

OPTIMIZE

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

VERSION 6.5

User's Guide

MERCURY™

BUSINESS TECHNOLOGY OPTIMIZATION

Mercury Diagnostics

User's Guide

Version 6.5

Document Release Date: December 21, 2006

MERCURY™

Mercury Diagnostics User's Guide Version 6.5

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

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

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

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

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

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

© 2004 - 2006 Mercury Interactive Corporation, All rights reserved

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

Table of Contents

Welcome to This Guide	xi
How This Guide Is Organized	xii
Mercury Diagnostics Online Documentation.....	xiii
Additional Online Resources.....	xv
Documentation Updates	xv
Typographical Conventions.....	xvi

PART I: INTRODUCTION

Chapter 1: Mercury Diagnostics Product Overview	3
Introducing Mercury Diagnostics	4
Diagnostics Solutions	5
Diagnostics Components and Data Flow.....	9
Viewing Diagnostics Data	11
Accessing the Mercury Diagnostics Views	16
Interpreting Diagnostics Data	19

PART II: WORKING WITH DIAGNOSTICS VIEWS

Chapter 2: Common Features and Controls in the Diagnostics Views	23
Common Diagnostics View Controls.....	24
Exporting Views to HTML Reports.....	34
Using Table Header Controls	35
Chapter 3: Viewing Diagnostics Data in Detail Layout	41
About the Diagnostics Detail Layout	42
Working With View Titles and View Filters.....	44
Controlling the Appearance of Charted Metrics	49

Chapter 4: Working with the Details Pane	57
About the Details Pane	57
Updating Custom Attributes	60
Charting a Metric in the Graph	62
Charting a Metric in an Incident	62
Setting Metric Thresholds	63
Investigating a Threshold Violation	63
Configuring Metrics	65
Chapter 5: Working with Detailed Layout Graphs	67
About Detailed Layout Graphs	68
Displaying Entities and Metrics in the Graph	71
Viewing Multiple Metrics on a Graph	71
Viewing the Data Behind a Charted Metric on the Graph	73
Chapter 6: Working with the Graph Entity Table	77
About the Graph Entity Table	78
Understanding the Columns in the Graph Entity Table	79
Customizing the Graph Entity Table	81
Working with the Graph Entity Table and the Graph	82
Searching for an Entity in the Graph Entity Table	83
Viewing Additional Entity Information	84
Performing Common Tasks for a Selected Entity	85
Chapter 7: Alert Notification	91
About Alert Notification	91
Configuring Alert Notification	92
Working with Alert Notification Rules	97
Reviewing Alert Notification Events	103
Chapter 8: Performing Incident Analysis	107
About Incident Analysis	108
Monitoring Performance Versus Analyzing Incidents	109
Create New Incidents	110
Renaming an Incident	111
Deleting an Incident	112
Using the Analyze Incidents View	112
Chapter 9: Instance Trees	121
Introducing Instance Trees	122
Solving Problems with Instance Trees	124
Cross-VM Trees	128
When Instance Trees Are Not Sufficient	130
Aggregate Trees	131

Chapter 10: Customizing Diagnostics Views	137
About Mercury Diagnostics Custom Views.....	138
Understanding Diagnostics Custom View Retention	138
Creating View Groups	139
Hiding View Groups.....	140
Saving a Customized View	142
Creating a New View	144
Renaming a View.....	148
Deleting a View	149
Modifying a Custom View	150
Sharing Custom Views	150
Upgrading Custom Views From Previous Diagnostics Versions.....	151

PART III: UNDERSTANDING DIAGNOSTICS STANDARD VIEWS

Chapter 11: Hosts View	155
Using the Hosts View	156
Description of the Hosts View.....	156
Accessing the Hosts View	157
Customizing the Hosts View	158
Interpreting the Hosts View	158
Drilling Down from the Hosts View	160
Maintaining an Alert Rule or Comments	160
Chapter 12: Load View	161
Using the Load View	161
Description of the Load View.....	162
Accessing the Load View	163
Customizing the Load View	164
Interpreting the Load View	164
Drilling Down to a Layer in the Graph Entity Table	167
Chapter 13: Outbound Calls View	169
About the Outbound Calls View.....	170
Accessing the Outbound Calls View	170
Description of the Outbound Calls View.....	171
Drilling Down from the Outbound Calls View	172

