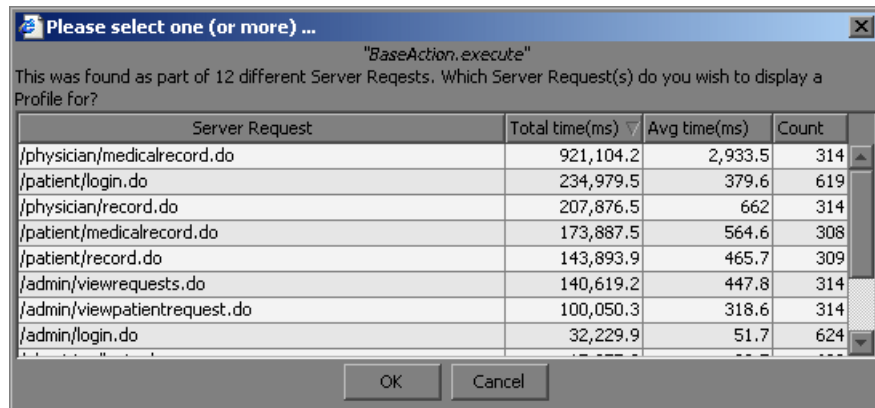
- ▶ **Layer.** The Layer associated with this method according to the instrumentation in the **auto_detect** Points file. The layers are displayed on one level and there is no distinction made between layers and sub-layers.

Note: All of the metrics in the All Methods tab are counted from the time you enter the system or click the **Reset** button.



To view the call profile for a method call, double-click the appropriate row. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

If the method is part of more than one server request, when you double click the method, the following dialog box opens and asks you to select the relevant server request:



Double-click the appropriate server request row to view the call profile.

To create call profiles from more than one server requests select the first server request with a single click and select subsequent server request using control click. When you have finished making your selections, click **OK** to instruct the Profiler to create the call profiles. The call profile for each selected server request is displayed in a separate window.

Analyzing Performance Using the All SQL Tab

The All SQL tab displays the SQL statements in a table.

SQL	Total time(ms)	Avg time(ms)	Count	Exceptions
SELECT WLO.record_date , WLO.diagnosis , WLO.id , WLO.notes , WLO.pat...	32,978.9	196.3	168	0
SELECT WLO.id , WLO.record_date , WLO.pat_id , WLO.symptoms , WLO.re...	31,882.2	148.3	215	0
SELECT WLO.id , WLO.date_prescribed , WLO.dosage , WLO.drug , WLO.fre...	24,339	146.6	166	0
SELECT WLO.id , WLO.city , WLO.country , WLO.state , WLO.street1 , WLO...	22,025.8	108	204	0
SELECT WLO.id , WLO.address_id , WLO.dob , WLO.email , WLO.first_name ...	19,609.3	121	162	0
SELECT WLO.id , WLO.date_prescribed , WLO.dosage , WLO.drug , WLO.fre...	14,297.3	143	100	0
SELECT WLO.id , WLO.record_date , WLO.diagnosis , WLO.notes , WLO.pat...	14,105.6	134.3	105	0
SELECT WLO.username , WLO.password , WLO.status FROM medrec_user ...	13,345.7	158.9	84	0
SELECT PASSWORD FROM medrec_user WHERE username = ? AND status ...	11,608.7	124.8	93	0
SELECT WLO.id , WLO.address_id , WLO.dob , WLO.email , WLO.first_name ...	10,575.7	112.5	94	0
SELECT WLO.id , WLO.blood_pressure , WLO.height , WLO.pulse , WLO.tem...	9,656.4	106.1	91	0
SELECT group_name FROM GROUPS GROUPS WHERE groups.username = ?	9,249.7	123.3	75	0
SELECT WLO.id , WLO.address_id , WLO.dob , WLO.email , WLO.first_name ...	8,682.7	118.9	73	0
SELECT COUNT (*) FROM MedRecJMSState	488	61	8	0

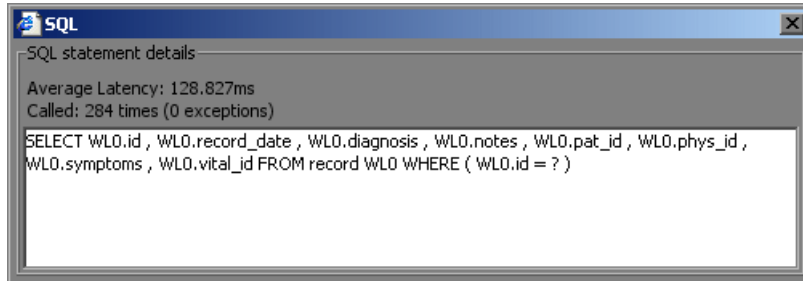
Understanding the All SQL Tab

The All SQL tab displays the SQL Statement table, which contains the following columns.

- ▶ **SQL.** The name of the SQL statement that was invoked by the application server.
- ▶ **Total Time.** The total latency of all invocations of the SQL statement.
- ▶ **Avg Time.** The average latency of all invocations of the SQL statement.
- ▶ **Count.** The number of times the SQL statement was invoked by the application server.
- ▶ **Exceptions.** The number of times that the statement generated an exception.

Viewing SQL Statement Details

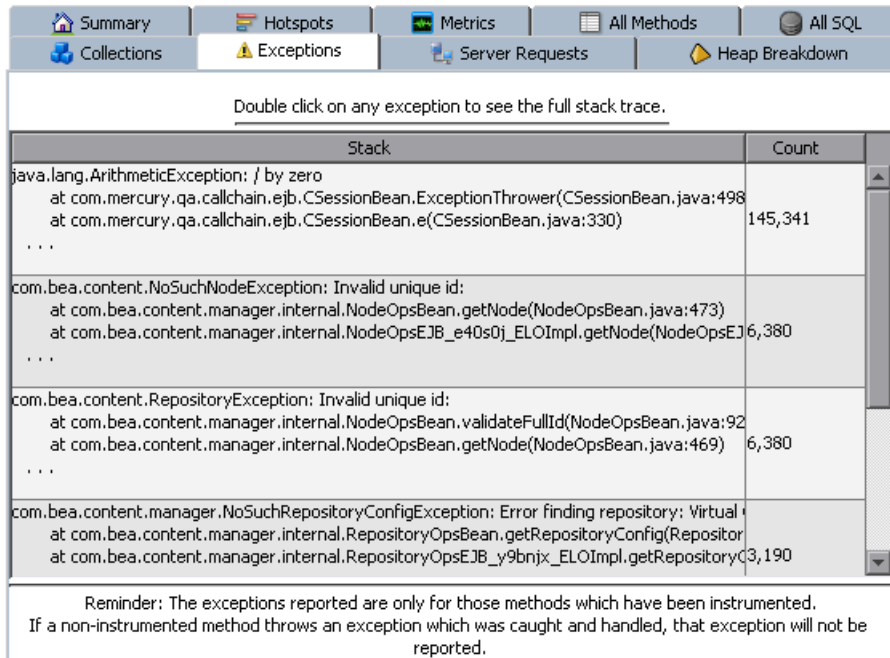
To view the SQL statement details, double-click on the relevant statement. The SQL statement details dialog box opens, displaying all the information shown in the SQL table for each statement.



The SQL statement details dialog box enables you to view the full string of the SQL statement and to copy the text.

Analyzing Performance Using the Exceptions Tab

The Exceptions tab displays all the exceptions that were generated in the application server for methods that have been instrumented.



Double click on any exception to see the full stack trace.

Stack	Count
<pre>java.lang.ArithmeticException: / by zero at com.mercury.qa.callchain.ejb.CSessionBean.ExceptionThrower(CSessionBean.java:498) at com.mercury.qa.callchain.ejb.CSessionBean.e(CSessionBean.java:330) ...</pre>	145,341
<pre>com.bea.content.NoSuchNodeException: Invalid unique id: at com.bea.content.manager.internal.NodeOpsBean.getNode(NodeOpsBean.java:473) at com.bea.content.manager.internal.NodeOpsEJB_e40s0j_ELOImpl.getNode(NodeOpsEJB_e40s0j_ELOImpl.java:330) ...</pre>	6,380
<pre>com.bea.content.RepositoryException: Invalid unique id: at com.bea.content.manager.internal.NodeOpsBean.validateFullId(NodeOpsBean.java:92) at com.bea.content.manager.internal.NodeOpsBean.getNode(NodeOpsBean.java:469) ...</pre>	6,380
<pre>com.bea.content.manager.NoSuchRepositoryConfigException: Error finding repository: Virtual at com.bea.content.manager.internal.RepositoryOpsBean.getRepositoryConfig(RepositoryOpsBean.java:330) at com.bea.content.manager.internal.RepositoryOpsEJB_y9bnjx_ELOImpl.getRepositoryConfig(RepositoryOpsEJB_y9bnjx_ELOImpl.java:330) ...</pre>	3,190

Reminder: The exceptions reported are only for those methods which have been instrumented. If a non-instrumented method throws an exception which was caught and handled, that exception will not be reported.

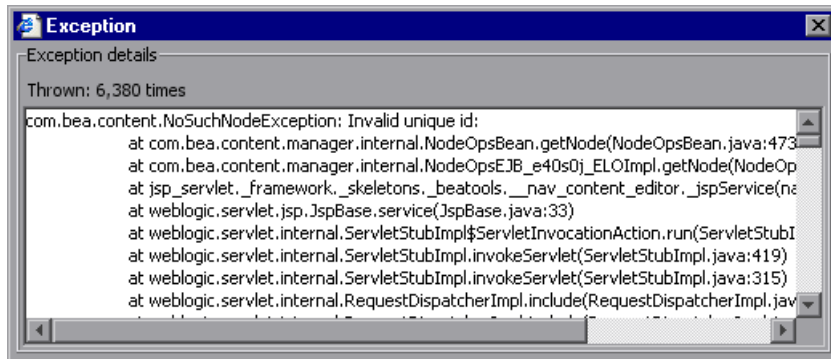
Note: If a non-instrumented method throws an exception which was caught and handled, that exception will not be reported

Understanding the Exceptions Tab

The Exceptions tab displays the Exceptions table which contains the following columns:

- **Stack.** Shows the first three lines of the exception stack trace.
- **Count.** The number of times the exception was generated.

To see the full stack trace of the exception, double-click the row containing the exception to open the Exception dialog box.



Analyzing Performance Using the Server Requests Tab

The **Server Requests** tab displays information about the server requests made to the application server.

Server Request	Total time(ms)	Avg time(ms)	Count	Avg CPU(ms)	Layer
/estore/control/verifysignin	419.9	419.9	1.0	120.0	
/estore/control/commitorder	390.1	390.1	1.0	270.0	
Static Content	351.8	1.0	353.0	0.8	
/estore/control/productdetails	242.9	242.9	1.0	100.0	
weblogic.servlet.internal.ServletRequestImpl.ge...	216.2	0.3	778.0	0.3	Web Tier/Session
/estore/control/cart	165.2	165.2	1.0	110.0	
/estore/control/product	76.1	76.1	1.0	40.0	
/estore/control/signout	59.1	59.1	1.0	20.0	
/estore/control/language	58.7	58.7	1.0	40.0	
/estore/control/rchecknuit	49.0	49.0	1.0	40.0	

Double click one of these Slow Instances to see a call profile

Server Request	Start T...	End Time...	Total ti...	Threw Ex...
/estore/control/verifysignin	05/18/06 10:25...	05/18/06 10:2...	419.9	

Understanding the Server Requests Tab

The aggregated server request table at the top of the tab lists the aggregated performance information for all instances of the server requests.

When you select a server request in this table by clicking the row, the table at the bottom of the tab is populated with the three server request instances that have the worst total time.

When you double click on a server request in this table, the Profiler displays the call profile for the selected aggregated server request in a new window. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

The aggregated Server Requests contains the following columns:

Server Request. The URI or the root method for the server request.

Note: The URI parameters are trimmed. To break down server requests according to URI parameters, contact Mercury Customer Support. (<http://support.mercury.com>)

- ▶ **Total Time.** The total latency of all invocations of the server request.
- ▶ **Average Time.** The Average latency of all invocations of the server request.
- ▶ **Count.** The number of times this server request was invoked.
- ▶ **Avg CPU (not enabled by default).** The CPU time that the method used during an average invocation.

The **Avg CPU** metric is not enabled by default. To enable it, change **use.cpu.timestamps** in the **capture.properties** file located in `<probe_install_directory>\etc` from **false** to **true**.

Note: CPU time is supported on Windows, Solaris 8 and above, and AIX 5 and above.

- ▶ **Layer.** Displays the layer for server requests that were invoked by root methods that are not part of an HTTP request. HTTP server requests do not have a layer.

Viewing the Slowest Instances for a Server Request

When you click on a server request, the bottom section of the window displays a table containing the three slowest instances of the server request.

To view the instance call profile for an instance of a server request, double-click a server request instance. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

Server Request		Total time(ms)	Avg time(ms)	Count	Layer
/physician/search.do		22,499.9	35.8	629	
/physician/login.do		17,955.9	28.5	630	
com.pointbase.net.netJDBCPreparedStatement.executeQuery()		865	50.9	17	Database/JDBC/...
/index.jsp		48.9	0.2	315	

Double click one of these Slow Instances to see a call profile

Server Request	Start Timestamp	End Timestamp	Total time(ms)	Threw Exception?
com.pointbase.net.netJDBCPr...	12/07/05 10:02:44.046	12/07/05 10:02:44.171	125	
com.pointbase.net.netJDBCPr...	12/07/05 09:42:44.062	12/07/05 09:42:44.155	94	
com.pointbase.net.netJDBCPr...	12/07/05 09:37:44.078	12/07/05 09:37:44.217	140	

The table contains the following columns:

- ▶ **Server Request.** The name of the server request.
- ▶ **Start Timestamp.** Point in time when the server request instance was invoked.
- ▶ **End Timestamp.** Point in time when the server request ended.
- ▶ **Total Time.** Total amount of time the server request took to execute.
- ▶ **Threw Exception.** Indicates whether or not an exception was thrown during the processing of this server request instance.

Analyzing Performance Using the Call Profile Window

The Call Profile Window displays a graphical representation of the method call stack for a selected server request. The depicted server request can be an aggregation of all of the calls made to the selected server request or a single instance of the server request depending on the server request on which you drilled down to open the call profile window. The metrics depicted in the graphical representation of the call stack are also depicted in the Call Tree Table on the same tab.

Types of Call Profile Windows

There are two types of call profile windows that are displayed depending on the how you navigated to the tab:

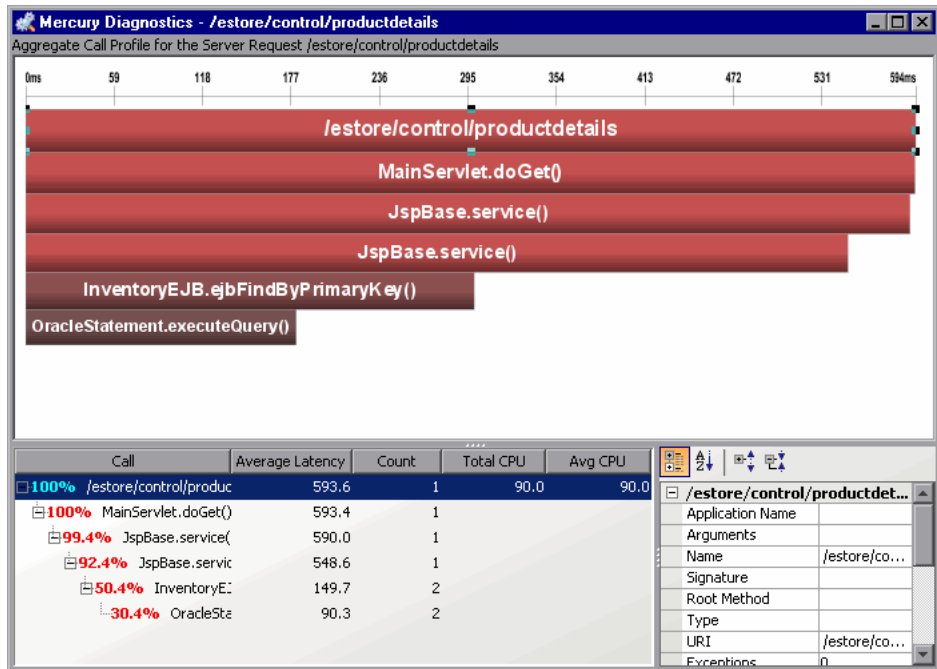
- **The Instance Call Profile window** displays the method calls that were made during the processing of the server request on which you drilled down.
- The **Aggregate Profile window** displays an aggregation of all of the method calls that were made during the processing of all of the server requests that were the same as the one on which you drilled down.

Description of Call Profile Window

The Call Profile Window is made up of three areas:

- Call Profile Graph graph
- Call Tree Table
- Metrics Inspector

The following image is an example of an Aggregate Call Profile Window showing all three of these areas:

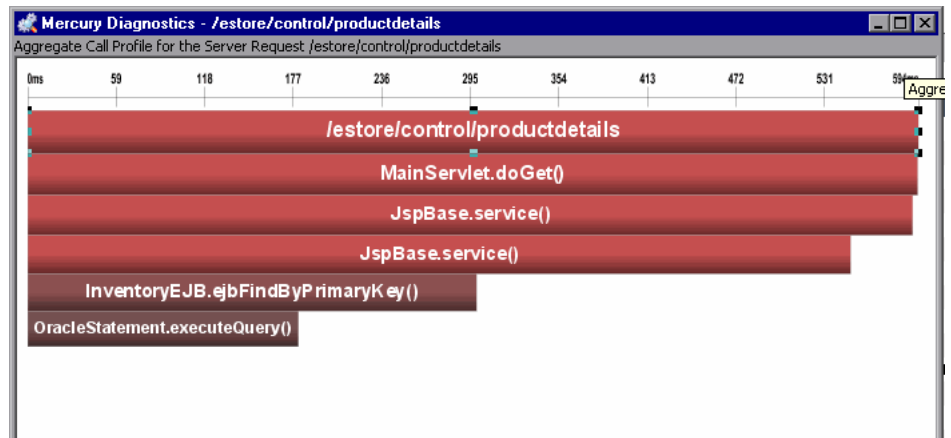


The three areas of the Call Profile Window work together to provide you with the information that you need to diagnose performance issues in your applications. When you click a call box in the Call Profile graph, the corresponding row is selected in the Call Tree table and the metrics for the selected call are displayed in the Metrics Inspector. When you click on a row in the Call Tree table the corresponding call box in the Call Profile graph is selected and the metrics for the selected call are displayed in the Metrics Inspector.

Note: There are differences in the layout and the metrics that are displayed in the Call Profile Window depending on the type of call profile that Diagnostics is displaying. These differences will be noted as each of the areas of the window are described.

Call Profile Graph

The following image is an example of an instance call profile graph.



The horizontal axis of the Call Profile represents elapsed time, where time progresses from left to right. For aggregated call profiles, the scale across the top of the profile denotes the total time.

For instance call profiles, the calls are distributed across the horizontal axis based upon the actual time when they occurred and so their positions help to show the sequence of each call relative to each other. The scale across the top of the instance call profile denotes the elapsed time since the server request was started.

The vertical axis of the call profile depicts the call stack depth or nesting level. Calls that are made at the higher levels of the call stack are shown at the top of the call profile and those made at deeper levels of the call stack are shown at the lower levels of the profile.

Each call box in the instance call profile represents a method call. The left edge of the box is the start time of the method call and the right edge is the return time from the call. The duration of the call is therefore represented by the length of the box. The position of the call box along the horizontal axis indicates the actual time when the call started and ended. The call boxes that appear directly beneath a call box are the child calls that are invoked by the parent call above them.

The gaps between the call boxes on a layer of the instance profile indicate one of the following processing conditions:

- ▶ The processing that took place during the gap occurred in code that is local to the parent at the previous higher level in the call profile and not in child calls in a lower layer.
- ▶ The call was waiting to acquire a lock or mutex.
- ▶ The processing that took place during the gap occurred in a child call that was not instrumented or included in a capture plan for the run.

The Critical Path in the Call Profile Graph

The path through the call profile that has the highest total latency is the critical path. Call boxes that are part of the critical path are colored red so that you can identify the methods that make up the critical path. Call boxes that represent calls that are not a part of the critical, high-latency path are colored grey.