Chapter 14: Probes View	173
Using the Probes View.....	174
Description of the Probes View	175
Accessing the Probes View	176
Customizing the Probes View	177
Interpreting the Probes View	177
Drilling Down from a Probe in the Graph Entity Table	180
Chapter 15: SQL Statements View	181
Using the SQL Statements View.....	181
Accessing the SQL Statements View.....	184
Description of the SQL Statements View	185
Chapter 16: Server Requests View	187
Using the Server Requests View	187
Accessing the Server Requests View	189
Customizing the Server Requests View	190
Interpreting the Server Requests View	191
Drilling Down from a Server Request in the Graph Entity Table	194
Drilling Down to an Instance Tree for a Server Request.....	195
Chapter 17: Status View	201
About the Status View	202
Accessing the Status View	204
Customizing the Status View	205
Interpreting the Status View	206
Drilling Down from the Status View.....	208
Chapter 18: Topology View	213
About the Topology View	214
Description of the Topology View	215
Accessing the Topology View.....	221
Working with the Topology Diagram.....	221
Drilling Down from the Topology View.....	226
Chapter 19: Transactions View	227
Using the Transactions View.....	228
Description of the Transactions View.....	228
Accessing the Transaction View.....	229
Customizing the Transactions View	230
Interpreting the Transactions View	230
Drilling Down from a Transaction in the Graph Entity Table	232

Chapter 20: Trended Methods View	233
Using the Trended Methods View.....	233
Accessing the Trended Methods View	235
Description of the Trended Methods View	236
Chapter 21: Layers View	239
Using the Layers View	240
Description of the Layers View	241
Accessing the Layers View	242
Customizing the Layers View.....	243
Interpreting the Layers View	243
Drilling Down from a Layer in the Graph Entity Table.....	245
Chapter 22: Call Profile View	247
Using the Call Profile View	248
Description of the Call Profile View.....	249
Accessing the Call Profile View	254
Interpreting the Call Profile View	254
Analyzing J2EE to SAP R/3 Remote Function Calls.....	256
Analyzing Remote Method Invocation (RMI).....	257
Analyzing Life Cycle Methods for Portlets.....	257
SOAP Fault Call Profiles.....	257

PART IV: UNDERSTANDING DIAGNOSTICS SPECIALIZED VIEWS

Chapter 23: Portal Views	261
Using the Portal Views	261
Chapter 24: Portal Components View	265
Using the Portal Components View.....	266
Description of the Portal Components View	266
Accessing the Portal Components View	267
Customizing the Portal Components View	268
Interpreting the Portal Components View	268
Drilling Down from a Portal Component in the Graph Entity Table	269
Chapter 25: Web Services Views	271
About Web Services Diagnostics Views.....	272
Using the Web Services Views.....	273
Viewing Inbound Web Service Calls.....	274
Viewing Outbound Web Service Calls	278
Viewing Web Services Correlations.....	279
Viewing SOAP Faults	281

Chapter 26: SAP Views	285
Using the SAP Views.....	285
Chapter 27: Oracle Database Views	293
Using the Oracle Views	293
Chapter 28: BEA WebLogic Views	297
Using the BEA WebLogic Views	297
Chapter 29: IBM WebSphere Views	299
Using the IBM WebSphere Views.....	299
Chapter 30: CICS Views	303
Using the CICS Views.....	303