Call Profile Graph Teletypes

If the duration of a call is very short or if the call appears further down in the call stack, the size of the call box can cause the name of the method that the call box represents to become too small to read. You can view the name of the method along with other details for a selected method by holding your pointer over the call box to cause the tooltip to be displayed. You can also see the details for a method selected from the call profile in the Metrics Inspector.

The tooltip contains the following details for the selected call box:

Method Detail	Description	Window Type
Method Name	Name of the method represented by the call box	Aggregate Instance
Layer Name	The name of the Diagnostics layer where the call occurred.	Aggregate Instance

Method Detail	Description	Window Type
Total Contribution	The percentage contribution to the total latency of the server request that the methods processing contributed.	Aggregate Instance
Call Count	The total number of times that the method was called during the execution of the aggregated server requests instances.	Aggregate
Total Latency	The cumulative latency attributed to the processing of the method.	Aggregate Instance
Average Latency	The average latency that can be attributed to each of the method executions for the aggregated server request instances.	Aggregate

Call Tree Table

The **Call Tree** table appears directly below the **Call Profile**. This table shows the same information that is represented in the **Call Profile**. The following image is an example of a Call Tree table for an aggregate profile.

Call	Average Latency	Count	Total CPU	Avg CPU
100% /estore/control/verif	84.9	268	13,860.0	51.7
99.9% MainServlet.doPo	84.8	268	950.0	3.5
99.8% MainServlet.do	84.7	268	950.0	3.5
86.6% MainServlet.	73.6	268	830.0	3.1
0% \$Proxy0.getF	0.0	11		
0% ShoppingCli	0.0	10		
0% ProfileMg	0.0	6		
0% ProfileMg	0.0	3		

The first row in the table contains the root of the call stack which is the server request on which you drilled down when the **Call Profile Window** was displayed. The children rows in the tree are the method calls that were made as a result of the server call. The method calls that are parents in the call stack have the expand / collapse control in front of them so that you can control whether the parent's children are displayed or not.

Note: When you click on a row call in the Call Tree table, the corresponding box is selected in the call profile graph and the metrics for the selected call are displayed in the Metrics Inspector.

The Call Tree Table contains the following columns:

Column Label	Description	Window Type
Call	The name of the Server Request or Method Name. The percentage contribution of the method call to the total latency of the service request precedes the name. The percentage is colored red for those calls which are on the call tree's critical path.	Aggregate Instance
Average Latency	The average latency that can be attributed to each of the method executions for the aggregated server request instances.	Aggregate
Count	The total number of times that the method was called during the execution of the aggregated server requests instances.	Aggregate
Total Latency	The cumulative latency attributed to the processing of the method.	Instance
Total CPU	The total amount of CPU time used by the processing for the selected method or server request.	Aggregate Instance
Average CPU	The average amount of CPU time used by each of the aggregated method calls included in the selected method or server request.	Aggregate

Note: The Total Latency for a parent call includes not only the sum of the latency of each of its children but also the latency for the processing that the method did on its own.

Metrics Inspector

The **Metrics Inspector** lists the metrics related to the server request or method selected in the Call Profile Graph or in the Call Tree Table. The following example shows the Metrics Inspector for the aggregated call profile.

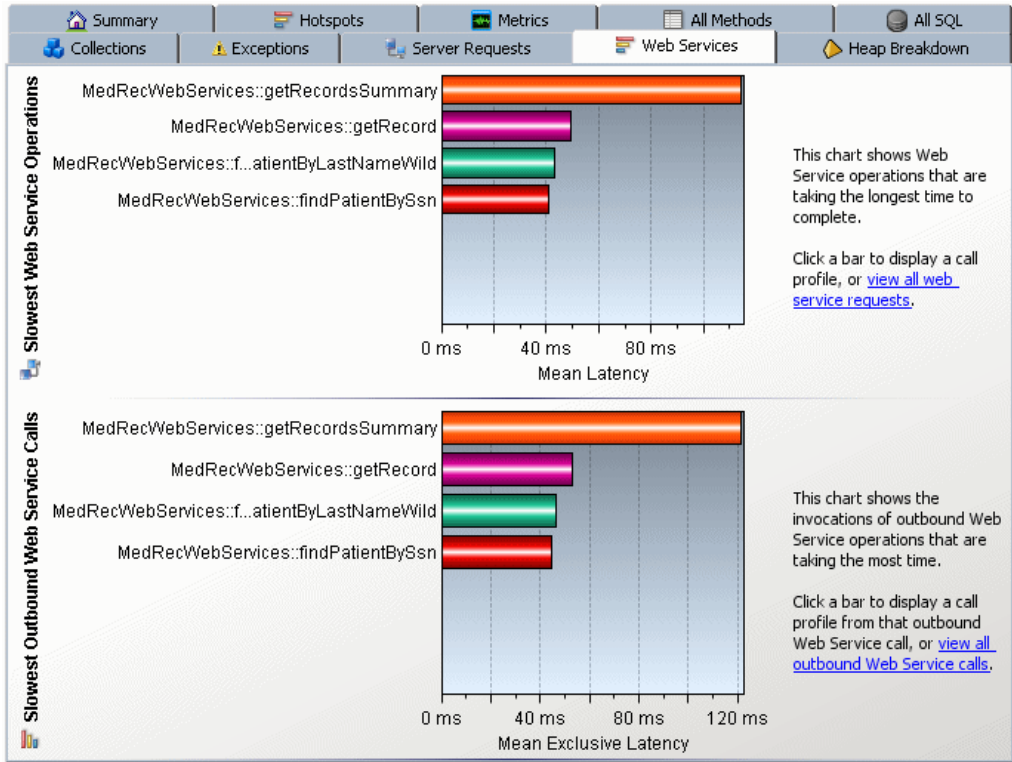
[-] /estore/control/cart	
Application Name	
Arguments	
Name	/estore/control/cart
Signature	
Root Method	
Type	
URI	/estore/control/cart
Exceptions	0
Count	1
Timeouts	0
[-] Latency	
Total CPU	80.0 ms
Exclusive Total Latency	0.1 ms
Max. Latency	258.4 ms
Min. Latency	258.4 ms
Standard Deviation	0.0 μ s
Total Latency	258.4 ms
Average Latency	258.4 ms

To view the details of a particular call in the Metrics Inspector, select the call from the Call Tree Table or in the Call Profile Graph.

The metrics that are included in a metric category can be hidden or displayed by expanding or collapsing the list of metrics using the plus sign (+) and minus sign (-) next to the category name. Alternatively, you can double-click the category name to expand or collapse the list of metrics.

Analyzing Performance Using the Web Services Tab

The Web Services tab contains graphs displaying the slowest Web service operations (inbound Web service calls) received and processed in your monitored environment and the slowest outbound Web service calls made from within your monitored environment.



Web service operations and calls are displayed in the graphs, in the following format:

<Web-service-name>::<operation-name>.

For example, MedRecWebServices::getRecordsSummary.

Understanding the Web Services Tab

The Web Services tab contains the following two graphs:

Slowest Web Service Operations Graph

The Slowest Web Service Operations graph displays the slowest Web service operations (inbound Web service calls) received and processed in your monitored environment.

The J2EE Diagnostics Profiler displays Web service operations as a type of server request.

You can view the call profile for a Web service operation displayed in the graph, by clicking the bar representing the relevant Web service operation. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

You can view a list of all the Web service operations in the Server Requests tab, by clicking the **view all web service requests** link to the right of the graph. For more information about the Server Requests tab, see “Analyzing Performance Using the Server Requests Tab” on page 244.

Slowest Outbound Web Service Calls Graph

The Slowest Outbound Web Service Calls graph displays the slowest outbound Web service calls made from within your monitored environment.

The J2EE Diagnostics Profiler displays outbound Web service calls as remote calls within a server request.

You can view the call profile for the server request containing a particular outbound Web service call displayed in the graph. To view the call profile, click the bar representing the relevant Web service call. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 247.

Note: If the remote call is part of more than one server request, when you double click the method, a dialog box opens and asks you to select the relevant server request. Double-click the appropriate server request row to view the call profile.

You can view all the outbound Web service calls in the All Methods tab, by clicking the **view all outbound Web service calls** link to the right of the graph. For more information about the All Methods tab, see “Analyzing Performance Using the All Methods Tab” on page 236.

12

Analyzing Memory with J2EE Diagnostics Profiler Screens

This chapter provides a detailed description of the screens, graphs, and tables that are used to present the J2EE memory diagnostics metrics for the application that is being analyzed.

This chapter describes:	On page:
Analyzing Memory Using the Collections Tab	257
Analyzing Memory Using the Heap Breakdown Tab	262

Analyzing Memory Using the Collections Tab

The J2EE Diagnostics Profiler can monitor your applications' memory usage using Light Weight Memory Diagnostics (LWMD). LWMD monitors the memory used by your applications by tracking collection objects.

Accessing the Collections Tab

By default, LWMD is disabled so that the J2EE Probe will not impose the additional overhead on its host when you do not need the memory diagnostics metrics.

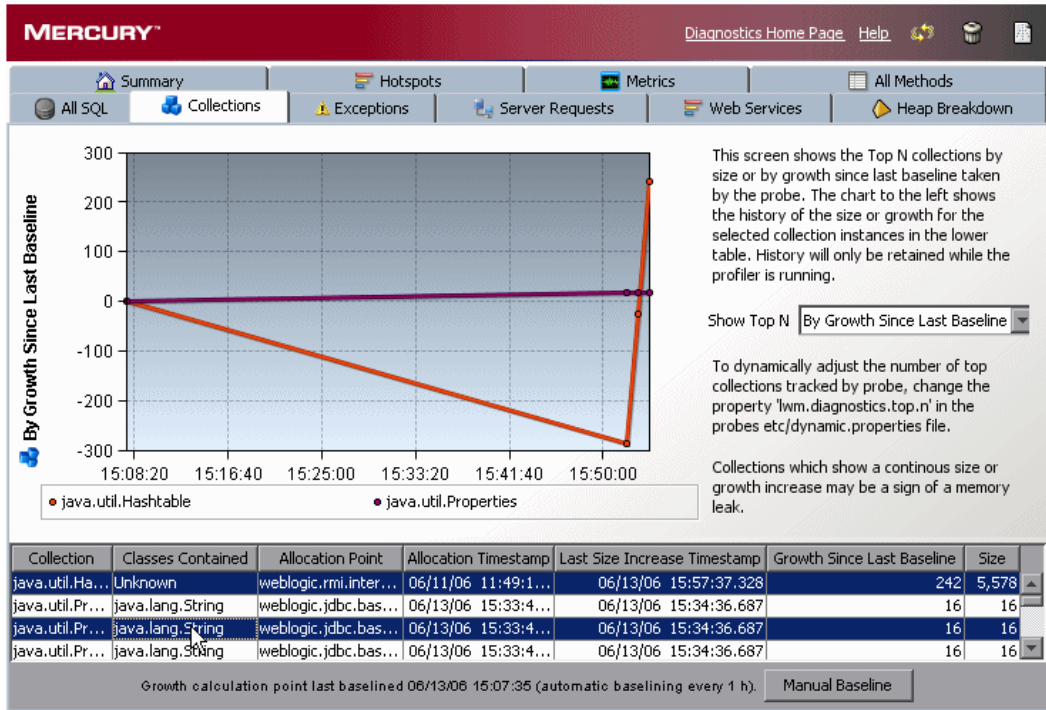
To enable LWMD and the Collections tab:

In the `auto_detect.points` file located in `<probe_install_directory>\etc`, uncomment the following 2 lines:

```
:[Light-Weight Memory Diagnostics]  
;keyword = lwmd
```

Understanding the Collections Tab

The Collections tab displays the memory metrics in a graph and a corresponding table that display the collections that are growing the fastest and that have become the largest.



You can choose to display the top collections according to either size or growth since the last baseline by selecting either **By Size** or **By Growth in Last Baseline** in the **Show Top N** box.

The Collections tab is divided into the following sections:

Collection Description Table

Collection	Classes Contained	Allocation Point	Allocation Timest...	Last Size In...	Grow...	Size ▾
java.util.Hashtable	weblogic.servlet.internal...	weblogic.servlet.internal.sess...	06/11/06 11:49:...	06/13/06 1...	158	14,828 ▲
java.util.Hashtable	java.util.Vector	weblogic.servlet.internal.Http...	06/12/06 11:18:...	06/13/06 1...	158	14,828
java.util.HashMap	weblogic.utils.collections...	weblogic.utils.collections.Soft...	06/11/06 11:49:...	06/13/06 1...	-140	14,812
java.util.TreeSet	java.io.File	weblogic.servlet.logging.Log...	06/11/06 16:55:...	06/11/06 1...	0	4,662 ▼

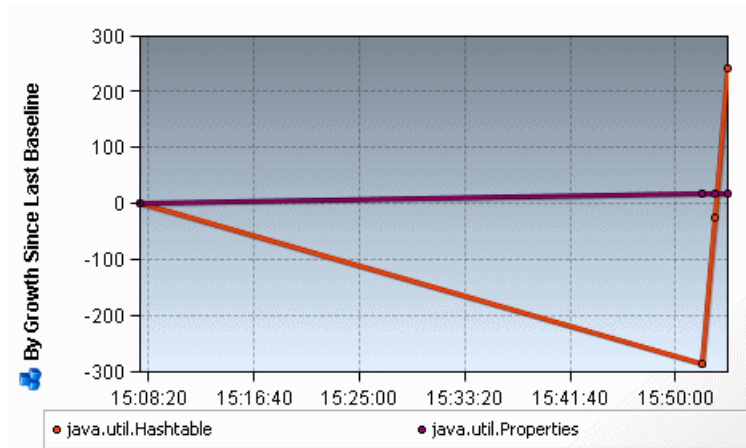
Growth calculation point last baselined 06/13/06 16:07:37 (automatic baselining every 1 h).

The Collection Description table contains the following columns:

- **Collection.** The collection type.
- **Classes Contained.** The type of the objects contained within the collection. If there are multiple types of objects found within the collections, the value in the table appears as **Unknown**.
- **Allocation Point.** The location where the collection is allocated in the code.
- **Allocation Timestamp.** The time at which the collection was allocated.
- **Last Size Increase Timestamp.** The last time that a size increase was captured.
- **Growth Since Last Baseline.** The increase or decrease in the number of objects within the collection since the last baseline.
- **Size.** The number of objects in the collection.

Collection Display Graph

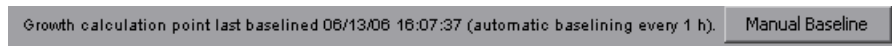
When you click a collection in the Collection Description table, the Collection Display graph shows either the size or the growth of the collection since the last baseline, depending on what you selected in the **Show Top N** box to the right of the graph.



Note: You can select more than one collection to display on the graph using the **Control** key.

Setting a New Baseline

The baseline determines the time from which the growth is going to be measured. You can view the time that the last baseline was set at the bottom of the Collections tab window.



You can force a new baseline by clicking **Manual Baseline**.

By default a new baseline is set automatically every hour. You can change the automatic baselining interval in the **dynamic.properties** file.

Note: There is no need to stop the application server when you change the automatic baselining interval.

To change the automatic baselining interval

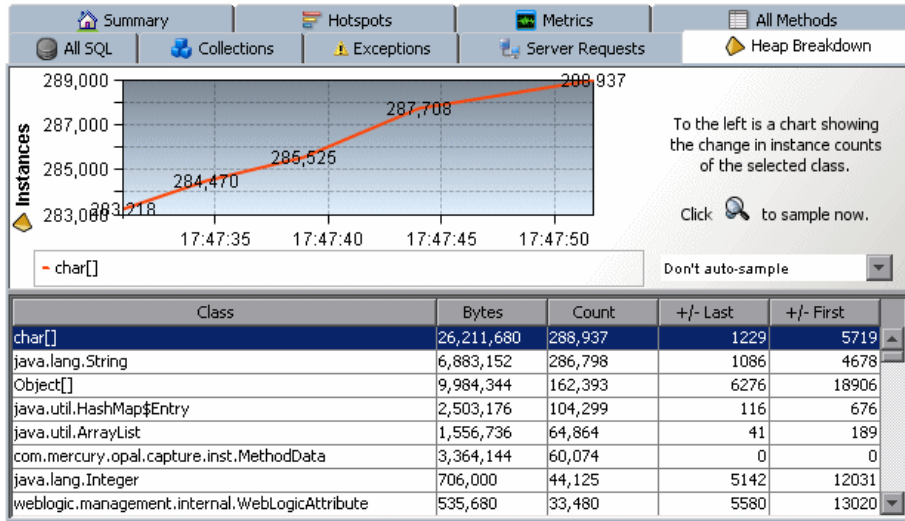
- 1** In the `<probe_installation>\etc` directory, in the `dynamic.properties` file, locate the following line:

`lwm.diagnostics.auto.baseline.interval=60m`
- 2** In that line, change the time interval according to your needs, as explained in the file.

Note: If you want to stop automatic baselining, enter **0** for the time interval.

Analyzing Memory Using the Heap Breakdown Tab

The J2EE Diagnostics Profiler can monitor your applications' memory usage by performing Heap Breakdown analysis. The **Heap Breakdown** tab displays the memory metrics in a table and in a corresponding graph.



Accessing the Heap Breakdown Tab

By default, Heap Breakdown is disabled so that the J2EE Probe will not impose additional overhead on its host when you do not need the memory diagnostics metrics.

To enable Heap Breakdown and display the Heap Breakdown tab:

- 1 Add the files in the directory `<probe_installation>\lib\<platform>` to the OS environment path.
- 2 Add `'-Xrunheapdump'` to the command line that starts the JVM and the application server.

Understanding the Heap Breakdown Tab

The Heap Breakdown tab is divided into the following sections:

Heap Metrics Table

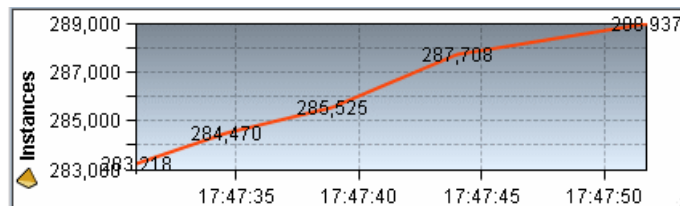
Class	Bytes	Count	+/- Last	+/- First
char[]	121,086,088	354,515	0	0
java.lang.String	8,203,872	341,828	0	0
Object[]	10,839,416	149,283	0	0
java.util.HashMap\$Entry	3,139,416	130,809	0	0
java.util.ArrayList	1,780,824	74,201	0	0
com.mercury.opal.capture.inst.MethodData	3,578,960	63,910	0	0
com.mercury.opal.capture.LWMDCaptureAgent\$LWMDDData	1,976,920	22,465	0	0
java.util.HashMap	855,640	21,391	0	0

The Heap Metrics table contains the following columns:

- **Class.** The name of the class.
- **Bytes.** Actual amount of memory, in bytes, that has been allocated or used by objects belonging to this class.
- **Count.** The number of objects belonging to this class that are stored in the JVM.
- **+/-Last.** The count change since the most recent time a heap snapshot was taken.
- **+/-First.** The count change since the initial heap snapshot was taken

Heap Breakdown Graph

When you click a class name in the Heap Breakdown table, the Heap Breakdown graph shows the count over time of objects belonging to that class.



Note: You can select more than one class to display on the graph using the **Control** key.

Sampling

The data displayed in the Heap Breakdown tab is a snapshot of the system. You can choose how frequently the system takes an automatic sample by selecting one of the autosample options in the autosampling list box next to the graph. You can manually take a sample snapshot by clicking the sample button.



Note: The Profiler does not support Heap Breakdown in JRockit. Instead, you can use BEA's memory leak detector tool, which is built into JRockit, to monitor memory usage.

Part IV

Using the Mercury Diagnostics Profiler for .NET

13

Using the .NET Diagnostics Profiler

This chapter provides an overview of the processing of the Mercury Diagnostics Profiler for .NET, descriptions of the screens, and instructions for using some of the global user interface controls.

This chapter describes:	On page:
Accessing the .NET Diagnostics Profiler	268
Mercury Diagnostics Profiler for .NET Processing	269
Common .NET Profiler Tab Navigation and Display Controls	270

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

For more information about licensing, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

If you installed the probe from the Mercury Web site and you want to use it with a Diagnostics Server, contact Mercury Support.