PART V: USING THE MERCURY DIAGNOSTICS PROFILER FOR J2EE

Chapter 31: Using the J2EE Diagnostics Profiler	307
Accessing the Mercury Diagnostics Profiler for J2EE	308
Mercury Diagnostics Profiler for J2EE Processing	309
Common J2EE Diagnostics Profiler Tab Navigation and Display Controls	311
Chapter 32: Analyzing Method Latency with J2EE Diagnostics Profiler Tabs	313
Analyzing Performance Using the Summary Tab	314
Analyzing Performance Using the Hotspots Tab	318
Analyzing Performance Using the Metrics Tab.....	320
Analyzing Performance Using the Threads Tab.....	322
Analyzing Performance Using the All Methods Tab.....	328
Analyzing Performance Using the All SQL Tab	331
Analyzing Performance Using the Exceptions Tab.....	333
Analyzing Performance Using the Server Requests Tab.....	335
Analyzing Performance Using the Call Profile Window	338
Analyzing Performance Using the Web Services Tab	345
Using the Configuration Tab	347
Chapter 33: Analyzing Memory with J2EE Diagnostics Profiler Tabs	349
Analyzing Memory Using the Collections Tab	349
Analyzing Memory Using the Allocation Analysis Tab	353
Analyzing Memory Using the Memory Analysis Tab	358
Analyzing Memory Using the Heap Breakdown Tab.....	359

PART VI: USING THE MERCURY DIAGNOSTICS PROFILER FOR .NET

Chapter 34: Using the .NET Diagnostics Profiler	365
Accessing the .NET Diagnostics Profiler.....	366
Mercury Diagnostics Profiler for .NET Processing.....	367
Common .NET Profiler Tab Navigation and Display Controls	369
Chapter 35: Analyzing Method Latency with .NET Diagnostics Profiler Screens	371
Analyzing Performance Using the Server Requests Tab.....	372
Analyzing Performance Using the SQL Tab	377
Analyzing Performance Using the Methods Tab	380
Analyzing Performance Using the Exceptions Tab.....	383
Analyzing Performance Using the Call Tree Tab	385
Chapter 36: Analyzing Memory Using .NET Diagnostics Profiler Screens	393
Analyzing Memory Using the Collections Tab	393
Analyzing Memory Using the Heap Tab	399

PART VII: DIAGNOSTICS INTEGRATION WITH OTHER MERCURY PRODUCTS

Chapter 37: Viewing Diagnostics Data in Mercury Business Availability Center	407
About Viewing Diagnostics Data in Mercury Business Availability Center	408
Accessing the Diagnostics Screens.....	409
Monitoring Diagnostics Performance Data from Dashboard	409
Drilling Down to Diagnostics from Dashboard	417
Diagnostics Performance Reports in Mercury Business Availability Center	421
Drilling down to Diagnostics Data from Mercury Business Availability Center Reports	425
Chapter 38: Viewing Diagnostics Data in LoadRunner	435
About Viewing Mercury Diagnostics Data in LoadRunner 8.1	435
Configuring LoadRunner Scenarios to use Mercury Diagnostics	436
Drilling Down to Diagnostics Data During a Load Test Scenario.....	440
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	442

Chapter 39: Viewing Diagnostics Data in Performance Center	443
About Viewing Mercury Diagnostics Data	
in Performance Center 8.1	444
Configuring Performance Center Load Tests to Use	
Mercury Diagnostics	445
Drilling Down to Diagnostics Data During a Load Test	450
Analyzing Offline Diagnostics Data Using Mercury	
LoadRunner Analysis	452
Index.....	453

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.

This chapter describes:	On page:
How This Guide Is Organized	xii
Mercury Diagnostics Online Documentation	xiii
Additional Online Resources	xv
Documentation Updates	xv
Typographical Conventions	xvi

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 Working with Diagnostics Views

Describes how to work with the Mercury Diagnostics views once the components are installed and configured.

Part III Understanding Diagnostics Standard Views

Describes each the Mercury Diagnostics standard views.

Part IV Understanding Diagnostics Specialized Views

Describes each the Mercury Diagnostics specialized view groups.

Part V Using the Mercury Diagnostics Profiler for J2EE

Describes how to use the Mercury Diagnostics Profiler for J2EE.