Accessing the .NET Diagnostics Profiler

The .NET Diagnostics Profiler is installed as part of the Mercury Diagnostics Probe for .NET (.NET Probe).

Once you have installed and configured the .NET Probe and you have started the application that is being monitored, you can access the .NET Diagnostics Profiler from your browser and view diagnostics data. You can also access the .NET Diagnostics Profiler by drilling down from the Mercury Diagnostics screens.

Important! When the .NET Probe is installed to work with a Mercury Diagnostics Server, the Mercury Diagnostics Profiler for .NET is disabled by default. To enable the .NET Diagnostics Profiler, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

To access the .NET Diagnostics Profiler from your browser:

- 1 In your browser, go to the .NET Diagnostics Profiler URL:
http://<probe_host>:<probeport>/profiler.

The probes are assigned to the first available port beginning at 35000.

- 2 Type your username and password.

You are prompted to enter a username and password.

The default username is `admin`. The default password is `admin`.

For more details about usernames and passwords, see the *Mercury Diagnostics Installation and Configuration Guide*

To access the Diagnostics Profiler from Mercury Diagnostics:

- ▶ From any Status screen in Mercury Diagnostics, right-click the probe entity and select **Open Mercury Diagnostics Profiler** from the menu.
- ▶ Alternatively, in the probe standard view screen in Mercury Diagnostics, right-click the probe entity in the detail table and select **Open Mercury Diagnostics Profiler** from the menu.

If the Profiler fails to open when performing the drill down, ensure that you have set a default browser within your operating system.

Mercury Diagnostics Profiler for .NET Processing

This section describes the way in which the .NET Probe monitors your application and how this data is displayed in the .NET Diagnostics Profiler.

Monitoring Method Latency and Call Stacks

The Mercury Diagnostics Probe for .NET (.NET Probe) monitors your application and keeps track of the metrics for all of the instrumented methods that your application calls. As it is monitoring, it captures the call stack for the three slowest instances and the single fastest instance of each server request. When a new server request instance is encountered that is slower than one of the currently captured instances for the server request, it replaces one of the previously captured instances. In the same manner the captured call stack for the fastest instance is replaced when an instance that is even faster is encountered.

The .NET Diagnostics Profiler displays metrics for all of the instrumented methods. You can drill down to the instances of the methods that were included in one of the four server request call stacks that were captured when you accessed the .NET Diagnostics Profiler user interface. While you are analyzing the information displayed on the various tabs of the .NET Diagnostics Profiler, you are working with the methods and call stacks captured from the time that the user interface was started. In the meantime the .NET Probe continues to monitor your application, capture method metrics, and capture call stacks.

For more information on monitoring method latency with the .NET Diagnostics Profiler, see Chapter 14, “Analyzing Method Latency with .NET Diagnostics Profiler Screens.”

Monitoring Application Memory

The .NET Diagnostics Profiler allows you to monitor your application's memory usage using one of the following methods:

- Light Weight Memory Diagnostics
- Heap Breakdown

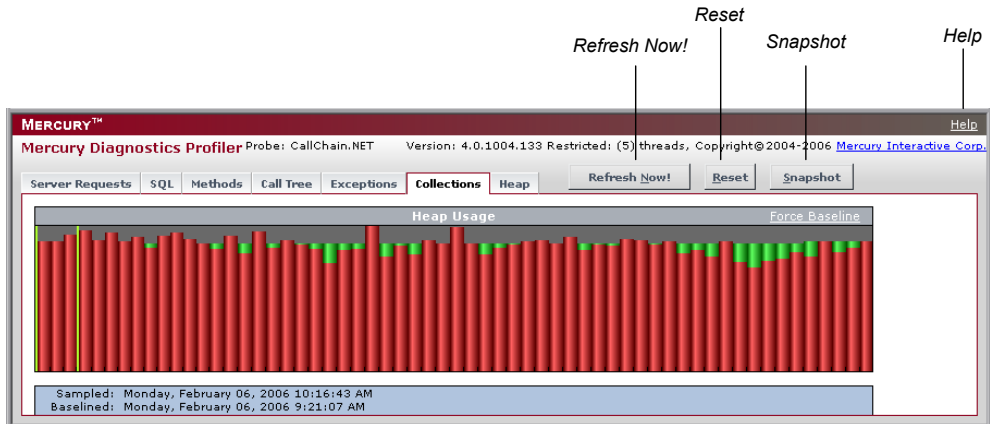
Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you to identify objects that may be leaking.

For more information about Light Weight Memory Diagnostics, see “Analyzing Memory Using the Collections Tab” on page 293.

For more information about Heap Breakdown, see “Analyzing Memory Using the Heap Tab” on page 301.

Common .NET Profiler Tab Navigation and Display Controls

This section describes the following features and controls that are common to all of the .NET Profiler tabs: **Refresh now**, **Reset**, **Snapshot** and **Help**:



Refreshing Metrics

Click **Refresh Now** to refresh the information displayed on the tabs with the latest metrics and call stacks.

A rectangular button with a light gray background and a thin border. The text "Refresh Now!" is centered in a dark gray font. The word "Now" is underlined.

After you refresh the metrics, the .NET Diagnostics Profiler continues to monitor and collect metrics using the same baseline for the calculations of instance counts, average latency, and slowest latency. It also continues to use the captured call stacks as a basis of comparison for finding new call stacks to capture.

Resetting Metrics

You can force the .NET Diagnostics Profiler to use new baselines for the calculation of instance counts, average latency, and slowest latency, and to force-drop all captured call stacks, by clicking **Reset**.

A rectangular button with a light gray background and a thin border. The text "Reset" is centered in a dark gray font.

After you reset the metrics, the .NET Diagnostics Profiler begins collecting data with new baselines and starts processing the instance trees as though the profiler had just been started.

Note: You may want to click **Reset** once your system has warmed up so that you can do your performance analysis using metrics that are more representative of the processing that takes place when your application is running in steady state.

Taking a Snapshot

You can capture a snapshot of the data from your profiler session into an .xml formatted file, by clicking Snapshot.



The resulting snapshot can be used, for example, as a report that is distributed to your colleagues or as a point of reference when you are about to make changes to your applications. The snapshot includes the profiler tabs so that you can review and analyze the data in the snapshot in the same way that you would view it in the Profiler.

The Profiler displays a dialog box that indicates the path to where the .xml file is stored. When you open the snapshot, the saved profiler data is displayed in your browser.

Accessing Help

When you click **Help**, on the top right hand corner of the screen, you access the on-line help manual for the Mercury Diagnostics Profiler for .NET.

14

Analyzing Method Latency with .NET Diagnostics Profiler Screens

This chapter provides a detailed description of the screens, graphs, and tables that are used to present the .NET performance metrics for the application that is being analyzed.

This chapter describes:	On page:
Analyzing Performance Using the Server Requests Tab	274
Analyzing Performance Using the SQL Tab	279
Analyzing Performance Using the Methods Tab	281
Analyzing Performance Using the Exceptions Tab	284
Analyzing Performance Using the Call Tree Tab	286

Analyzing Performance Using the Server Requests Tab

Introducing the Server Requests Tab

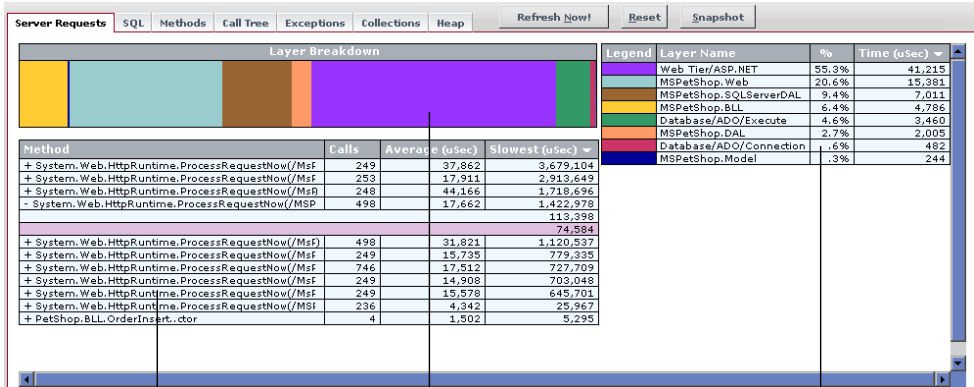
The .NET Diagnostics Profiler keeps track of all of the method calls made by your application. The **Server Requests** tab displays information about the server request methods. The server request methods are listed in a table that shows the number of times that each method was executed, along with the average latency and the slowest execution time for all of the calls to the method. You can expand each server request listed in the table, to reveal the latency for the three slowest instances of the server request along with the single fastest instance.

Note: The .NET Diagnostics Profiler captures call trees for the three slowest instances and the single fastest instance of each server request. The .NET Diagnostics Profiler lets you drill into the captured call trees from the Server Requests tab.

The Server Requests Tab at a Glance

The Server Requests tab is divided into the following sections:

- Server Request Method Table
- Layer Breakdown Table
- Layer Breakdown Graph



Server Request Method Table

Layer Breakdown Graph

Layer Breakdown Table

Server Request Method Table

The Method table lists the server requests that have been called. You can sort the table by clicking the column headers.

Method	Calls	Average (uSec)	Slowest (uSec) ▼
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	249	37,862	3,679,104
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	253	17,911	2,913,649
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	248	44,166	1,718,696
- System.Web.HttpRuntime.ProcessRequestNow(/MSPetS	498	17,662	1,422,978
			113,398
			74,584
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS)	498	31,821	1,120,537
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	249	15,735	779,335
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	746	17,512	727,709
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	249	14,908	703,048
+ System.Web.HttpRuntime.ProcessRequestNow(/MsPetS	249	15,578	645,701
+ System.Web.HttpRuntime.ProcessRequestNow(/MSPetS	236	4,342	25,967
+ PetShop.BLL.OrderInsert.ctor	4	1,502	5,295

The following columns are included in the table:

- **Method.** The server request methods that were called.

If a server request method was called more than once, the method name is preceded by a plus sign (+) or a minus sign (-) to indicate that the instance specific latency information is available for the server request.

- **Calls.** The number of times that the server request method was invoked.
- **Average.** The average latency for all of the calls to the server request method. The average latency is shown in microseconds.
- **Slowest.** The response time of the instance with the longest latency. The slowest response time is shown in microseconds.

Layer Breakdown Table

The Layer Breakdown table shows the amount of processing time that was spent in each layer while executing a selected instance of a method call. The table can be sorted by clicking the column headers.

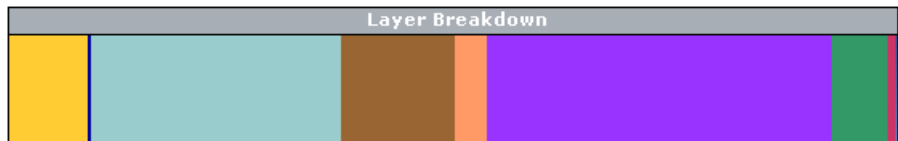
Legend	Layer Name	%	Time (uSec) ▼
	Web Tier/ASP.NET	55.3%	41,215
	MSPetShop.Web	20.6%	15,381
	MSPetShop.SQLServerDAL	9.4%	7,011
	MSPetShop.BLL	6.4%	4,786
	Database/ADO/Execute	4.6%	3,460
	MSPetShop.DAL	2.7%	2,005
	Database/ADO/Connection	.6%	482
	MSPetShop.Model	.3%	244

The following columns are included in the table:

- ▶ **Legend.** The color that is used in the Layer Breakdown graph to depict the processing that took place in the layer.
- ▶ **Layer Name.** The name of the layer where the processing for the server request took place.
- ▶ **%.** The percentage of processing time that was spent in each layer, for a selected server request.
- ▶ **Time.** The latency measured for the processing that took place in the layer, for a selected server request. The time is shown in microseconds.

Layer Breakdown Graph

The Layer Breakdown graph shows the amount of processing time that was spent in each layer while executing a selected instance of a method call. It is a graphical representation of the information shown in the Layer Breakdown table.



The graph is divided so that each layer is depicted as an area on the graph that is proportional to the percentage of processing that was performed in the layer. Each layer is displayed in a different color, as shown in the Legend column in the Layer Breakdown table.

Viewing Instance Information for Server Requests

If a server request method was called more than once, the method name in the Server Request Method table is preceded by a plus sign (+) or a minus sign (-). When you click the plus sign, the entry is expanded to reveal the three slowest instances of the method along with the single fastest method. Click the minus sign to collapse instances shown.

If a server request method was called only once, the entry listed on the Server Request Method table is not preceded by a plus or minus sign and the entry itself represents the single instance of the method call. The value in the Slowest column is the instance's latency.

Viewing the Layer Breakdown for a Server Request Instance

You can view the Layer Breakdown for a server request instance listed in the Server Request Method table by moving the mouse pointer over any row that contains an instance of a server request method call. (A row that does not have a plus sign (+) or a minus (-) sign before the method name, or that only has a latency value, is a server request instance.)

When you move the mouse pointer over a server request instance, the Profiler shows the layer breakdown information for the indicated instance in both the Layer Breakdown table and Layer Breakdown graph.

Viewing the Call Tree for a Server Request Instance

You can view the call tree for a server request instance listed in the Server Request Method table by clicking on any row that contains an instance of a server request method call. (A row that does not have a plus sign (+) or a minus (-) sign before the method name or that only contains a latency value is a server request instance.)

When you click on a row with a server request instance, the Profiler switches to the Call Tree tab and displays the call tree for the selected server request instance. The method call for the selected server request is highlighted in blue in the call tree.

For more information on the Call Tree tab, see "Analyzing Performance Using the Call Tree Tab" on page 286.

Analyzing Performance Using the SQL Tab

The .NET Diagnostics Profiler keeps track of all of the method calls that your application makes. The **SQL** tab displays the SQL methods only. The SQL methods are listed in the Method table which shows the number of times that each method was executed, along with the average latency and the slowest execution time for all of the calls to the method. The Method table also shows the actual SQL statement when it was included in the SQL method call.

Each SQL method listed in the table can be expanded to reveal the latency for each instance of the method that was included in a captured call tree.

Note: The .NET Diagnostics Profiler captures call trees for the three slowest instances and the single fastest instance of each server request. You can drill down to the captured call trees from the SQL tab.

SQL Method Table

The SQL tab contains the SQL Method table.

Method	Calls	Average (uSec)	Slowest (uSec) ▼	SQL
+ PetShop.SQLServerDAL.SQLEn	998	4,403	1,401,536	
+ System.Data.SqlClient.SqlCon	249	7,805	1,400,624	SELECT Account.Email, L...
- System.Data.SqlClient.SqlConn	498	1,996	85,143	SELECT Item.ItemIdte...
				1,327
				1,203
				1,149
				933
System.Data.SqlClient.SqlCon	1	41,496	41,496	SELECT ProductId, Category...
+ PetShop.SQLServerDAL.SQLEn	249	2,340	28,950	
+ System.Data.SqlClient.SqlCon	249	1,276	21,984	SELECT Qty FROM ItemId
+ System.Data.SqlClient.SqlCon	249	1,461	16,101	SELECT Account.Firstoun...
System.Data.SqlClient.SqlCon	1	1,558	1,558	SELECT ItemId, Attr: IN...

This table lists the SQL methods that have been called, and displays latency information for instances of the SQL method calls that were included in the captured call trees. The table can be sorted by clicking the column headers.

The following columns are included in the table:

- ▶ **Method.** The SQL methods that were called. If an SQL method has two or more instances in the captured call trees, the method name is preceded by a plus sign (+) or a minus sign (-) to indicate additional instance specific latency information can be viewed for the SQL call.
- ▶ **Calls.** The number of times that the SQL method was invoked. This count includes all instances, whether or not they are included in the captured call trees.
- ▶ **Average.** The average latency for all of the calls to the SQL method. The average latency is shown in microseconds.
- ▶ **Slowest.** The response time for the instance with the longest latency. The slowest response time is shown in microseconds.
- ▶ **SQL.** The first part of the SQL statement that was executed by the SQL method call.

Note: You can display a tooltip containing the entire SQL statement by holding the mouse pointer over a row in the SQL column.

Viewing Instance Information for SQL Method Calls

The latencies for instances of SQL methods can be displayed if they are included in one of the captured call trees.

If two or more instances of an SQL method are included in the captured call trees, that method's name is preceded by a plus sign (+) or a minus sign (-) in the Method table. The plus sign indicates that the entry can be expanded to reveal the latency for each of the captured instances for the selected method. Click the minus sign to collapse the visible instances.

If only one instance of an SQL method was included in the captured call trees, the method name in the SQL Method table is not preceded by a plus sign or minus sign. In this case the table entry itself represents the single instance of the method call, and the value in the Slowest column is the instance's latency.

If no instances of a SQL method were included in the captured call trees, the method is not preceded by a plus sign or minus sign, and when you click the method, you get a message indicating that although this method was called there is no data captured for it.

Viewing the Call Tree for an SQL Method Instance

You can view the call tree for an SQL method instance listed in the SQL Method table by clicking on any row that contains an instance of an SQL method call. (A row that does not have a plus sign (+) or a minus (-) sign before the method name, or that only contains a latency value, is an SQL instance.)

When you select a row with an SQL method instance, the Call Tree tab opens, and displays the call tree for the selected SQL method instance. The method call for the selected SQL method is highlighted in blue in the call tree.

For information on the Call Tree tab, see “Analyzing Performance Using the Call Tree Tab” on page 286.

Analyzing Performance Using the Methods Tab

The .NET Diagnostics Profiler keeps track of all of the method calls that your application makes. The **Methods** tab is used to list all of the methods. The methods are listed in the Method table, which shows the number of times each method was executed, along with the average latency and the slowest execution time for all of the calls to the method. The methods listed in the Methods tab include the server requests methods listed in the Server Requests tab, the SQL methods listed in the SQL tab, and the methods that generated exceptions shown in the Exceptions tab.

Each method listed in the table can be expanded to reveal the latency for each instance of the method that was included in one of the captured call trees. The .NET Diagnostics Profiler captures call trees for the three slowest instances and the single fastest instance of each server request. The .NET Diagnostics Profiler lets you drill down to the captured call trees from the Methods tab.

The Method Table

The Methods tab contains the Method table.

Method	Calls	Average (uSec)	Slowest (uSec) ▼
+ System.Web.HttpRuntime.ProcessRequestNow	3475	21,627	3,679,104
+ PetShop.Web.Global.Application_Error	248	14,396	1,701,324
+ PetShop.Web.OrderProcess.OnLoad	248	23,314	1,556,395
+ PetShop.Web.ProcessFlow.CartController.PurchaseCart	248	23,061	1,550,664
+ PetShop.Web.SignIn.SubmitClicked	249	11,543	1,405,137
+ PetShop.Web.ProcessFlow.AccountController.ProcessLogin	249	10,951	1,404,444
+ PetShop.BLL.Account.SignIn	249	10,094	1,403,582
+ PetShop.SQLServerDAL.Account.SignIn	249	9,545	1,403,022
+ PetShop.SQLServerDAL.SqlHelper.ExecuteReader	998	4,403	1,401,536
+ System.Data.SqlClient.SqlCommand.ExecuteReader	998	3,351	1,400,624
+ PetShop.BLL.Cart.GetOrderLineItems	248	3,466	720,331

This table lists the methods that have been called, and displays latency information for instances of the method calls that are included in the captured call trees. This table can be sorted by clicking the column headers.

The following columns are included in the table:

- ▶ **Method.** The name of the methods that were called. If a method has two or more instances included in the captured call trees, the method name is preceded by a plus sign (+) to indicate additional instance specific latency information can be viewed for the method call.
- ▶ **Calls.** The number of times that the method was invoked. This count includes all instances, whether or not they are included in the captured call trees.
- ▶ **Average.** The average latency for all of the calls to the method. The average latency is shown in microseconds.
- ▶ **Slowest.** The response time for the instance with the longest latency. The slowest response time is shown in microseconds.

Viewing Instance Information for Method Calls

You can view the latency for instances of methods if they are included in one of the captured call trees.

If two or more instances of a method are included in the captured call trees, the method name in the Method table is preceded by a plus sign (+) or a minus sign (-). The plus sign indicates that you can expand the entry to reveal the latency for each of the captured instances for the selected method. Click the minus sign to collapse the visible instances.

If no instances of a method were included in the captured call trees, the method is not preceded by a plus sign or minus sign, and when you click the method, you get a message indicating that although this method was called there is no data captured for it.

Viewing the Call Tree for a Method Instance

You can view the call tree for a method instance listed in the Method table by clicking on any row that contains an instance of a method call. (A row that does not have a plus sign (+) or a minus (-) sign before the method name, or that only contains a latency value, is a method instance.)

When you click a row with a method instance, the Call Tree tab opens and displays the call tree for the selected method instance. The method call for the selected method is highlighted in blue in the call tree.

For information on the Call Tree tab, see “Analyzing Performance Using the Call Tree Tab” on page 286.

Analyzing Performance Using the Exceptions Tab

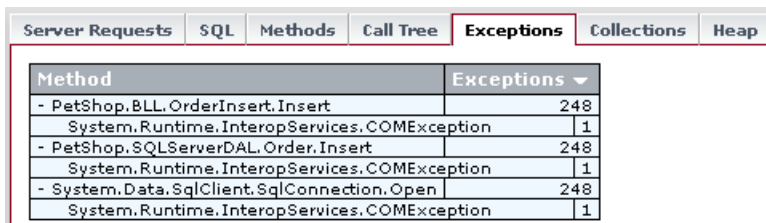
The .NET Diagnostics Profiler keeps track of all of the method calls that your application makes. The **Exceptions** tab is used to list only the methods that generated exceptions. The calling methods that generated exceptions are listed in a table that shows the number of times that each method threw an exception. This information allows you to quickly determine if your application is throwing exceptions, and exactly what those exceptions are.

If the exception was included in one of the captured call trees, the exception class will also be listed in the table along with the latency for each instance of an exception.

Note: The .NET Diagnostics Profiler captures call trees for the three slowest instances and the single fastest instance of each server request. You can drill down to the captured call trees from the Exceptions tab.

Exception Table

The **Exceptions** tab contains the Exception Method table.



Method	Exceptions
- PetShop.BLL.OrderInsert.Insert	248
System.Runtime.InteropServices.COMException	1
- PetShop.SQLServerDAL.Order.Insert	248
System.Runtime.InteropServices.COMException	1
- System.Data.SqlClient.SqlConnection.Open	248
System.Runtime.InteropServices.COMException	1

This table lists the methods calls that generated exceptions and allows you to view latency information for instances of the exceptions that were included in the captured call trees. The rows in this table can be sorted by clicking the column headers.

The following columns are included in the table:

- ▶ **Method.** The name of the methods generated exceptions. If a method generated two or more exceptions and they were included in the captured call trees, the method name is preceded by a plus sign (+) or a minus sign (-) to indicate that additional instance-specific latency information can be viewed for the exception.
- ▶ **Exceptions.** The number of times that the method generated an exception. This count includes all instances of all classes of exceptions, whether or not they are included in the captured call trees.

Viewing Instance Information for Exceptions

The latency for instances of exceptions are available to be displayed if they are included in one of the captured call trees.

If an instance of an exception for a particular method call was included in one of the captured call trees, the method name in the Exceptions table is preceded by a plus sign (+) or a minus sign (-). The plus sign indicates that when you click on the row in the table, the entry expands to reveal additional rows with the exception class for each of the captured instances of the exception. The minus sign indicates that when you click on the row in the table, the entry contracts so that the exception class row is hidden.

If two or more instances of an exception class were included in the captured call trees, the exception class name in the Exceptions table is preceded by a plus sign (+) or a minus sign (-). The plus sign indicates that when you click on the row in the table, the entry expands to reveal the latency for each of the captured instances for the selected exception class. The minus sign indicates that when you click on the row in the table, the entry contracts so that the latency for the captured exception class is hidden.

If only one instance of an exception class was included in the captured call trees, the exception class in the Exceptions table is not preceded a plus sign or minus sign. In this case, the table entry itself represents the single instance of the exception class and the value in the latency for the exception can be determined from the Call Trees tab.

Viewing the Call Tree for an Exception

You can view the call tree for an exception listed in the Exceptions table by clicking on any row that contains an instance of an exceptions class. (A row that does not have a plus sign (+) or a minus (-) sign before the exception class or that only contains a latency value is an exception class instance.)

When you click on a row with an exception class instance, the profiler switches to the Call Tree tab and displays the call tree for the selected exception instance. The method call that generated the exception for the selected exception class is highlighted in blue in the call tree.

For information on the Call Tree tab, see “Analyzing Performance Using the Call Tree Tab” on page 286.

Analyzing Performance Using the Call Tree Tab

Introducing the Call Tree Tab

The .NET Diagnostics Profiler captures call trees for the three slowest instances and the single fastest instance of each server request. The captured server request call trees are displayed on the **Call Tree** tab, in the Call Breakdown graph and in the Call Tree table.

As you analyze the methods presented on the Server Requests, SQL, Exceptions, and Methods tabs, you navigate to the Call Tree tab to understand the context of the processing associated with particular instances of the method’s execution. The call tree allows you to see the calling and the callee methods for the method of interest as well as the contribution of those methods to the measured latency.

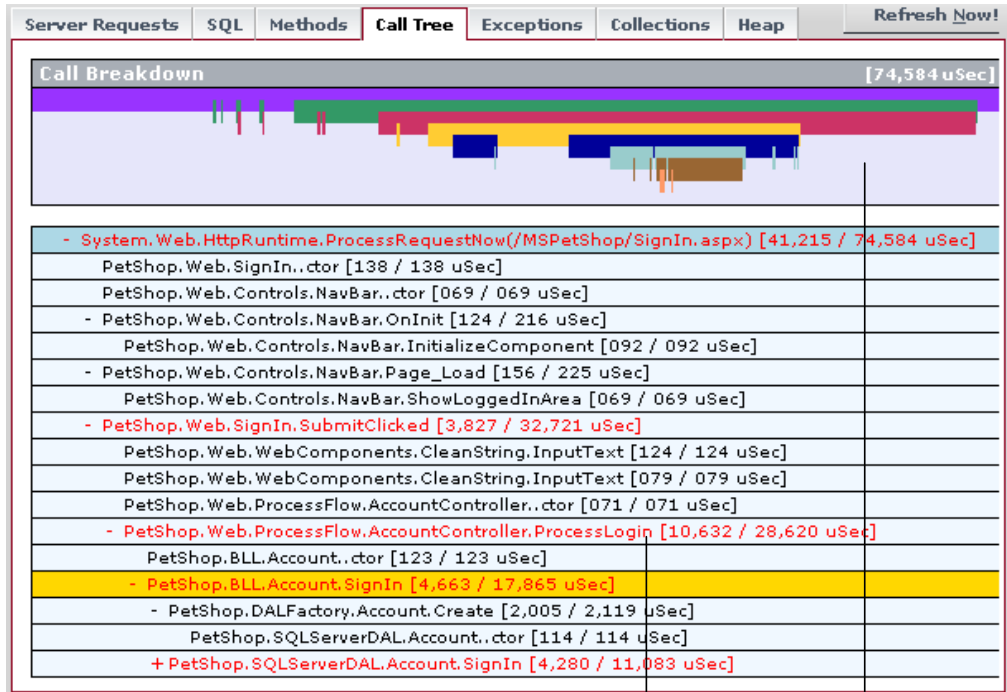
Accessing the Call Tree Tab

You can access the **Call Tree** tab directly by clicking the tab or by clicking one of the method instances listed on the Server Requests, SQL, Exceptions, and Methods tabs. For information on accessing the Call Tree tab from one of the other tabs, see the description for the tab in this chapter.

The Call Tree Tab at a Glance

The Call Tree tab is divided into the following sections:

- Call Breakdown Graph
- Call Tree Table

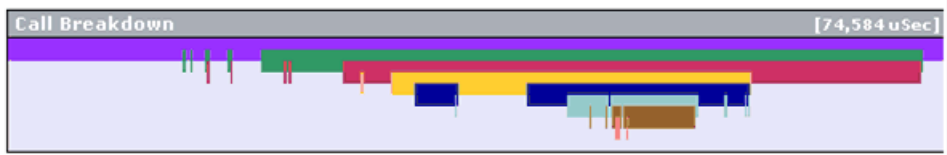


Call Tree
Table

Call Breakdown
Graph

Call Breakdown Graph

The Call Breakdown graph shows the processing time that was spent at each level of the call tree hierarchy.



Each level in the graph represents the processing at the corresponding level in the call stack. The length of the bar is proportional to the length of time spent in performing the methods at that level of the call stack. The positions where a bar starts and stops indicates the relative time, in relationship to the other levels, that the processing for the level began and ended. A gap in a bar, where the bar ends and then resumes again, indicates that the processing returned to a higher level in the hierarchy before once again proceeding at the lower level.

There are two ways that you may identify the method associated with a particular location on the Call Breakdown graph as you mouse over the bars in the graph.

- ▶ As you slide the pointer along a bar in the graph, a tooltip is displayed with the name of the method associated with each segment of the graph bar.
- ▶ As you slide the pointer along a bar in the graph, the Call Tree table scrolls so that the method associated with the selected location in the graph is displayed in the table. The row that contains the selected method is highlighted in gold.

Call Tree Table

The Call Tree table lists method calls that are part of a captured server request call tree in a hierarchical structure.

- System.Web.HttpRuntime.ProcessRequestNow(/MSPetShop/SignIn.aspx) [41,215 / 74,584 uSec]
PetShop.Web.SignIn..ctor [138 / 138 uSec]
PetShop.Web.Controls.NavBar..ctor [069 / 069 uSec]
- PetShop.Web.Controls.NavBar.OnInit [124 / 216 uSec]
PetShop.Web.Controls.NavBar.InitializeComponent [092 / 092 uSec]
- PetShop.Web.Controls.NavBar.Page_Load [156 / 225 uSec]
PetShop.Web.Controls.NavBar.ShowLoggedInArea [069 / 069 uSec]
- PetShop.Web.SignIn.SubmitClicked [3,827 / 32,721 uSec]
PetShop.Web.WebComponents.CleanString.InputText [124 / 124 uSec]
PetShop.Web.WebComponents.CleanString.InputText [079 / 079 uSec]
PetShop.Web.ProcessFlow.AccountController..ctor [071 / 071 uSec]
- PetShop.Web.ProcessFlow.AccountController.ProcessLogin [10,632 / 28,620 uSec]
PetShop.BLL.Account..ctor [123 / 123 uSec]
- PetShop.BLL.Account.SignIn [4,663 / 17,865 uSec]
- PetShop.DALFactory.Account.Create [2,005 / 2,119 uSec]
PetShop.SQLServerDAL.Account..ctor [114 / 114 uSec]
+ PetShop.SQLServerDAL.Account.SignIn [4,280 / 11,083 uSec]

Call Tree Methods

Each method in the call tree is depicted on a separate line containing two parts: the method name and the latency.

The latency for each method is shown in brackets following the method name. There are two numbers in the brackets separated by a slash: the exclusive latency and the total latency.

- Exclusive Latency is the amount of latency that is attributable to just the processing in the selected method.
- Total Latency is the amount of latency that is attributable to the selected method and all of its callee methods.

For the method in the following example, the exclusive latency is 156 and the total latency is 225.

- PetShop.Web.Controls.NavBar.Page_Load [156 / 225 uSec]
--

Method of Interest in the Call Tree

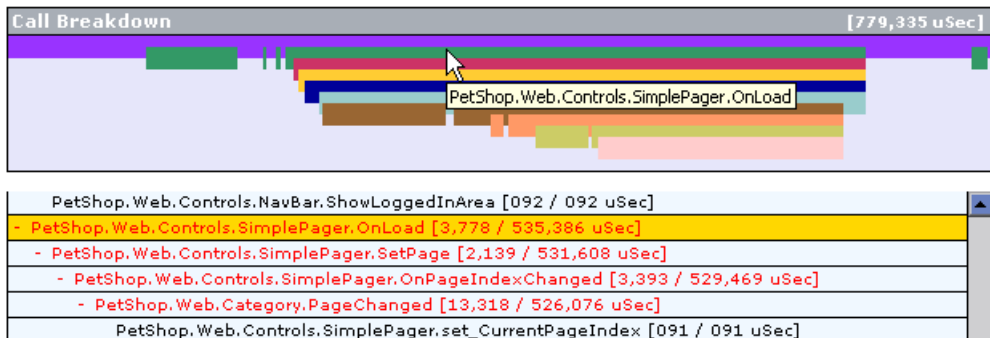
To see a captured call tree on the Call Tree tab you must select a method instance from one of the other .NET Diagnostics Profiler tabs. When you select an method instance from a tab the Call Tree tab opens with the call tree that contains the selected instance scrolled so that the selected method is visible. The selected method instance is highlighted in blue as shown in the following example:

- PetShop.BLL.Product.GetProductsByCategory [5,502 / 456,258 uSec]
- PetShop.DALFactory.Product.Create [63,361 / 63,468 uSec]
PetShop.SQLServerDAL.Product.ctor [107 / 107 uSec]
- PetShop.SQLServerDAL.Product.GetProductsByCategory [41,028 / 387,288 uSec]
PetShop.SQLServerDAL.SQLHelper.ctor [6,847 / 6,847 uSec]
- PetShop.SQLServerDAL.SQLHelper.ExecuteReader [21,379 / 335,257 uSec]
System.Data.SqlClient.SqlConnection.set_ConnectionString [27,044 / 27,044 uSec]

The method of interest will remain highlighted until a different method is selected on one of the other tabs.

Call Breakdown Methods in the Call Tree

You may identify the method associated with a particular location on the Call Breakdown graph by mousing over the bars in the graph. As you slide the pointer along a bar in the graph, the Call Tree table scrolls so that the method associated with the selected location in the graph is displayed in the table. The row that contains the selected method is highlighted in gold, as shown in the following example:



The row remains highlighted until another location in the Call Breakdown graph is selected.

Critical Path Methods in the Call Tree

The path through the call tree that has the longest latency is called the *critical path*. Methods in the Call Tree table that are on the critical path are written using a red font as shown in the following example:

- PetShop.Web.Controls.NavBar.Page_Load [2,061 / 2,153 uSec]
PetShop.Web.Controls.NavBar.ShowLoggedInArea [092 / 092 uSec]
- PetShop.Web.Controls.SimplePager.OnLoad [3,778 / 535,386 uSec]
- PetShop.Web.Controls.SimplePager.SetPage [2,139 / 531,608 uSec]
- PetShop.Web.Controls.SimplePager.OnPageIndexChanged [3,393 / 529,469 uSec]
- PetShop.Web.Category.PageChanged [13,318 / 526,076 uSec]
PetShop.Web.Controls.SimplePager.set_CurrentPageIndex [091 / 091 uSec]

15

Analyzing Memory Using .NET Diagnostics Profiler Screens

This chapter provides a detailed description of the screens, graphs, and tables that are used to present the .NET memory diagnostics metrics for the application that is being analyzed.

This chapter describes:	On page:
Analyzing Memory Using the Collections Tab	293
Analyzing Memory Using the Heap Tab	301

Analyzing Memory Using the Collections Tab

Introducing the Collections Tab

The .NET Diagnostics Profiler can monitor your applications' memory usage using Light Weight Memory Diagnostics (LWMD). LWMD monitors the memory used by your applications by tracking the collections. The metrics from LWMD are displayed on the **Collections** tab. The memory metrics are shown in a graph of heap usage, and in tables that list the collections that are growing the fastest and that have become the largest. The Collection tab displays these problems, enabling identification of memory issues.

Accessing the Collections Tab

By default, LWMD and the Collections tab are disabled so that the .NET Probe will not impose the additional overhead on its host when you do not need the memory diagnostics metrics.

To enable LWMD and the Collections tab:

- 1 Edit the probe configuration file, `<probe_install_dir>/etc/probe.config.xml`. The `lwmd enabled` property must be set to true, as shown in the following example:

```
<probeconfig>
...
  <lwmd enabled="true" sample="1m" autobaseline="1h" growth="10"
size="10"/>
...
</probeconfig>
```

- 2 Edit the points file, `<probe_install_dir>/etc/ASP.NET.points`. The LWMD section in the points file must be uncommented, as shown below:

```
[LWMD]
keyWord   = lwmd
```

The Collections Tab at a Glance

The Collections tab is divided into the following sections:

- Heap Usage Graph
- Sample and Collection Detail Table
- Collections by Growth Table
- Collections by Size Table

The screenshot shows the 'Collections' tab in the .NET Diagnostics Profiler. At the top, there are navigation tabs: Server Requests, SQL, Methods, Call Tree, Exceptions, Collections (selected), and Heap. Below the tabs is a 'Heap Usage' graph with a 'Force Baseline' button. The graph consists of 15 vertical bars, each with a red base and a green top. Below the graph is a text box with the following information:

Sampled: Sunday, November 27, 2005 5:20:08 PM
 Baselined: Sunday, November 27, 2005 4:55:06 PM
 Contains: ProductInfo
 Allocated In: System.VoidPetShop.Web.Controls.SimplePager.OnDataBinding(System.EventArgs)

Below the text box are two tables:

Collections by Growth	
Growth	Class
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.Hashtable
2	System.Collections.ArrayList
2	System.Collections.ArrayList
2	System.Collections.ArrayList

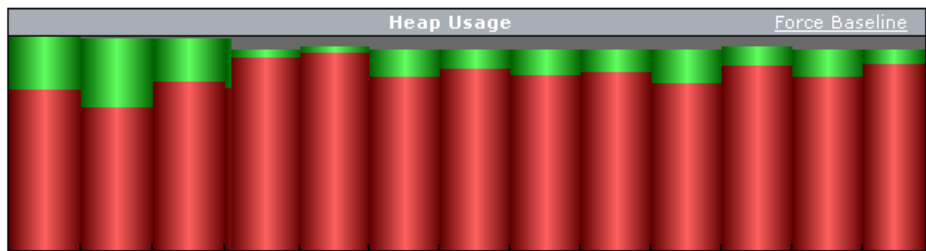
Collections by Size	
Size	Class
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.Hashtable
2	System.Collections.ArrayList
2	System.Collections.ArrayList
2	System.Collections.ArrayList
2	System.Collections.ArrayList

Labels at the bottom of the screenshot point to different sections:

- Sample and Collection Data table (points to the text box)
- Collection by Growth table (points to the 'Collections by Growth' table)
- Collection by Size table (points to the 'Collections by Size' table)
- Heap Usage graph (points to the 'Heap Usage' graph)

Heap Usage Graph

The Heap Usage graph shows the memory that was committed and used at periodic sample intervals. (The default sample interval is 1 minute.) For each sample interval, a bar is displayed on the graph.



- ▶ The height of the bar indicates the total amount of heap that was committed when the sample was taken.
- ▶ The red portion of the bar indicates the amount of the heap that was committed and used when the sample was taken.
- ▶ The green portion of the bar indicates the amount of the heap that was committed, but not used, when the sample was taken.

The Heap Usage graph controls the information displayed in the Sample and Collections detail table and in the collection tables.

To see the details for a Heap Usage sample:

In the Heap Usage graph, hold the mouse pointer over that sample's bar.

A tooltip is displayed showing the size of the heap that was used, followed by the size of the heap that was committed for the selected sample.

The information displayed in the Collections by Growth table and the Collections by Size table changes to reflect the collection information for the selected sample.

Sample and Collection Detail Table

The Sample and Collection detail table displays additional information about the sample selected in the Heap Usage graph, and about the collection selected from the collection tables.

Sampled:	Sunday, November 27, 2005 5:20:08 PM
Baselined:	Sunday, November 27, 2005 4:55:06 PM
Contains:	ProductInfo
Allocated In:	System.VoidPetShop.Web.Controls.SimplePager.OnDataBinding(System.EventArgs)

It contains the following information:

- ▶ **Sampled.** The date and time when the selected Heap Usage sample was taken.
- ▶ **Baselined.** The date and time of the last baseline prior to the sample being taken.
- ▶ **Contains.** The type of object contained in the selected collection.
- ▶ **Allocated In.** The method that allocated the selected collection.

Collections by Growth Table

The Collections by Growth table lists the top ten collections in relation to the growth in the number of objects contained in the collection since the last baseline. The top-ten list of collections changes from sample to sample as the growth rates for each collection fluctuate. When a new baseline is established, the growth rate is calculated in relation to the new baseline, so the list of collections can change significantly.