Part VI Using the Mercury Diagnostics Profiler for .NET

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

Part VII Diagnostics Integration with Other Mercury Products

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

Mercury Diagnostics Online Documentation

Your Mercury Diagnostics application comes with the following documentation:

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

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

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

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

Additional Online Resources

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

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

Documentation Updates

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

To download updated documentation:

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

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

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

Typographical Conventions

This guide uses the following typographical conventions:

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

Part I

Introduction

1

Mercury Diagnostics Product Overview

This chapter introduces you to Mercury Diagnostics 6.5. 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	9
Viewing Diagnostics Data	11
Accessing the Mercury Diagnostics Views	16
Interpreting Diagnostics Data	19

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, Portlets 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 probes that gather your applications performance metrics, the Mercury Diagnostics Profiler is automatically installed as well. The Profiler is an independent diagnostics application that can be accessed either directly through the built in Profiler UI or through the Mercury Diagnostics UI.

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 37, “Viewing Diagnostics Data in Mercury Business Availability Center.”

Integration with LoadRunner and 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 38, “Viewing Diagnostics Data in LoadRunner.” For more information about the Diagnostics integration with Performance Center, see Chapter 39, “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 if the Mercury Application Delivery or 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:

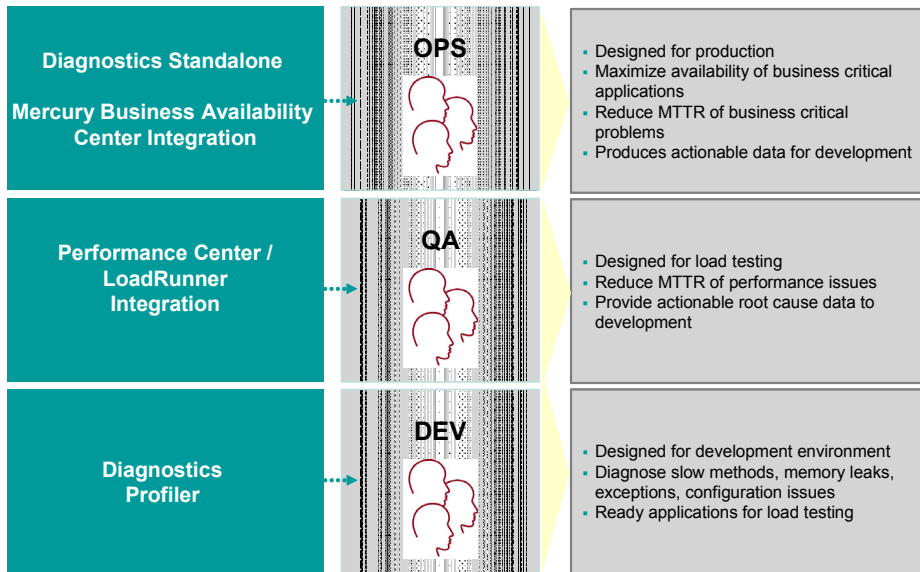
- ▶ where time is spent in an application either processing data or waiting for a response from another part of the application.
- ▶ the slowest layers.
- ▶ the slowest server request which are the application entry points.
- ▶ outliers to help diagnose intermittent problems.
- ▶ threads that may be contributing to performance issues.
- ▶ memory problems and garbage collection issues.
- ▶ 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 31, “Using the J2EE Diagnostics Profiler.” For more information about using the Mercury Diagnostics Profiler for .NET, see Chapter 34, “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 Probes.** Responsible for capturing events from your application, aggregating the metrics, and sending the performance metrics to a Diagnostics Server. The Probe captures events such as method invocations, the beginning and end of business transactions, and server transactions.

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

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

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

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

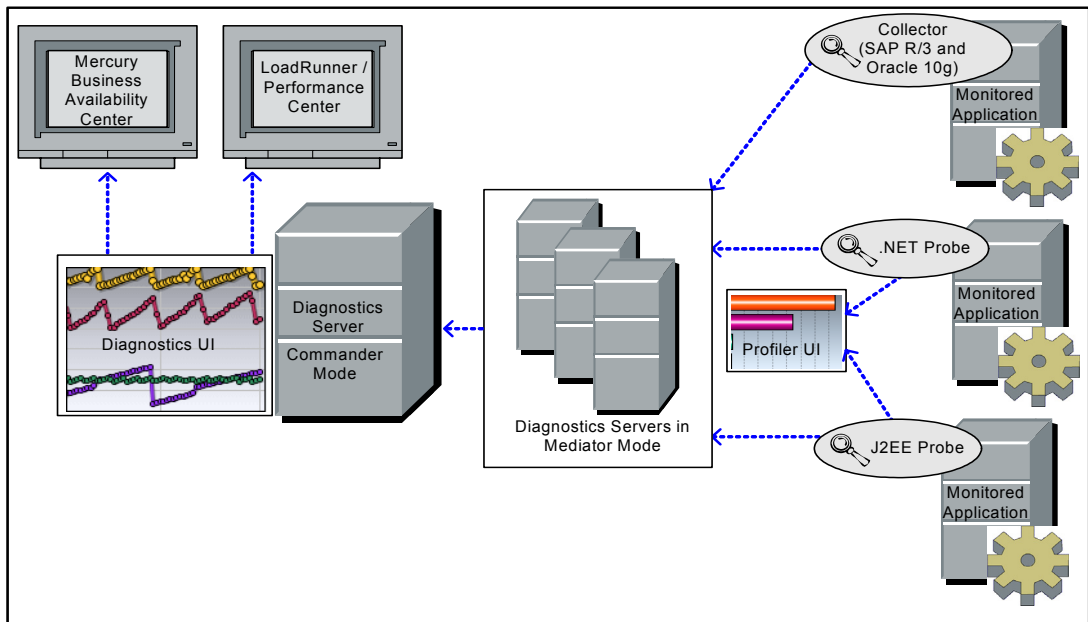
The Diagnostics Server in Commander mode is responsible for the command and control functions between the various Diagnostics components and the components of the other Mercury products with which Diagnostics is working. The Diagnostics Server in Commander mode keeps track of the location and status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

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

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

Diagnostics Data Flow

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



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

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

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

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

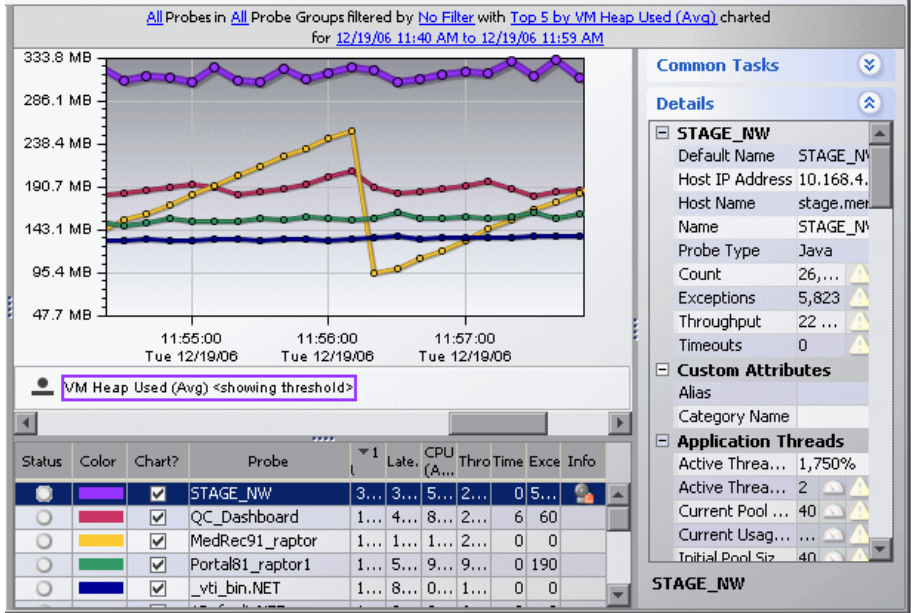
Viewing Diagnostics Data

Diagnostics presents the performance metrics for your applications in a variety of view layouts including: detail layouts, dashboard layouts, status layouts, call profile layouts, and topology layouts. The views show the performance metrics in both tabular and graphical formats. A variety of controls let you customize the way that you see the data. Navigation controls let you drill down from the metrics to observe underlying behaviors.

For more information, see Part II, “Working with Diagnostics Views.”