Collections by Growth	
Growth	Class
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.Hashtable
2	System.Collections.ArrayList
2	System.Collections.ArrayList
2	System.Collections.ArrayList

The table contains the following information:

- **Growth.** The number of objects that were added to the collection since the last baseline.
- **Class.** The class name for the collection.

Collections by Size Table

The Collections by Size table lists the top ten collections relative to the size of the collection for the selected Heap Usage sample. The size of a collection is based upon the total number of objects in the collection.

Collections by Size	
Size	Class
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.ArrayList
4	System.Collections.Hashtable
2	System.Collections.ArrayList
2	System.Collections.ArrayList
2	System.Collections.ArrayList

The table contains the following information:

- **Size.** The total number of objects in the collection at the end of the sample period.
- **Class.** The class name for the collection.

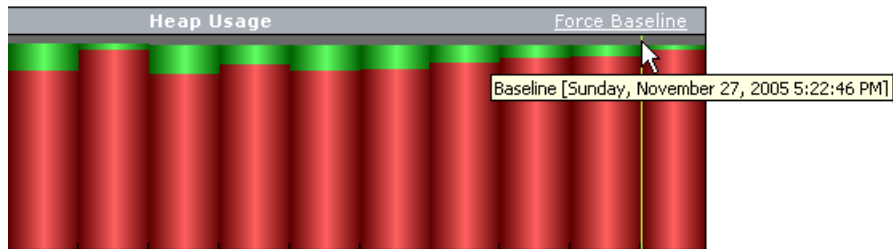
Viewing Details for a Selected Collection

To view the details for a collection listed in the Collection by Growth table or the Collections by Size table, hold the mouse pointer over the row for that collection in the table. The row is highlighted in pink, and the details for the collection are displayed in the Samples and Collections details table.

Setting a New Baseline

By default, the LWMD process establishes a new baseline for measuring the growth of collections every hour. You can force a new baseline to be set by clicking the **Force Baseline** link at the upper-right corner of the Heap Usage graph.

When the .NET Diagnostics Profiler establishes a new baseline, a green line is inserted between the last sample of the previous baseline and the first sample of the next baseline to mark the point where the baseline was set.



The calculation for the growth of collections that is used to determine which collections are included in the Collections by Growth table, is based on the number of collections added since the last baseline.

Analyzing Memory Using the Heap Tab

The .NET Diagnostics Profiler can monitor your applications' memory usage by performing Heap Breakdown analysis. The metrics from the Heap Breakdown are displayed on the **Heap** tab. The memory metrics are shown in a graph that breaks down heap usage by generation, and in a table that shows objects that are stored in the heap during the last sample. Using the Heap tab you can get an understanding of how the heap is being used by your application, and if memory is being leaked.

Accessing the Heap Tab

By default, Heap Breakdown and the Heap tab are disabled so that the .NET Probe will not impose additional overhead on its host when you do not need the memory diagnostics metrics

To enable Heap Breakdown and display the Heap tab:

Edit the probe configuration file, `<probe_install_dir>/etc/probe.config.xml`, to add an attribute named **monitorheap** to each of the processes for which you want to monitor the heap.

Add the **monitorheap** attribute to the relevant processes, as shown in the following example:

```
<probeconfig>
...
  <process name="ASP.NET" monitorheap="true">
...
</probeconfig>
```

The Heap Tab at a Glance

The Heap tab is divided into the following sections:

- Heap Breakdown by Generation Graph
- Heap Metrics Table
- Heap Sample Object Detail Table.

Server Requests SQL Methods Call Tree Exceptions Collections Heap

Heap Metrics: Sunday, November 27, 2005 3:32:36 PM			
Section	Count	Size (bytes)	
Gen 0	277140	20994836	
Gen 1	3010	18902812	
Gen 2	82462	45304828	
Large	22	4119024	
Committed	-----	89321500	
Total	362634	89321500	

Heap Breakdown by Generation

Class	Count	Size (bytes)
System.Byte[]	6352	85110908
System.String	74670	5922332
System.Collections.Hashtable, bucket[]	3605	745032
System.Object[]	12692	674332
System.Int32	30091	361092
System.Char[]	2751	337628
System.Collections.ArrayList	9789	234936
System.Int32[]	7124	221144
System.Int64[]	1405	217492
System.Collections.Hashtable	3524	183248
System.String[]	6846	165340
System.WeakReference	9972	159552
System.Net.Sockets.OverlappedAsyncResult	1252	150240
System.Collections.Specialized.NameValueCollectionBase.NameObjectEntry	6116	97856
System.Net.ConnectStream	906	94224
System.Net.HttpWebRequest	453	92412
System.Web.Configuration.CapabilitiesAssignment	4275	85500
System.Web.Configuration.CapabilitiesPattern	4946	79136
System.Web.UI.LiteralControl	899	68324
System.Net.WebHeaderCollection	946	56760
Microsoft.Win32.NativeMethods.PERF_COUNTER_DEFINITION	1171	56208
Mercury.Capture.Data.SymbolTable.Symbol	3218	51488
System.Collections.Specialized.NameValueCollection	946	49192
System.Net.NestedSingleAsyncResult	567	47628
System.Net.LazyAsyncResult	910	47320

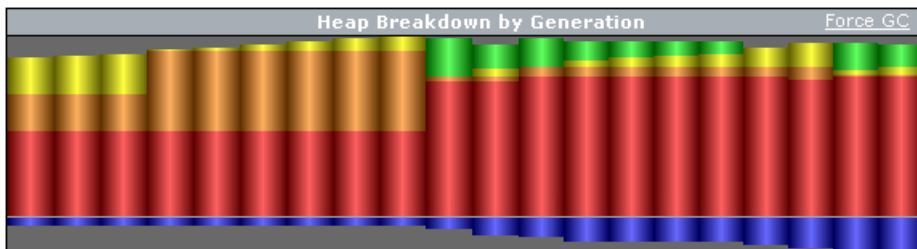
Heap Metrics
table

Heap Sample
Object Detail
table

GC Heap
Breakdown by
Generation graph

Heap Breakdown by Generation Graph

The Heap Breakdown by Generation graph shows the memory that was committed and used at periodic sample intervals. (The default sample interval is 1 minute.)



For each sample interval, a bar is displayed on the graph. The height of the bar indicates the total amount of memory that was committed during the sample period.

The bars in the chart are aligned so that the amount of memory committed to the Heap is shown extending upwards towards the top of the graph, and the amount of memory committed to the Large Object Heap is shown extending downwards towards the bottom of the graph.

The color of the bars is used to indicate the portion of the committed and used memory that is allocated to each generation of the heap. The legend in the Heap Metrics table describes the meaning of each color in the graph.

The Heap Breakdown by Generation graph controls the information displayed in the Heap Metrics table. To see the details for a Heap Breakdown sample, hold the mouse pointer over the bar for that sample in the Heap Breakdown by Generation graph.

Force Garbage Collection

When you want to deallocate used memory, you can forcibly perform garbage collection inside the application server by clicking the **Force GC** link at the upper-right corner of the Heap Breakdown by Generation graph.

Heap Metrics Table

The Heap Metrics table displays the details for the sample selected from the Heap Breakdown by Generation graph.

Heap Metrics: Sunday, November 27, 2005 3:32:36 PM		
Section	Count	Size (bytes)
Gen 0	277140	20994836
Gen 1	3010	18902812
Gen 2	82462	45304828
Large	22	4119024
Committed	-----	89321500
Total	362634	89321500

The table contains the following information:

- ▶ **Sample Heading.** The date and time when the selected Heap Breakdown sample was taken.
- ▶ **Section.** Indicates the area of memory that is applicable to the metrics that are reported in the table row.
- ▶ **Count.** The number of objects that are stored.
- ▶ **Size.** Actual amount of memory, in bytes, that has been allocated or used.

Heap Sample Object Detail Table

The Heap Sample Object detail table lists the objects that were found in the heap when the most recent sample was taken. This table does not show objects for earlier samples. The table can be sorted by clicking the column headers.

Class	Count	Size (bytes) ▼
System.Byte[]	6352	85110908
System.String	74670	5922332
System.Collections.Hashtable.bucket[]	3605	745032
System.Object[]	12692	674332
System.Int32	30091	361092
System.Char[]	2751	337628
System.Collections.ArrayList	9789	234936
System.Int32[]	7124	221144
System.Int64[]	1405	217492
System.Collections.Hashtable	3524	183248
System.String[]	6846	165340
System.WeakReference	9972	159552
System.Net.Sockets.OverlappedAsyncResult	1252	150240
System.Collections.Specialized.NameObjectCollectionBase.NameObjectEntry	6116	97856
System.Net.ConnectStream	906	94224
System.Net.HttpWebRequest	453	92412
System.Web.Configuration.CapabilitiesAssignment	4275	85500
System.Web.Configuration.CapabilitiesPattern	4946	79136
System.Web.UI.LiteralControl	899	68324
System.Net.WebHeaderCollection	946	56760

The table contains the following information:

- **Class.** The class name for the objects that were found in the heap.
- **Count.** The number of objects of the specified class found in the heap.
- **Size.** The amount of memory, in bytes, that has been used storing the objects of the specified class.

Part V

Diagnostics Integration with Other Mercury Products

16

Viewing Diagnostics Data in Mercury Business Availability Center

This chapter explains how to view Diagnostics performance data from within Mercury Business Availability Center.

This chapter describes:	On page:
About Viewing Diagnostics Data in Mercury Business Availability Center	310
Accessing the Diagnostics Screens	311
Monitoring Diagnostics Performance Data from Dashboard	311
Drilling down to Diagnostics from Dashboard	319
Diagnostics Performance Reports in Mercury Business Availability Center	323
Drilling down to Diagnostics Data from Mercury Business Availability Center Reports	327

About Viewing Diagnostics Data in Mercury Business Availability Center

Before viewing Diagnostics data in Mercury Business Availability Center, you need to configure the Diagnostics Server and the relevant Mercury Business Availability Center components according to the guidelines set out in the *Mercury Diagnostics Installation and Configuration Guide*.

From within Mercury Business Availability Center, you can actively track the performance status of your applications that are being monitored by Mercury Diagnostics. The Diagnostics integration with Mercury Business Availability Center allows you to:

- ▶ access the Mercury Diagnostics screens from within Mercury Business Availability Center.
- ▶ drill down to Diagnostics data from specific Mercury Business Availability Center configuration items and reports.
- ▶ generate high level reports in Mercury Business Availability Center about the performance of applications and Business Process Monitor (BPM) transactions that are monitored by Diagnostics.

Accessing the Diagnostics Screens

You can access the Mercury Diagnostics screens in one of the following ways:

- ▶ Select **Applications > Diagnostics**.
- ▶ On the **Site Map**, click the **Diagnostics** link.

You can also drill down to Diagnostics from specific configuration items and reports. For more information, see:

- ▶ “Drilling down to Diagnostics from Dashboard” on page 319
- ▶ “Drilling down to Diagnostics Data from Mercury Business Availability Center Reports” on page 327

Note: Mercury Business Availability Center displays Diagnostics by means of a signed applet. The Java version that runs the applet is J2SE Runtime Environment 1.4.2 or later. If the Java version is not installed on the machine, Mercury Business Availability Center opens the J2SE Runtime Environment installation wizard. Follow the instructions in the wizard to install the J2SE Runtime Environment.

For detailed information about using the Mercury Diagnostics screens, see Part II, “Using Mercury Diagnostics.”

Monitoring Diagnostics Performance Data from Dashboard

From Mercury Business Availability Center’s Dashboard, you can actively track the performance status of your applications that are being monitored by Mercury Diagnostics.

Monitoring Performance Status Using KPIs

You track performance status in Dashboard, using Key Performance Indicators (KPIs). The Dashboard KPIs provide quantifiable measurements that help you monitor how well your application is achieving its objectives. The KPIs provide real-time assessment of the present status of you application, enable you to track critical performance variables over time, and help assess the impact of problems in the application.

A color-coded icon (LED) is displayed in Dashboard for each KPI, representing the performance status assigned to that component for its current performance level.

Diagnostics related KPIs are known as Application KPIs. Application KPIs reflect the status of:

- ▶ Diagnostics probes and probe groups.
- ▶ Business Process Monitor (BPM) transactions that are monitored by Diagnostics.

In the following example of a screen in a BPM related view in Dashboard, the KPIs in the **Application** column display the Diagnostics performance status for each transaction:

The screenshot shows a dashboard interface with a menu bar at the top containing 'Top View', 'Console', 'Filters', 'Geographical Map', 'Custom Map', 'Topology Map', and 'Reports'. Below the menu bar, there is a dropdown menu for 'diagnostics_tests' and three status indicators: 'Performance' (green), 'Availability' (green), and 'Application' (green). The main content is a table with the following columns: 'CI Name', 'Performance', 'Availability', and 'Application'. The 'Application' column is highlighted with a red box. The table contains seven rows of data, each representing a different transaction type.

CI Name	Performance	Availability	Application
buy_cat	● ●	● ●	● ●
buy_dog	● ●	● ●	● ●
buy_parrot	● ●	● ●	● ●
checkout	● ●	● ●	● ●
login	● ●	● ●	● ●
pen7001	● ●	● ●	● ●
search_bird	● ●	● ●	● ●
sign_out	● ●	● ●	● ●

Last Update - 01:14:36 PM

Understanding Application KPI Status Information

The status reflected by the Application KPI is defined by specific thresholds, which you set in the Mercury Diagnostics application.

- ▶ For Diagnostics probes and probe groups, the Application KPI status is defined by all the probe related thresholds, including the server request thresholds and probe metrics thresholds.
- ▶ For BPM transactions that are monitored by Diagnostics, the Application KPI status is defined by the average latency of transaction thresholds.

When you hold your pointer over an Application KPI, a tooltip is displayed with details about the KPI status.

In the following example of a screen in a BPM related view in Dashboard, a tooltip displays the Application KPI status information for a specific BPM transaction.

The screenshot displays the Mercury Business Availability Center dashboard for the 'MedRec BPM profile'. The interface includes tabs for 'Top View', 'Console', 'Filters', 'Geographical Map', 'Custom Map', 'Topology Map', and 'Reports'. The 'Console' tab is active, showing a table of KPIs for 'Performance', 'Availability', 'Application', and 'Ack'. A tooltip is displayed over the 'Application' KPI for the 'admin_login' transaction, providing detailed status information.

Name	Performance	Availability	Application	Ack
admin_login	OK	OK	OK	OK
admin_login	OK	OK	OK	OK
edit_p...	OK	OK	OK	OK
patient...	OK	OK	OK	OK
physici...	OK	OK	OK	OK
Simple...	OK	OK	OK	OK

Details - Application

CI name: admin_login
Status: OK
Calculation Rule: Diagnostics for J2EE/.Net General
Description: No threshold violations
Platform: UNUSED
Server time: 0.218294 sec
Server Requests count: 1.0
Exceptions count: 0
Timeout count: 0

Last Update - 10:57:50 AM

The Application KPI status is explained in the following tooltip fields:

- ▶ **Status.** Can be defined as **OK**, **Warning** or **Critical**.
- ▶ **Description.** Describes the reason for the status.

For example, a **Critical** status for a transaction, may be explained in the **Description** field as follows: **15% violation on latency**. This would indicate that the average latency of the transaction exceeded the threshold that was set in Diagnostics by 15% and therefore the status of this transaction is defined as critical.

Note: The **Description** field in the Application KPI tooltip appears in Mercury Business Availability Center 6.1 and later.

The Application KPI tooltip also includes the following fields:

- ▶ **Server Time** (BPM Transaction tooltips only). The average time taken for the server to process the transaction.
- ▶ **Average Time** (probe tooltips only). The average latency of all the server requests on the VM monitored by the probe over the last five minute period.
- ▶ **Exceptions Count.** The amount of exceptions generated over the last five minute period.
- ▶ **Timeout Count.** The amount of timeouts that occurred during the last five minute period

For information about setting thresholds in Mercury Diagnostics, see “Setting Metric Thresholds” on page 53.

For more information about working with KPIs in Dashboard, refer to the section on “Understanding KPI Status” in *Using Dashboard* in the *Mercury Business Availability Center Documentation Library*.

Note: In Mercury Business Availability Center 6.1, you need to enable Application KPIs for BPM related views, before you can start using them to view the status of BPM transactions monitored by Diagnostics. For more information, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

Monitoring Diagnostics Probe Data from Dashboard

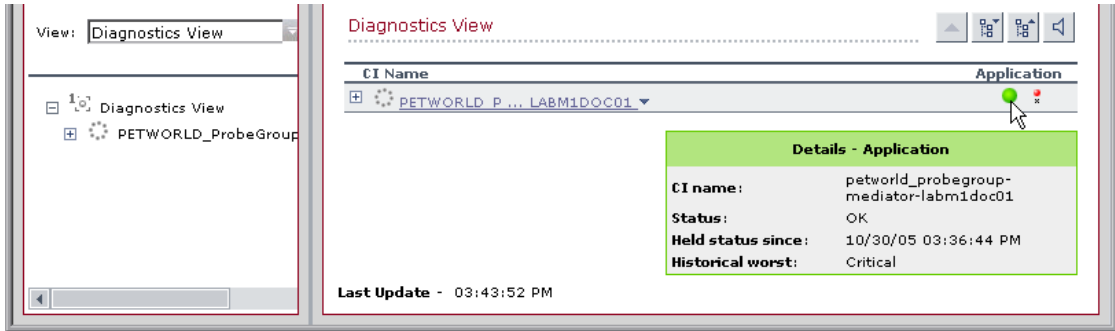
In the Diagnostics View in Dashboard, you can view the status of your applications that are being monitored by Mercury Diagnostics Probes. You can view the status of an individual Diagnostics probe or of a group of Diagnostics probes known as a probe group.

Note: Mercury Diagnostics Probes are assigned to probe groups as part of the probe installation procedure. For more information, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

You monitor the status of Diagnostics probe data using the *Application Key Performance Indicators (KPIs)*. For more information, see “Monitoring Performance Status Using KPIs” on page 312.

To view the status of Diagnostics Probes in Dashboard:

- 1** Select **Applications > Dashboard** to open Dashboard.
- 2** Click the **Console** tab to present the available views.
- 3** In the left pane, in the **View** box, select **Diagnostics View** to open the Diagnostics View. The Application KPI displays the status of the Diagnostics probe group.



You can expand each probe group to view the status of each individual probe.



Note: From the Diagnostics View in Dashboard, you can also drill down to the Diagnostics screens that display data about your applications that are being monitored by Diagnostics probes and probe groups. For more information, see “Drilling down to Diagnostics from Dashboard” on page 319.

Monitoring Transactions from Dashboard

From Business Process Monitor (BPM) related views in Dashboard, you can actively track the performance status of transactions that are monitored by Diagnostics. You use BPM data collectors to generate these transactions on your monitored application.

Introducing Business Process Monitor (BPM) Data Collectors

If a BPM data collector is deployed in your Mercury Business Availability Center environment, you can create new *Business Process Profiles* to collect performance data from the application server that you wish to monitor. The Business Process Profile includes *transaction monitors* (scripts) containing the transactions that you want your data collectors to run.

For more information about creating Business Process Profiles and adding transaction monitors in Mercury Business Availability Center, refer to the section on “Managing Business Process and Client Monitor Profiles” in *Monitor Administration* in the *Mercury Business Availability Center Documentation Library*.

Enabling Diagnostics Viewing for Transactions

Before viewing Diagnostics data for transactions included in a transaction monitor, you need to enable Diagnostics breakdown.

To enable Diagnostics breakdown for a transaction monitor:

- 1** In Mercury Business Availability Center, select **Admin > Monitors** to open the Monitor Administration page.
- 2** In the **Monitors** tab, select the relevant transaction monitor from the monitor tree, and click the **Properties** tab.
- 3** Under the **Transaction Breakdown Settings** section, select **Enable diagnostics breakdown**.

Viewing the Status of Transactions in BPM Related Views

After you have enabled Diagnostics breakdown, you can monitor the performance status of transactions From BPM related views in Dashboard. You monitor the status of these transactions using the *Application Key Performance Indicators (KPIs)*. For more information, see “Monitoring Performance Status Using KPIs” on page 312.