Detail Layout Views

Diagnostics comes with a set of predefined detail layout 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 layout views for that entity. The Probes view shown below is an example of a detail layout view:

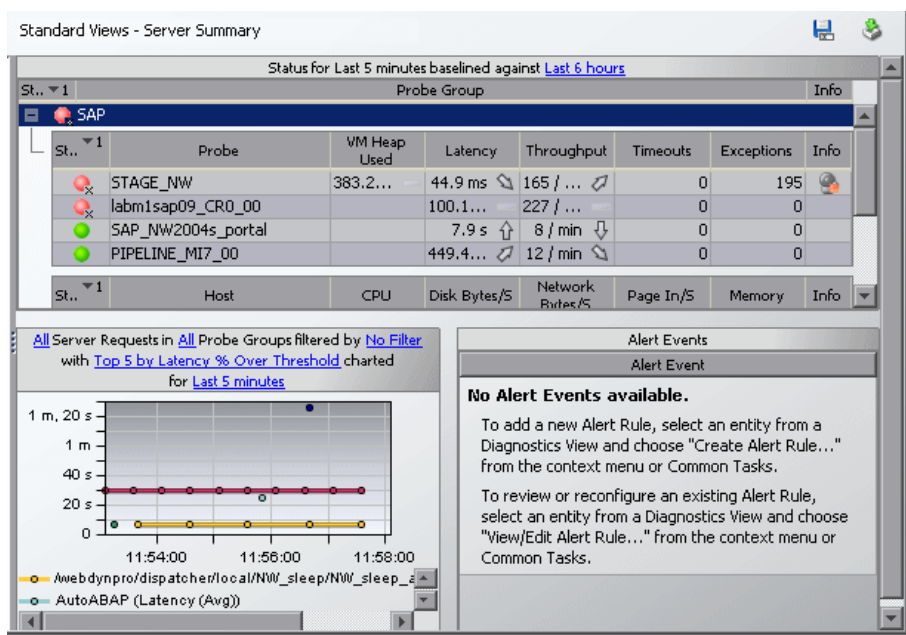


For more information on the features and controls of the detail layout, see Chapter 3, "Viewing Diagnostics Data in Detail Layout."

For information about using a specific standard detail layout view, see Part III, "Understanding Diagnostics Standard Views."

Dashboard Layout Views

Dashboard layout views present key performance characteristics of your applications at a glance. The dashboard layout views can be made up of two or more detail, status, Alert Event, or Topology views. In a view with a dashboard layout, selected views are displayed in an abbreviated format known as a concise view. You can create customized Dashboard layout views, or use one of the predefined views installed with Diagnostics. The Server Summary view shown below is an example of a predefined view with a dashboard layout:



A standard or custom view opens in a dashboard as a concise version of the view, where only the graph and the graph legend are included. You may manipulate the view filters that are included in the concise view title just as you can in the detail layout view. You can also see the tooltips for the nodes of the charted trend lines and the edges of the stacked areas by holding your mouse pointer over the appropriate location in the graph. When you hold your mouse pointer over the items in the legend in the concise views, Diagnostics displays a tooltip with additional information.

To see the full-size version of a concise view, double-click the concise view. Diagnostics displays the full-size version of the detail layout view that you selected and sets the bread crumb to indicate that you navigated from the Dashboard view.

For more information on detail layout views, see Part III, “Understanding Diagnostics Standard Views.”

Status View

The *Status View* is a table displaying the probe groups, their probes, and the hosts for those probes. For each level in the table, Diagnostics provides a status of each component compared to the thresholds that you set for the performance metrics. In addition, you can see the performance trends for each component. The trending indicator shows whether metrics performance trend is going up or down and at what rate.