To view the status of transactions in Business Process Monitor (BPM)

Related Views:

- 1** Select **Applications > Dashboard** to open Dashboard.
- 2** Click the **Console** tab to present the available views.
- 3** In the left pane, in the **View** box, select a BPM Related View, such as **End User Monitors View**.
- 4** Click the Business Process Profile that you created to view the status of each transaction included in the profile. The Application KPI shows the status of the BPM transaction from the server perspective, according to thresholds which you set in the Mercury Diagnostics application.

The screenshot shows a web interface with a navigation bar at the top containing tabs: Top View, Console (selected), Filters, Geographical Map, Custom Map, Topology Map, and Reports. Below the navigation bar, there is a dropdown menu set to 'diagnostics_tests' and several control icons. A summary row shows three categories: Performance (green circle), Availability (green circle), and Application (green circle), each followed by a small green dot. Below this is a table with the following data:

CI Name	Performance	Availability	Application
+ buy_cat	● ●	● ●	● ●
+ buy_doq	● ●	● ●	● ●
+ buy_parrot	● ●	● ●	● ●
+ checkout	● ●	● ●	● ●

Note: From BPM related views in Dashboard, you can also drill down to the Diagnostics screens that display Diagnostics data about each specific transaction. For more information, see “Drilling down to Diagnostics from Dashboard,” below.

Drilling down to Diagnostics from Dashboard

The Diagnostics drilldowns differ, depending on which version of Mercury Business Availability Center you are using.

Mercury Business Availability Center 6.1

In Dashboard, you can drill down from specific Configuration Items (CIs) to the Diagnostics screens displaying data about that item.

To drill down to Diagnostics screens from Dashboard:

- 1 In the relevant view (Diagnostics View or any BPM related view), in the **Console** tab, choose the CI from which you want to drill down.

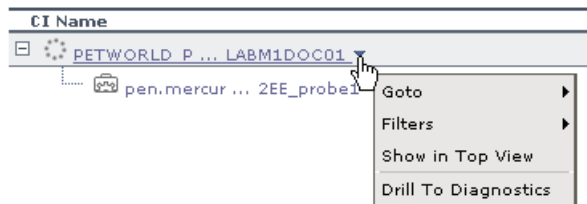


Note: In the BPM related views, you drill down to Diagnostics from the **BPM Transaction CI**. Click the **Expand All** button to expand the **Business Process Step** CIs and to view the individual **BPM transaction CIs**.

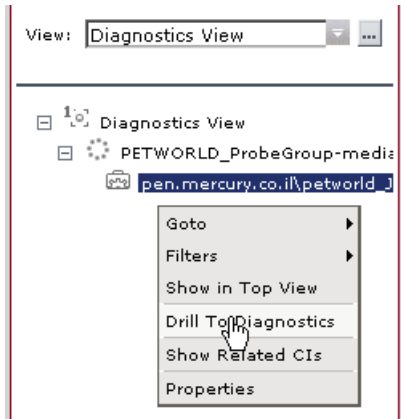
In Diagnostics View, you either drill down to Diagnostics from a **Diagnostics Probe Group CI** or from an individual **Diagnostics Probe CI**.

- 2 Click the down arrow to the right of the CI name to access the menu options and select **Drill to Diagnostics** to open the appropriate screen in the Mercury Diagnostics application.

Diagnostics View



Alternatively, you can right click the CI name in the left pane, and choose **Drill to Diagnostics** from the menu.



Mercury Business Availability Center 6.2

In Dashboard, you can drill down from selected CIs (Configuration Items) to the Diagnostics views displaying data about that item. You can drill down from the following Dashboard views:

- **BPM related views.** From selected CIs in BPM related views, you can drill down to the Diagnostics Transactions view, which displays performance metrics for the transactions that are being executed by your applications. You can also drill down directly into the view displaying the layers for the selected transactions.

For more information about the Diagnostics Transactions view, see “Using the Transactions View” on page 126.

- **RUM related views.** From selected CIs in RUM related views, you can drill down to the Diagnostics Server Requests view, which displays the performance metrics for the monitored server requests in your application.

For more information about the Diagnostics Server Requests view, see “Using the Server Requests View” on page 115.

- **The Diagnostics view.** From Diagnostics probe CIs in the Diagnostics view, you can drill down to the Probe Summary view or into the Load view for that particular probe. From Probe Group CIs, you can drill down to the Probe Group Summary view or into the Load view for that particular probe group.

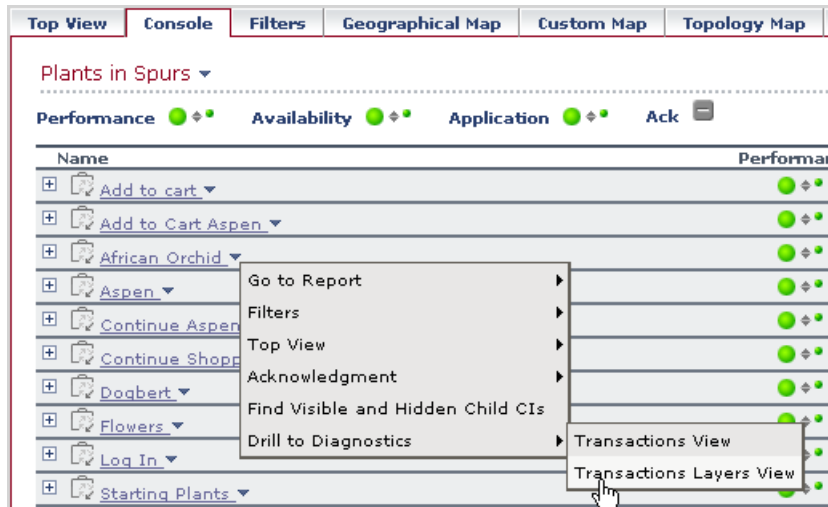
The load view displays the performance metrics for the Diagnostics layers where processing has taken place in your application. For more information about the Diagnostics Load view, see “Using the Load View” on page 103.

The following table displays the Diagnostics drilldown options for CIs in each of the relevant Dashboard views

Dashboard View	CI Type	Diagnostics drilldown options
BPM related view	Business Process Group	<ul style="list-style-type: none"> • Transactions View
	Business Process Step	<ul style="list-style-type: none"> • Transactions View • Transactions Layers View
	BPM Transaction from Location	<ul style="list-style-type: none"> • Transactions View • Transactions Layers View
RUM related view	End User Management Application Related Group	<ul style="list-style-type: none"> • Server Requests
	Business Process Step	<ul style="list-style-type: none"> • Server Requests
	RUM Page Monitor	<ul style="list-style-type: none"> • Server Requests
Diagnostics view	Diagnostics Probe Group	<ul style="list-style-type: none"> • Probe Group Summary • Load View
	Diagnostics Probe	<ul style="list-style-type: none"> • Probe Summary • Load View

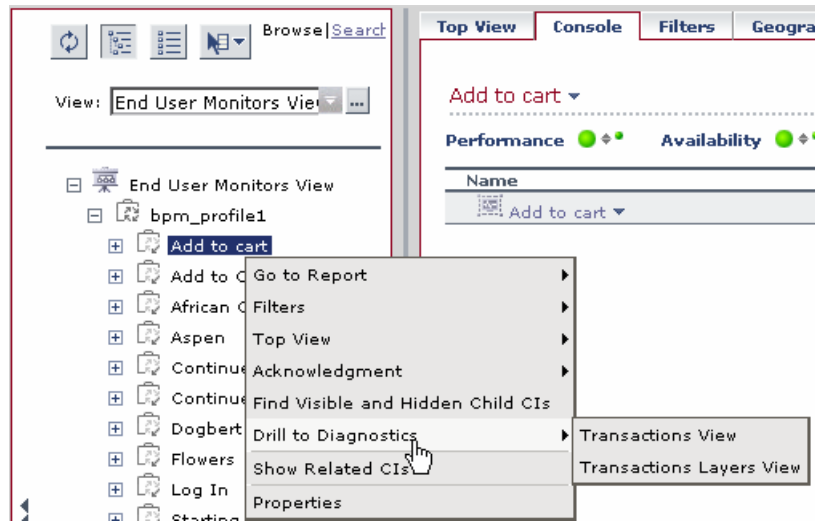
To drill down to Diagnostics screens from Dashboard:

- 1 In the relevant view in Dashboard, in the **Console** tab, choose the CI from which you want to drill down.
- 2 Click the down arrow to the right of the CI name to access the menu options and select **Drill to Diagnostics** to open the Diagnostics drilldown submenu.



From the submenu, select the Diagnostics view into which you want to drill down.

Alternatively, you can access the Diagnostics drilldown menu option (**Drill to Diagnostics**) by right clicking the CI name in the left pane.



Diagnostics Performance Reports in Mercury Business Availability Center

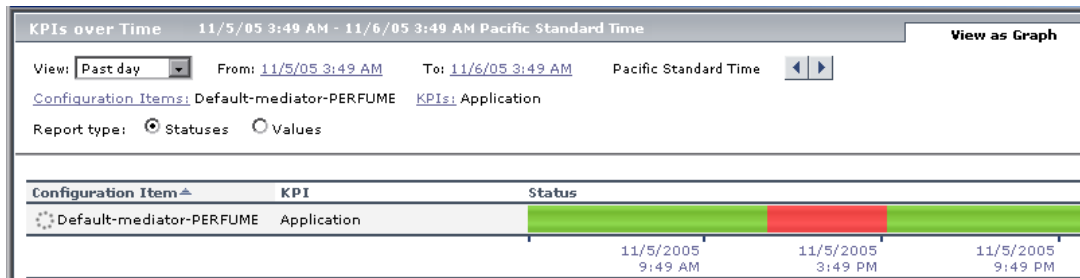
From Dashboard, you can generate the high level reports about the performance of your Diagnostics components and of specific transactions.

KPIs Over Time Report

KPIs Over Time reports show the status or value, over time, of selected CIs and KPIs that are accessible from the Dashboard application. You can generate KPIs Over Time reports to show the status during a certain period of time, of applications or specific transactions that are being monitored by Mercury Diagnostics.

Note: In Mercury Business Availability Center 6.2, you can drill down to Diagnostics data from KPI Over Time reports. For more information, see “Drilling down to Diagnostics Data from KPI Reports” on page 332

In the following example of a screen in Mercury Business Availability Center, a KPIs Over Time report has been generated for a Diagnostics probe Group:



In the above example, the report displays the status of the Diagnostics probe group as reflected by the Application KPI, over the past day. The Application KPI monitors the status of Mercury Diagnostics related data according to thresholds which you set in the Mercury Diagnostics application.

For more information about using Application KPIs to monitor performance of Diagnostics related variables, see “Monitoring Performance Status Using KPIs” on page 312.

For more information about KPIs Over Time reports, refer to the section on “KPIs Over Time Report” in *Using Dashboard* in the *Mercury Business Availability Center Documentation Library*.

Note: For instructions on how to generate Diagnostics related reports in Mercury Business Availability Center, see “Generating Diagnostics Related Reports” on page 326.

Configuration Item Status Alert Report

Mercury Business Availability Center allows you to create *Configuration Item Status* alerts that proactively inform you when predefined performance limits are breached. For Diagnostics related items, you can attach alerts to probes, probe groups or to specific transactions.

For more information about creating CI Status alerts refer to the section on “Configuring CI Status Alerts” in *Application Administration* in the *Mercury Business Availability Center Documentation Library*.

In Mercury Business Availability Center, you can view a report of all the alerts that occurred during a specified period of time by generating a Configuration Item Status Alerts report.

Below is an example of a Configuration Item Status Alerts report generated in Mercury Business Availability Center:

Status	Time	Alert Name	Configuration Item	KPI	Alert Action	Details
Warning	12/12/05 2:47 AM	worsen alert	MedRec Cluster	Application	Send E-mail to: Udi S	
Warning	12/11/05 11:12 PM	worsen alert	MedRec Cluster	Application	Send E-mail to: Udi S	
Critical	12/11/05 10:59 PM	worsen alert	MedRec Cluster	Application	Send E-mail to: Udi S	
Critical	12/11/05 2:14 PM	worsen alert	MedRec Cluster	Application	Send E-mail to: Udi S	
Critical	12/11/05 2:04 PM	worsen alert	MedRec Cluster	Application	Send E-mail to: Udi S	

For more information about Configuration Item Status Alert reports, refer to the section on “Configuration Item Status Alerts” in *Using Dashboard* in the *Mercury Business Availability Center Documentation Library*.

Note: For instructions on how to generate Diagnostics related reports in Mercury Business Availability Center, see “Generating Diagnostics Related Reports,” below.

Generating Diagnostics Related Reports

You can generate the reports either by drilling down from the relevant CI in the Dashboard or by configuring the reports in the Dashboard **Reports** tab.

To generate Diagnostics related reports from Dashboard:

- 1 In the relevant view (Diagnostics View or any BPM related view), choose the Diagnostics related CI from which you want to generate the report.
- 2 Click the down arrow to the right of the CI name to access the menu options.



- 3 From the **Goto** menu, select either **KPIs Over Time Report** or **Configuration Item Status Alerts** to generate the report.

Alternatively you can select **Application > Dashboard** and then select the appropriate report in the **Reports** tab. You then have to configure the report according to your specific requirements.

Drilling down to Diagnostics Data from Mercury Business Availability Center Reports

From certain Mercury Business Availability Center Reports, you can drill down to Diagnostics data.

Drilling down to Diagnostics Data from Transaction Breakdown Reports

In Mercury Business Availability Center you can generate *transaction breakdown reports*. Transaction breakdown reports enable you to assess whether poor transaction response times are caused by network or server problems, or by client delays, and to pinpoint exactly when the problems are occurring. Transaction breakdown reports, include the *Breakdown over Time* report and the *Breakdown Summary* report.

From certain measurements in the transaction breakdown reports, you can drill down to Diagnostics screens to further analyze the source of slow server time or download times. You drill down to Diagnostics screens from the following measurement categories (where available) in the transaction breakdown reports:

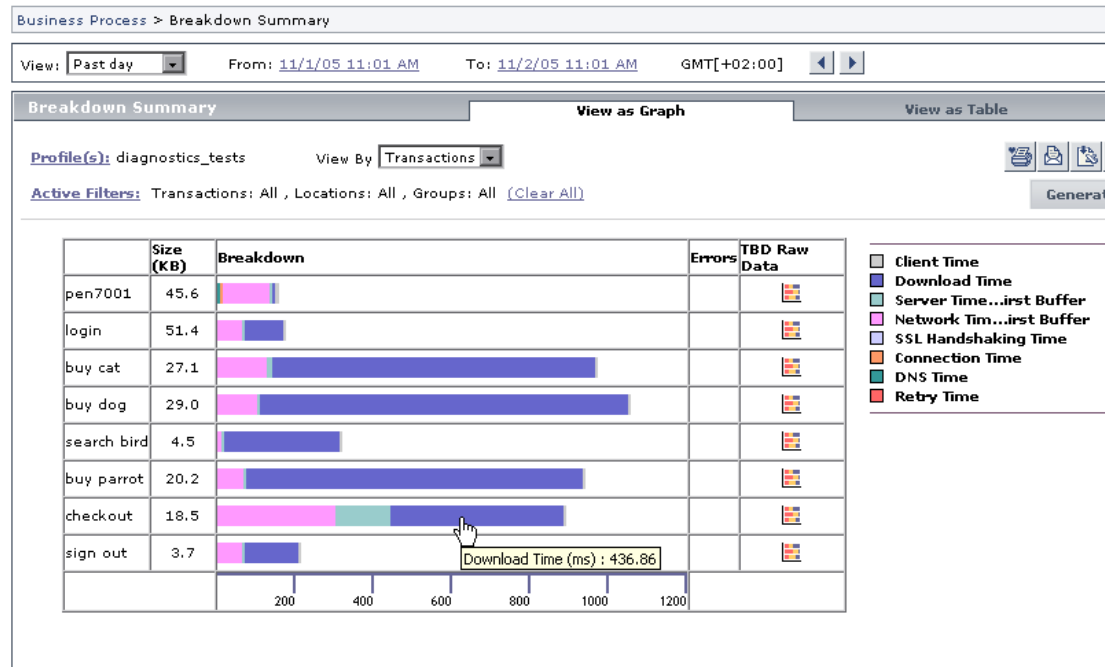
- **Server Time to First Buffer.** This measurement displays the average amount of time that passes from the receipt of ACK of the initial HTTP request (usually GET) until the first buffer is successfully received back from the Web server.
- **Download Time.** This measurement displays the time from the receipt of the first buffer until the last byte arrives.

For information about transaction breakdown reports and how to generate them, refer to the section on “Understanding the Transaction Breakdown Reports” in *Using End User Management* in the *Mercury Business Availability Center Documentation Library*.

To drill down to Diagnostics screens from a transaction breakdown report:

- 1 Generate a transaction breakdown report in Mercury Business Availability Center (either the Breakdown over Time report or the Breakdown Summary report).

In the following example of a screen in Mercury Business Availability Center, a Breakdown Summary report has been generated from a Business Process Profile containing Diagnostics data.



- 2 In the **View as Graph** tab, click on the segment in the graph that represents **Download Time**, or click on the segment that represents **Server Time to First Buffer**.

The appropriate screen opens in the Diagnostics Transactions view. For more details about interpreting data in the Diagnostics Transactions view see “Using the Transactions View” on page 126.

Alternatively, if you display the report as a table using the **View as Table** tab, you can drill down to Diagnostics screens by clicking the **Server Time to First Buffer** or **Download Time** data.

Breakdown Summary View as Graph **View as Table**

Profile(s): diagnostics_tests View By: Transactions

Active Filters: Transactions: All , Locations: All , Groups: All [\(Clear All\)](#) Generate

	Size (KB)	Retry Time (ms)	DNS Time (ms)	Connection Time (ms)	SSL Handshaking Time (ms)	Network Time to First Buffer (ms)	Server Time to First Buffer (ms)	Time to First Buffer (ms)	Download Time (ms)	Client Time (ms)	Avg. Response Time (ms)
pen7001	45.688	-	6.938	9.837	-	120.365	1.871	-	4.86	12.382	156.253
login	51.45	-	-	-	-	63.056	1.365	-	100.483	1.86	166.764
buy cat	27.131	-	-	-	-	125.955	15.68	-	817.713	5.438	964.787
buy dog	29.069	-	-	-	-	101.933	0.612	-	931.579	5.596	1039.719
search bird	4.527	-	-	-	-	13.685	2.404	-	291.798	2.551	310.438
buy parrot	20.22	-	-	-	-	68.135	0.612	-	851.77	2.843	923.36
checkout	18.579	-	-	-	-	299.253	147.489	-	437.522	5.258	884.522
sign out	3.71	-	-	-	-	63.287	3.275	-	136.584	2.146	205.292

The appropriate screen opens in the Diagnostics Transactions view. For more details about using the interpreting data in Diagnostics Transactions view see “Using the Transactions View” on page 126.

Drilling down to Diagnostics Data from Real User Monitor Reports

Note: This drill down option is available in Mercury Business Availability Center 6.1 and later.

The Real User Monitor is a Mercury Business Availability Center data collector that monitors real user traffic. You use Real User Monitor reports to view data collected by the Real User Monitor. These reports enable you to monitor the experience of real users that access your application.

From certain Real User Monitor reports that display server-related performance data, you can drill down to the Diagnostics Server Requests view to view a detailed breakdown of the worst performing server requests for a particular page. You can drill down to Diagnostics from the following Real User Monitor reports.

- ▶ Real User Monitor Page Summary report (from the Server Performance tab)
- ▶ Page Summary Over Time report
- ▶ Global Statistics report (from the Pages with Slowest Server Time table)

To drill down to Diagnostics Data from Real User Monitor Reports:



Click the View Diagnostics Data button in the row of the page for which you want to view the Diagnostics data. Diagnostics opens, displaying the Server Requests view.

In the following example of a Global Statistics report in Mercury Business Availability Center, the View Diagnostics Data button is displayed next to each page in the Pages with Slowest Server Time table.

Pages with Slowest Server Time (GMT +2) Asia/Jerusalem 1/18/06 1:48 PM - 1/25/06 1:48 PM			
Page URL	Server Time (sec.)	Download Time (sec.)	Hits
http://j2ee1/admin/login.do	0.42	0.57	43
http://j2ee1/physician/login.do	0.33	0.34	43
http://j2ee1/patient/login.do	0.29	0.34	43

Note: If the application server handling a particular page is not monitored by a Diagnostics probe, there will be no data displayed when you click the View Diagnostics Data button.

Real User Monitor Report Drill Down Limitations in Mercury Business Availability Center 6.1

The following limitations apply when you drill down to Diagnostics from Real User Monitor reports in Mercury Business Availability Center 6.1:

- ▶ If the URL of a page you have configured for monitoring in Monitor Administration has passed through a Web server that uses URL rewriting, the URL in Real User Monitor will differ from the corresponding URL in Mercury Diagnostics and a match will not be found when drilling down.
- ▶ If an application is installed on multiple servers working behind a load balancer, the URL of a page in Real User Monitor will have multiple corresponding URLs in Diagnostics. In such a case, when you drill down to the Server Requests view in Diagnostics, all the corresponding URLs will be selected, and one of them will be opened in the current view.
- ▶ When you drill down to the Server Requests view in Diagnostics from a Real User Monitor report, you will only view server requests that are included in the slowest 100 server requests in Diagnostics.
- ▶ Parameters aggregation is enabled by default in the Diagnostics probe points file. If you have turned off parameter aggregation in the probe points file, and the URL that you are drilling down from in the Real User Monitor report includes a parameter, an exact match will not be found when drilling down and you will have to manually locate the server request in the Server Requests view in Diagnostics.

For more information about Real User Monitor reports in Mercury Business Availability Center, refer to the section on “Real User Monitor Reports” in *Using End User Management* in the *Mercury Business Availability Center Documentation Library*.

For more information about the Mercury Diagnostics Server Requests view, see “Using the Server Requests View” on page 115.

Drilling down to Diagnostics Data from KPI Reports

Note: This drill down option is only available in Mercury Business Availability Center 6.2.

In Mercury Business Availability Center, you monitor the status of Diagnostics related data using the *Application Key Performance Indicators (KPIs)*. For more information, see “Monitoring Performance Status Using KPIs” on page 312.

You can generate KPIs Over Time reports to show the status during a certain period of time, of applications or specific transactions that are being monitored by Mercury Diagnostics.

In KPIs Over Time reports that include Application KPIs, you can drill down to Diagnostics from selected CIs. The following table displays the Diagnostics drilldown options for CIs in KPIs Over Time reports:

CI Type	Diagnostics drilldown options
Diagnostics Probe Group	<ul style="list-style-type: none"> • Summary View (Probe Group Summary) • Layers View (Load)
Diagnostics Probe	<ul style="list-style-type: none"> • Summary View (Probe Summary) • Layers View (Load)
Business Process Step	<ul style="list-style-type: none"> • Transactions View • Layers View

For more information about the Diagnostics views, see “Using Mercury Diagnostics” on page 15.

To drill down to Diagnostics from KPIs Over Time Reports:

- 1 Generate a KPIs Over Time report for one of the following CIs:
 - ▶ Diagnostics Probe CI
 - ▶ Diagnostics Probe Group CI

► BPM related Business Process Step CI

For more information about generating KPIs Over Time reports from Dashboard, see “Generating Diagnostics Related Reports” on page 326.

- 2 Click the down arrow to the right of the relevant CI name to access the **Drill to Diagnostics** menu options and select the Diagnostics view into which you want to drill down.

KPIs Over Time 5/30/06 8:11 AM - 5/31/06 8:11 AM America/Los_Angeles

View: Past day From: [5/30/06 8:11 AM](#) To: [5/31/06 8:11 AM](#) America/Los_Angeles

[Configuration Items: spurs.mercury.co.il\SPURS_WAS62](#) [KPIs: All](#)

Report type: Statuses Values

Configuration Item	KPI	Status
spurs.mercury.co.il\SPURS_WAS62	Application	

Drill to Diagnostics
Summary View
Layers View

OK Warning Minor Major Critical Downtime No Data

17

Viewing Diagnostics Data in LoadRunner

This chapter provides instructions for configuring Diagnostics parameters in LoadRunner before you run a load test scenario and explains how to view Diagnostics data from within LoadRunner, during and after the load test.

This chapter describes:	On page:
About Viewing Mercury Diagnostics Data in LoadRunner 8.1	335
Configuring LoadRunner Scenarios to use Mercury Diagnostics	336
Drilling Down to Diagnostics Data During a Load Test Scenario	340
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	343

About Viewing Mercury Diagnostics Data in LoadRunner 8.1

Setting Up LoadRunner to Use Mercury Diagnostics

Before you can use Mercury Diagnostics with LoadRunner, you need to ensure that you have specified the Diagnostics Server details in LoadRunner, according to the guidelines set out in the *Mercury Diagnostics Installation and Configuration Guide*.

Before you can view Mercury Diagnostics data in a particular load test scenario, you need to configure the Diagnostics parameters for that scenario, as described in “Configuring LoadRunner Scenarios to use Mercury Diagnostics” on page 336.

Viewing Diagnostics Data in LoadRunner

During a load test scenario, you can drill down to Mercury Diagnostics data for the whole scenario or for a particular transaction. For more information, see “Drilling Down to Diagnostics Data During a Load Test Scenario” on page 340.

After you have run your scenario, you can use Mercury LoadRunner Analysis to analyze offline Diagnostics data generated during the scenario. For more information, see “Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis” on page 343.

Configuring LoadRunner Scenarios to use Mercury Diagnostics

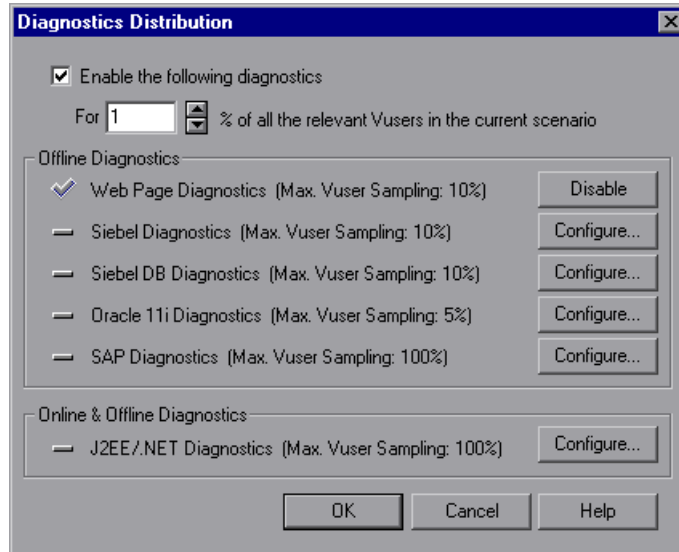
Each time you want to capture Mercury Diagnostics metrics in a load test scenario, you need to configure the Diagnostics parameters for the scenario and select the probes that will be included in the scenario. You configure your scenario for Diagnostics from the LoadRunner Controller.

Note: If you have saved a scenario with the Diagnostics settings already configured, you do not need to reconfigure the Diagnostics parameters each time you run that scenario.

To configure the Mercury Diagnostics parameters for a load test scenario:

- 1** Before configuring your scenario for Diagnostics, ensure that the application server that you are monitoring has been started.
- 2** Select **Start > Programs > Mercury LoadRunner > Applications > Controller** to launch the Mercury LoadRunner Controller.
- 3** Open the relevant load test scenario (**File > Open**) or create a new scenario (**File > New**).
- 4** From the LoadRunner Controller menu bar, select **Diagnostics > Configuration**.

The Diagnostics Distribution dialog box opens.



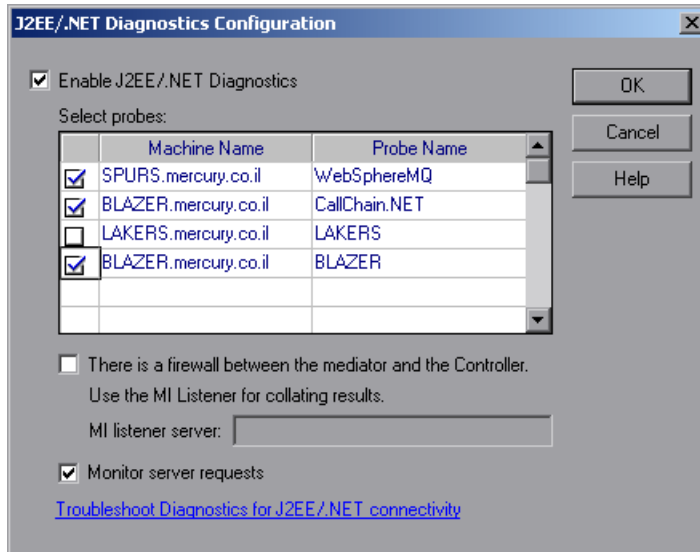
- 5 Select **Enable the following diagnostics**.
- 6 Set the percentage of Vusers to participate in the Mercury Diagnostics (**J2EE/.NET Diagnostics**) monitoring.

The maximum percentage of Vusers for which Mercury Diagnostics (**J2EE/.NET Diagnostics**) data can be collected is 100%, unless you have enabled other types of diagnostics. In this case, the percentage of Vuser participation in Mercury Diagnostics (**J2EE/.NET Diagnostics**) cannot exceed the maximum of any of the other types of diagnostics that you enabled.

For example, if you enabled **Web Page Diagnostics**, which has a maximum user participation of 10%, the percentage of Vuser participation for Mercury Diagnostics (**J2EE/.NET Diagnostics**) cannot exceed 10%.

The minimum amount of Vusers for which Mercury Diagnostics (**J2EE/.NET Diagnostics**) data can be collected is 1% or 1 virtual user per script.

- 7 In the **Online & Offline Diagnostics** section of the Diagnostics Distribution dialog box, next to **J2EE/.NET Diagnostics**, click **Configure**. The J2EE/.NET Diagnostics Configuration dialog box opens.



Note: This dialog box is read-only while a scenario is running.

- 8 Select **Enable J2EE/.NET Diagnostics**.
- 9 In the **Select probes** list, select the probes to be included in your load test scenario.
 - Select the check box adjacent to each probe that you want to monitor.
 - To disable a probe for the duration of a scenario, clear the check box.

Note: You must enable at least one probe in order to save the Diagnostics configuration.

- 10** If the Diagnostics Server (or a Diagnostics Server in Mediator mode in a distributed environment) is located behind a firewall, select **There is a firewall between the mediator and the Controller**, and enter the name of the MI listener server in the **MI listener server** box.
- 11** To capture a percentage of server requests which occur outside the context of any Vuser transaction select **Monitor server requests**.

The server requests will be captured at the same percentage as was selected for the percentage of Vusers in the Diagnostics Distribution dialog box.

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** To investigate any issues that you have with the connections between the Diagnostics components, click the **Troubleshoot Diagnostics for J2EE/.NET connectivity** link. This will open the System Health Monitor in a new browser window.

For details on how to use the System Health Monitor, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

- 13** Click **OK** to confirm your selections and close the J2EE/.NET Diagnostics Configuration dialog box.
- 14** Click **OK** on the Diagnostics Distribution dialog box to save your settings and complete the configuration.

Drilling Down to Diagnostics Data During a Load Test Scenario

During a load test scenario, you can view Mercury Diagnostics data for the whole scenario or you can drill down to Mercury Diagnostics data from a particular transaction.

To view a summary screen of the scenario in Mercury Diagnostics:

Click the **Diagnostics for J2EE/.NET** tab at the bottom of the LoadRunner window. Mercury Diagnostics opens, displaying the **LR / PC Summary** dashboard view.

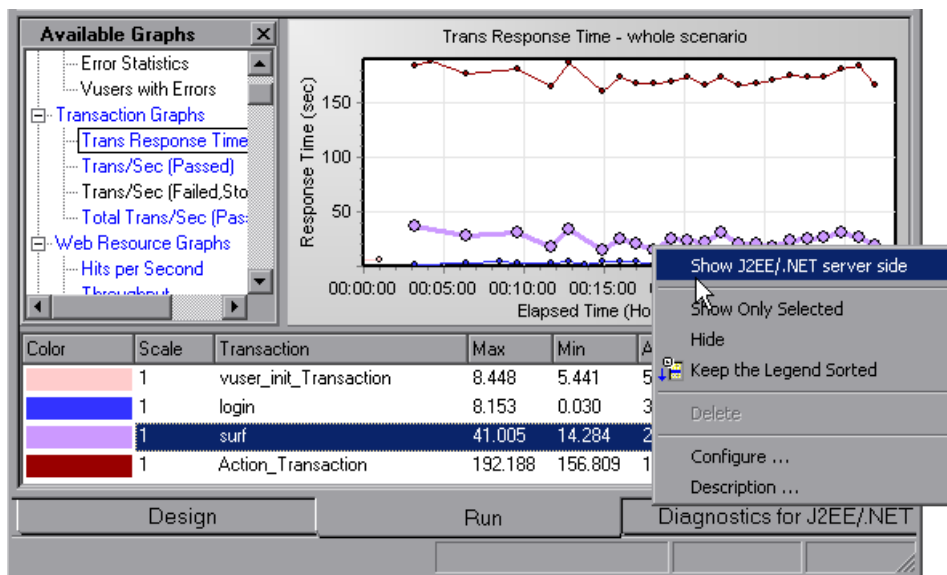


The **LR / PC Summary** dashboard view displays monitoring versions of the transactions, server requests, load, and probe views for the current run. For more information about the Mercury Diagnostics views, see Part II, “Using Mercury Diagnostics.”

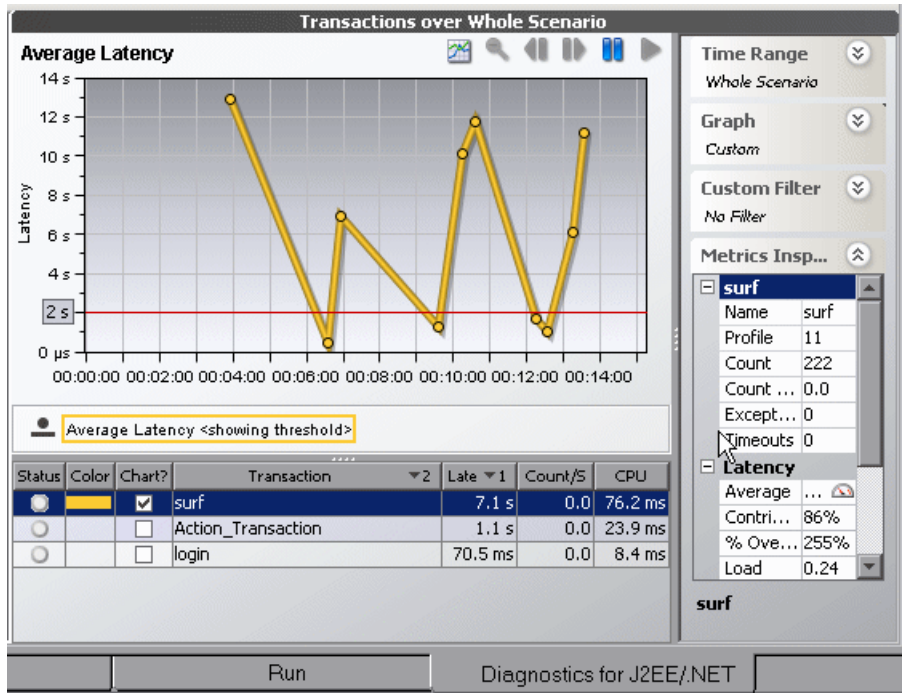
Note: During the load test scenario, you can navigate to other views in Mercury Diagnostics. If you move to another tab and then return to the **Diagnostics for J2EE/.NET** tab, the last screen that you were viewing is displayed.

To drill down to Mercury Diagnostics data from a particular transaction:

- 1 In the **Available Graphs** list, click one of the Transaction graphs, for example **Transaction Response Time** to open the graph.
- 2 Right-click the line on the graph representing the transaction from which you want to drill down to Mercury Diagnostics data and select **Show J2EE/.NET server side**. Alternatively you can right-click the relevant transaction in the graph legend, and select **Show J2EE/.NET server side**.



Mercury Diagnostics opens, displaying the Transactions view, which contains performance metrics and drill down options for the relevant transaction.



For more information about interpreting data in the Diagnostics Transactions view see “Using the Transactions View” on page 126.

Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis

Mercury LoadRunner Analysis provides offline graphs and reports with in-depth performance analysis information. Using these graphs and reports, you can pinpoint and identify the bottlenecks in your application and determine what changes need to be made to your system to improve its performance.

Analysis provides specific graphs that display Mercury Diagnostics performance metrics that were captured during the load test scenario.

After you have completed your load test scenario run, you can use Analysis to analyze offline Diagnostics data that was generated during the run.

There are two groups of Mercury Diagnostics graphs presented in Analysis:

- ▶ **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. These include requests generated by virtual user transactions and by real users.

For detailed information about using the Mercury Diagnostics graphs in Analysis, refer to the *Mercury LoadRunner Analysis User's Guide*.

18

Viewing Diagnostics Data in Performance Center

This chapter provides instructions for configuring Diagnostics parameters in Performance Center before you run a load test and explains how to view Diagnostics data from within Performance Center, during and after a load test.

This chapter describes:	On page:
About Viewing Mercury Diagnostics Data in Performance Center 8.1	346
Configuring Performance Center Load Tests to Use Mercury Diagnostics	347
Drilling Down to Diagnostics Data During a Load Test	351
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	355

About Viewing Mercury Diagnostics Data in Performance Center 8.1

Setting Up Performance Center to Use Mercury Diagnostics

Before you can use Mercury Diagnostics with Performance Center, you need to ensure that you have specified the Diagnostics Server details in Performance Center, according to the guidelines set out in the *Mercury Diagnostics Installation and Configuration Guide*.

Before you can view Mercury Diagnostics data in a particular load test, you need to configure the Diagnostics parameters for that load test, as described in “Configuring Performance Center Load Tests to Use Mercury Diagnostics” on page 347.

Viewing Diagnostics Data in Performance Center

During a load test, you can drill down to Mercury Diagnostics data for the whole load test or for a particular transaction. For more information, see “Drilling Down to Diagnostics Data During a Load Test” on page 351.

After you have run your load test, you can use Mercury LoadRunner Analysis to analyze offline Diagnostics data generated during the load test. For more information, see “Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis” on page 355.

Configuring Performance Center Load Tests to Use Mercury Diagnostics

Each time you want to capture Diagnostics metrics in a load test, you need to configure the Diagnostics parameters for the load test select the probes that will be included in the load test. You provide this information on the Load Test Configuration page of the Performance Center User Site.

To configure the Diagnostics 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 identified as 'Admin' and the project as 'Default'. The main heading is 'Load Tests'. A 'New Load Test' button is located above a table of existing tests. The table has two columns: 'Name (# of Runs)' and 'Last Modified Date'. The table contains six rows of test entries, each with a plus sign icon to its left.

Name (# of Runs)	Last Modified Date
asd (0)	11-Jul-2005
666 (0)	11-Jul-2005
sched1 (0)	4-Jul-2005
44 (0)	28-Jun-2005
+ dashboard (2)	28-Jun-2005
+ empty transaction test (2)	4-Jul-2005

- 3 Click an existing test that you want to configure in the **Name (# of Runs)** column or click **New Load Test** to create a new test.

4 Click the **Diagnostics** tab and select **Enable diagnostics**.

The screenshot shows the Mercury Diagnostics configuration interface. At the top, there are tabs for 'General', 'Design Groups', 'Scheduler', 'Monitors', and 'Diagnostics'. The 'Diagnostics' tab is active. Below the tabs, there is a text box: '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.' Below this, there is a checked checkbox labeled 'Enable diagnostics'. Underneath, it says 'Perform diagnostics breakdown on % of all relevant Vusers participating in the current Load Test'. There are two main sections: 'Offline Diagnostics' and 'Offline & Online Diagnostics'. The 'Offline Diagnostics' section contains five items: 'Web Page Breakdown (Max. Vuser Sampling: 10%)' with a 'Disable' button, 'Siebel Diagnostics (Max. Vuser Sampling: 10%)' with a 'Configure...' button, 'Siebel DB Diagnostics (Max. Vuser Sampling: 10%)' with a 'Configure...' button, 'Oracle 11i Diagnostics (Max. Vuser Sampling: 5%)' with a 'Configure...' button, and 'SAP Diagnostics (Max. Vuser Sampling: 100%)' with a 'Configure...' button. The 'Offline & Online Diagnostics' section contains one item: 'J2EE/.NET Diagnostics (Max. Vuser Sampling: 100%)' with a 'Configure...' button.

Note: You can disable diagnostics between load test runs without losing your configuration settings by clearing the **Enable diagnostics** check box (provided that you saved your settings).

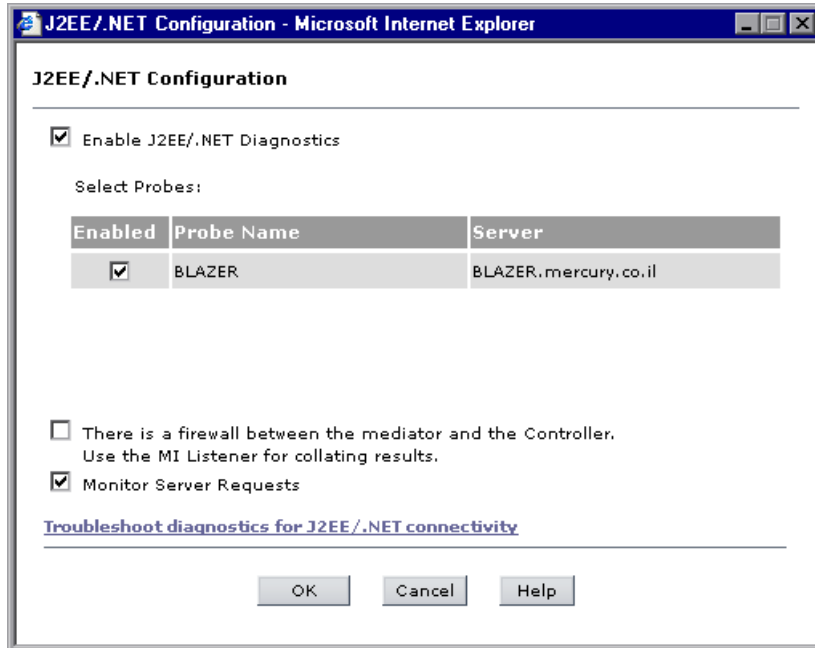
5 Set the percentage of Vusers to participate in the Mercury Diagnostics (**J2EE/.NET Diagnostics**) monitoring.

The maximum percentage of Vusers for which Mercury Diagnostics (**J2EE/.NET Diagnostics**) data can be collected is 100%, unless you have enabled other types of diagnostics. In this case, the percentage of Vuser participation in Mercury Diagnostics (**J2EE/.NET Diagnostics**) cannot exceed the maximum of any of the other types of diagnostics that you enabled.

For example, if you enabled **Web Page Diagnostics**, which has a maximum user participation of 10%, the percentage of Vuser participation for Mercury Diagnostics (**J2EE/.NET Diagnostics**) cannot exceed 10%.

The minimum amount of Vusers for which Mercury Diagnostics (**J2EE/.NET Diagnostics**) data can be collected is 1% or 1 virtual user per script.

- 6 In the **Offline and Online Diagnostics** section, click **Configure** to open the J2EE/.NET Configuration dialog box.



Note: This dialog box is read-only while a load test is running.

- 7 Select **Enable J2EE/.NET Diagnostics**.

- 8 In the **Select Probes** list, select the probes to be included in your load test.
 - Select the check box adjacent to each probe that you want to monitor.
 - 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 save the Mercury Diagnostics configuration.

- 9 If the Diagnostics Server (or a Diagnostics Server in Mediator mode in a distributed environment) is located behind a firewall, select **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 the *Mercury Diagnostics Installation and Configuration Guide*.

- 10 To capture a percentage of server requests which occur outside the context of any Vuser 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 Distribution dialog box.

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

To investigate any issues that you have with the connections between the Diagnostics components, click the **Troubleshoot diagnostics for J2EE/.NET connectivity** link. For details on how to use the System Health Monitor, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

- 11 Click **OK** to confirm your selections and to close the J2EE/.NET Diagnostics Configuration dialog box.
- 12 Click **Save** in the Diagnostics tab to save and validate your settings and complete the configuration.

Drilling Down to Diagnostics Data During a Load Test

During a load test, you can view Mercury Diagnostics data for the whole load test or you can drill down to Mercury Diagnostics data from a particular transaction.

To view a summary screen of the load test in Mercury Diagnostics:

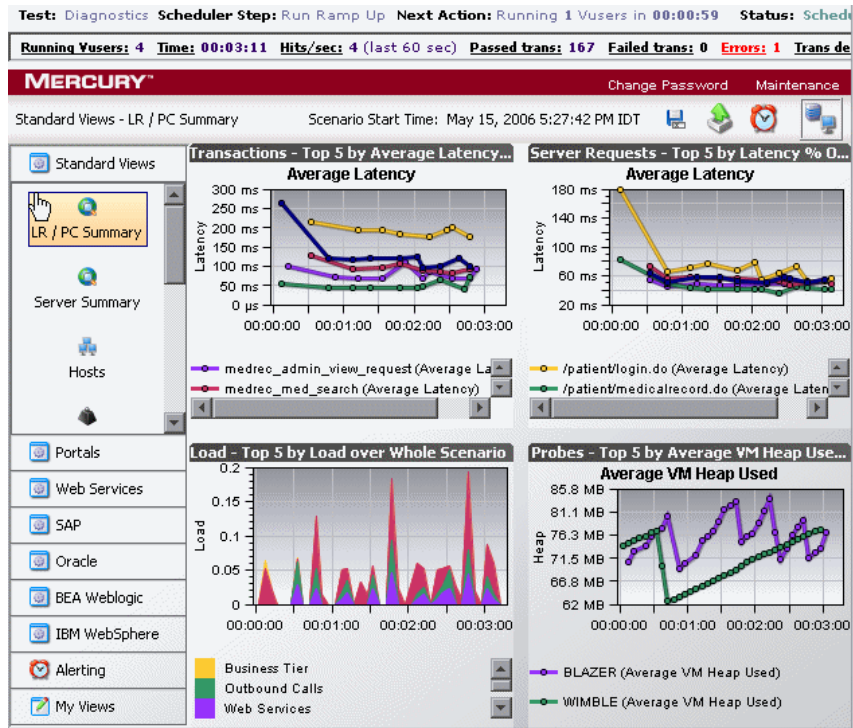
Click **View Diagnostics** on the right side of the Performance Center window.

Running Users: 5 Time: 00:13:24 Hits/sec: 7 (last 60 sec) Passed trans: 1588 Failed trans: 0

Group:	Down	Init	Ready	Run	Rendez	Exiting	Passed
Total:	0	0	0	5(0)	0	0	0
med_rec_blaze...				5(0)			

Buttons on the right: Run Users..., Stop Test, Users..., Design..., Output Window..., View Diagnostics

Mercury Diagnostics opens, displaying the **LR / PC Summary** dashboard view.

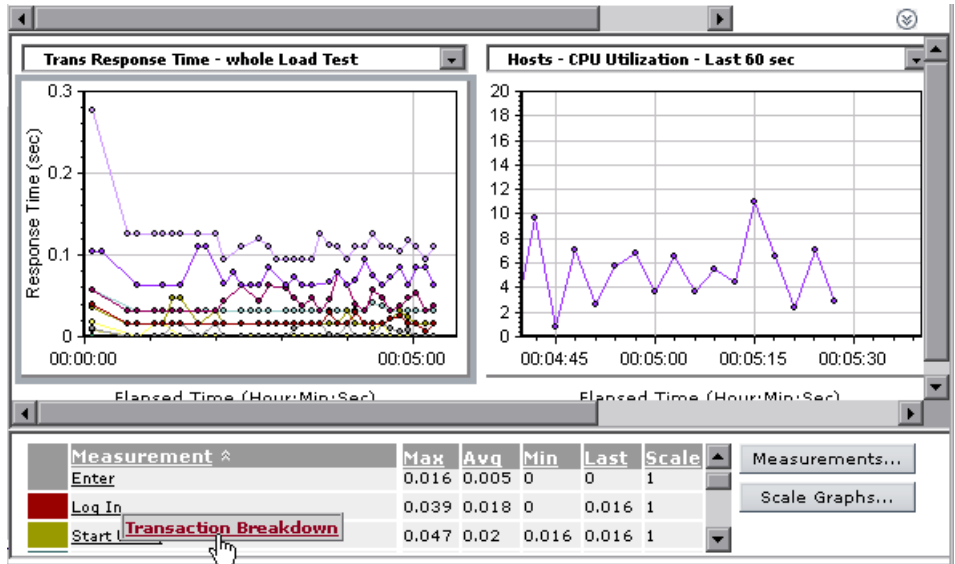


The **LR / PC Summary** dashboard view displays monitoring versions of the transactions, server requests, load, and probe views for the current run. For more information about the Mercury Diagnostics views, see Part II, “Using Mercury Diagnostics.”

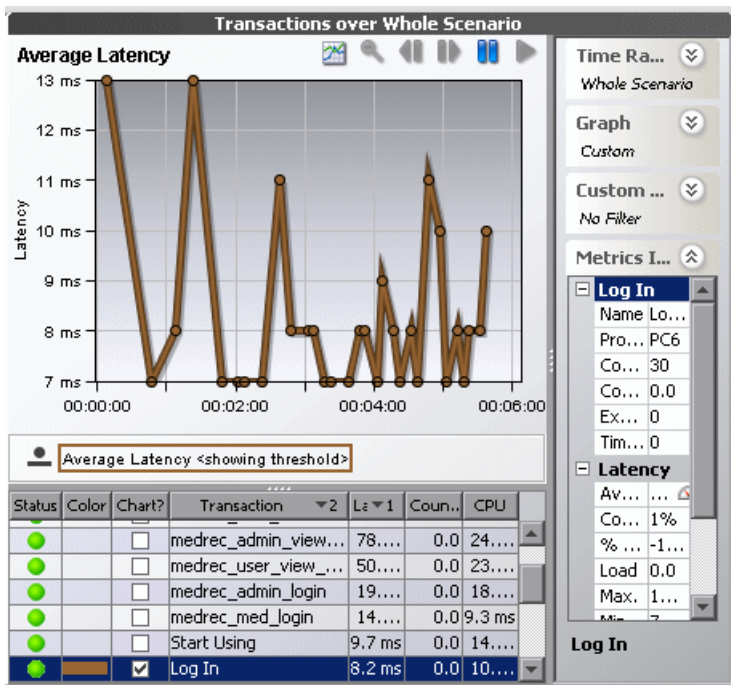
To drill down to Mercury Diagnostics data from a particular transaction:

- 1 Click the drop-down graph list located above any of the graphs and select one of the Transaction graphs, for example, **Transaction Response Time - whole Load Test**.
- 2 Click on the graph to display measurements for the graph in the graph legend below the graph pane.

- In the graph legend, click the transaction from which you want to drill down to Mercury Diagnostics. The **Transaction Breakdown** link is displayed. Click the **Transaction Breakdown** link.



Mercury Diagnostics opens, displaying the Transactions view, which contains performance metrics and drill down options for the relevant transaction.



For more details about interpreting data in the Diagnostics Transactions view see “Using the Transactions View” on page 126.

Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis

Mercury LoadRunner Analysis provides offline graphs and reports with in-depth performance analysis information. Using these graphs and reports, you can pinpoint and identify the bottlenecks in your application and determine what changes need to be made to your system to improve its performance.

Analysis provides specific graphs that display Mercury Diagnostics performance metrics that were captured during the load test.

After you have completed your load test run, you can use Analysis to analyze offline Diagnostics data that was generated during the run.

There are two groups of Mercury Diagnostics graphs presented in Analysis:

- ▶ **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. These include requests generated by virtual user transactions and by real users.

For detailed information about using the Mercury Diagnostics graphs in Analysis, refer to the *Mercury LoadRunner Analysis User's Guide*.

Index

A

- aggregate trees 212
 - resolving problems with 212
- alerts, status view 157
- application memory, monitoring 270

B

- BEA WebLogic views
 - EJB Pooled Resource Contention 184
 - JDBC Connection Status 184
 - JDBC Resource Contention 185
 - Server Threads 185
 - Summary view 184
 - using 184
- Business Availability Center, *See* Mercury Business Availibilty Center

C

- call profile view 130, 131
 - accessing 137
 - interpreting 137
- call stacks, monitoring 269
- CICS views
 - J2C Connection Pool 188
 - J2C Connections and Faults 188
 - J2C Load 188
 - J2C Usage Time 188
 - J2C Wait Time 188
 - Summary 188
 - using 187
- cross-VM trees 209
- custom views
 - modifying 201
 - sharing 201

D

- dashboard views 29
- data processing, Diagnostics 11
- detail table
 - columns 73, 92, 93
 - customizing 74
 - drilling down into host 103
 - drilling down into layer 108, 145
 - drilling down into portal component 151
 - drilling down into probe 114
 - drilling down into server request 120
 - drilling down into transaction 130
 - graphs 75
 - searching for element in 76
 - selecting row 50
 - viewing element details 77
 - working with 72
- detail views 28, 37
 - common features 38
 - common features, view filters 41
 - graph 57
- Diagnostics data 11
- Diagnostics Profiler for .NET, *See* NET Diagnostics Profiler
- Diagnostics Profiler for J2EE, *See* J2EE Diagnostics Profiler
- Diagnostics views, *See* Mercury Diagnostics views
- documentation updates xiii

E

- elements
 - displaying in graph 61
 - viewing data about 77
 - viewing in detail table 76

Index

exporting Diagnostics views to HTML reports
31

F

filtering views
by graphed metrics 45
by probe group 46, 47
by time range 42

G

graph
detail table 75
detail view 57
displaying elements and metrics in 61
magnification zooming 66
metric data 62
metrics view filter 45
multiple metrics 62
resolution zooming 67
working with 56, 97
zooming in on metrics 67
zooming out of metrics 70

H

host
drilling down into in detail table 103
hosts view 98
accessing 100
customizing 101
interpreting 101
HTML reports, exporting Diagnostics views
to 31

I

IBM WebSphere views
JDBC Connection Status 186
JDBC Resource Contention 186
MQ Connection Stats 186
MQ Message Driven Bean Stats 186
MQ Queue Stats 187
Server Threads 187
Servlet Session Management 187

Summary view 185
using 185
instance trees 204
aggregate trees 212
cross-VM trees 209
drilling down into for a server request
121
insufficient 211
solving problems with 205, 212

J

J2EE Diagnostics Profiler
accessing 222
All Methods tab 236
All SQL tab 240
baseline, setting new 260
call profile 249
call stacks 223
Call Tree table 251
Collections tab 257
Exceptions tab 242
garbage collection 226
Heap Breakdown sampling 264
Heap Breakdown tab 262
Hotspots tab 232
J2EE processing 223
memory, analyzing using the Heap
Breakdown tab 262
memory, monitoring application 224
method latency 223
Metrics Inspector 253
Metrics tab 234
metrics, refreshing 225
metrics, resetting 225
Server Requests tab 244
SQL statement details, viewing 241
Summary tab 228
Web Services tab 254

L

layers
drilling down into in detail table 108,
145

- layers view 130, 141
 - accessing 142
 - customizing 142
 - interpreting 142
- life-cycle methods for portlets, analyzing 140
- load view 103
 - accessing 105
 - customizing 106
 - interpreting 106
- LoadRunner
 - analyzing offline diagnostics data 343
 - Diagnostics data, drilling down to
 - during a load test scenario 340
 - Diagnostics data, viewing 335
 - loadtest, configuring to use Mercury
 - Diagnostics 336
 - scenario, configuring to use Mercury
 - Diagnostics 336

M

- magnification zooming 66
- Mercury Business Availability Center
 - Configuration Item Status Alert
 - Report 325
 - Dashboard 315, 317
 - Diagnostics data, viewing 310
 - Diagnostics reports, generating 326
 - Diagnostics screens, accessing 311
 - KPIs Over Time Report 323
 - Transaction Breakdown Reports 327
- Mercury Customer Support Web site xiii
- Mercury Diagnostics Profiler for .NET, *See*
 - NET Diagnostics Profiler
- Mercury Diagnostics Profiler for J2EE, *See*
 - J2EE Diagnostics Profiler
- Mercury Diagnostics views 18
 - accessing 19
 - creating new 194
 - customizing 190
 - dashboard views 29
 - deleting 200
 - detail views 28, 37
 - exporting to HTML reports 31
 - modifying custom views 201
 - navigation and display controls 23

- renaming 199
- saving 192
- sharing custom views 201
- sorting rows 34
- standard detail view 97
- status view 30, 154
- table columns 32
- view group 191
- Mercury Home Page xiii
- method latency, monitoring 269
- Metrics Inspector 48
 - reviewing metric details/settings 51
 - thresholds 53
 - viewing metrics 50
- metrics, .Net Profiler 270
- metrics, Diagnostics
 - data in graph 62
 - displaying in graph 52, 61
 - multiple 62
 - setting thresholds 53
 - threshold alerts 54
 - viewing data in graph 62
 - zooming in on 67
 - zooming out of 70
- metrics, J2EE Profiler 225

N

- navigation/display controls in Mercury
 - Diagnostics views 23
- NET Diagnostics Profiler
 - .NET processing 269
 - accessing 268
 - baseline, setting new 300
 - call stacks 269
 - Call Tree tab 286
 - Collections tab 294
 - Exceptions Tab 284
 - Heap tab 301
 - memory, analyzing using the
 - Collections tab 293
 - memory, analyzing using the Heap tab 301
 - method latency 269
 - Methods tab 281
 - monitoring application memory 270
 - refreshing metrics 270
 - resetting metrics 271
 - Server Requests tab 274
 - snapshots, taking 272
 - SQL method calls 280
 - SQL tab 279

O

- Oracle views
 - Probes view 181
 - Summary view 180
 - using 179
 - Wait Time view 183

P

- Performance Center
 - Diagnostics data, analyzing offline 355
 - Diagnostics data, viewing 346
 - Diagnostics, configuring load tests to use 347
 - Diagnostics, drilling down to 351
- portal component view
 - drilling down into in detail table 151
- portal components view 109

- accessing 147
- customizing 148
- interpreting 148

- Portal views
 - about 163
 - Business Process Summary view 164
 - LR PC Summary view 165
 - Portal Components view 166
 - Server Summary view 165

- Probe
 - details, status view 158
 - drilling down into in detail table 114
 - status view 160

- Probe Group
 - status view 159
 - view filter 46, 47

- Probes view 109
 - accessing 110
 - customizing 111
 - interpreting 111

- Profiler for .NET, *See* NET Diagnostics Profiler
- Profiler for J2EE, *See* J2EE Diagnostics Profiler

R

- Remote Function Call (SAP), Diagnostics
 - display 139
- Remote Method Invocation, *See* RMI (Remote Method Invocation)
- resolution zooming 67
- RFC (SAP), *See* Remote Function Call (SAP)
- RMI (Remote Method Invocation), analyzing 140

S

- SAP Remote Function Call, *See* Remote Function Call (SAP)
- SAP views
 - about 173
 - NetWeaver Requests 176
 - NetWeaver Summary 174
 - NetWeaver Threads 177
 - R3 Probes 178
 - R3 Server Requests 179
 - R3 Summary 175

- server request
 - drilling down into in detail table 120
 - drilling down into instance trees 121
- server requests view 115
 - accessing 116
 - customizing 117
 - interpreting 118
- standard detail views 97
 - call profile view 130
 - hosts view 98
 - layers view 140
 - load view 103
 - portal components view 146
 - probes view 109
 - server requests view 115
 - transactions view 126
- status view 30, 154
 - accessing 155
 - customizing 156
 - customizing tables 156
 - displaying probe details 158
 - drilling down 159
 - host, drilling down 161
 - interpreting 157
 - investigating alert conditions 157
 - Probe Group, drilling down 159
 - probe, drilling down 160

T

- tables in Diagnostics views
 - columns, customizing 32
 - customizing in status view 156
 - header controls 32
 - row, selecting 50
 - sorting rows 34
- threshold alerts 54
- time measurements, Diagnostics 18
- time range, filtering by 42
- transaction
 - drilling down into in detail table 130
- transactions view 126
 - accessing 127
 - customizing 128
 - interpreting 128

U

- updates, documentation xiii

V

- view filters 41
 - graphed metrics 45
 - probe group 46, 47
 - time range 42
- view group, creating 191
- views, *See* Mercury Diagnostics views

W

- Web Services views
 - about 167
 - Inbound Calls 169
 - Instance Trees 171
 - Outbound Calls 170

Z

- zooming in on a graph 65
 - magnification zooming 66
 - resolution zooming 67