You can drill down to the probe group, probe, or host to see the detail layout view for the component and to understand the alert status. Below is an example of the Status view:

The screenshot shows the Status View interface with the following components labeled:

- Probe Group Status:** Points to the top-level status bar.
- Probe Group:** Points to the 'Default' group.
- Probe:** Points to the 'WL81-W5' probe.
- Host:** Points to the 't-ic8.lab.performant.com' host.
- Status View Title:** Points to the title 'Status for Last 5 minutes baselined against Last 6 hours'.
- Baseline Control:** Points to the 'Last 6 hours' link.
- Probe Group Headers:** Points to the top row of the table.
- Host Status:** Points to the green status icon for the host.
- Threshold Alert Indicator:** Points to the red '65%' value in the CPU column.
- Trend Indicator:** Points to the up/down arrows next to the CPU values.
- Host Table Header:** Points to the header row of the host table.
- Probe Status:** Points to the green status icon for the probe.
- Probe Table Header:** Points to the header row of the probe table.

Group	Component	Metric	Value	Trend
Default	WL81-W5 (Probe)	VM Heap Used	42.1 MB	
		Latency	1.0 s	
		Count/S	0.033	
		Timeouts	0	
		Exceptions	5	
t-ic8.lab.performant.com (Host)	CPU	65%		↕
	Disk Bytes/S	13.6 KB		↕
	Network Bytes/S	926.9 KB		↕
	Page In/S	0		
	Memory	82%		
WAS_server1 (Probe)	VM Heap Used	83.1 MB		
ALMAGRO (Host)	CPU	2%		↕
	Disk Bytes/S	17.2 KB		↕
	Network Bytes/S	21.7 KB		↕
	Page In/S	2		
	Memory	44%		
almagro.lab.performant.com (Host)	CPU	12%		↕
	Disk Bytes/S	781.8 KB		↕
	Network Bytes/S	25.3 KB		↕
	Page In/S	165		
	Memory	97%		

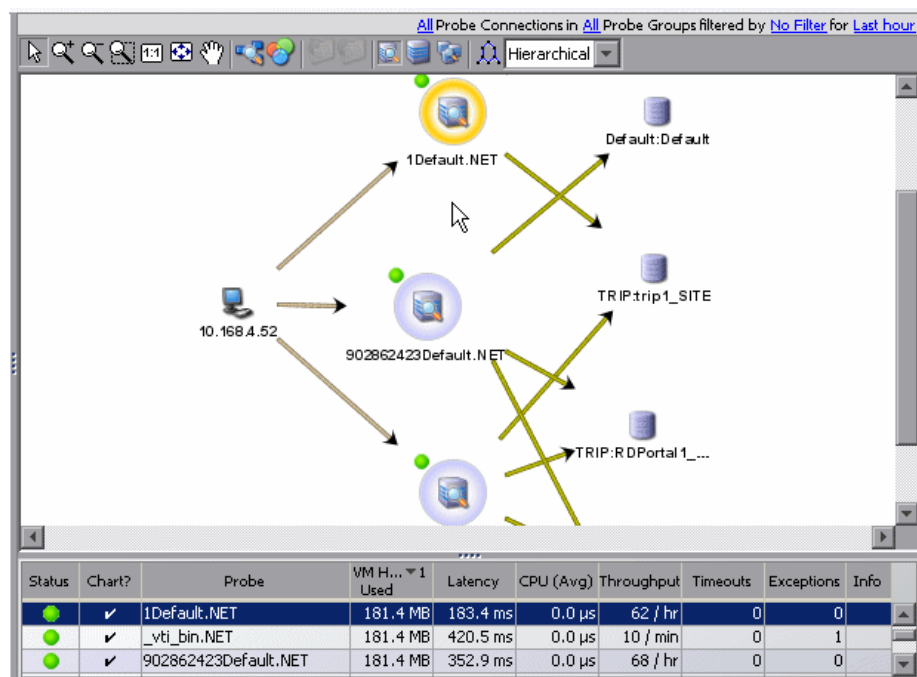
For more information on the features and controls of the Status view, see Chapter 17, “Status View.”

Topology View

The *Topology View* displays a pictorial graph of the network topology for your monitored applications. The graph shows the links between the components that make up your deployment, along with the monitoring information that depicts the performance of the components and their business processes.

To display metrics for an entity in the Details pane, you select the probe or probe connection from the tabs in the graph entity table. The metrics for the selected entity are displayed in the Details pane.

The following example shows the topology graph for a selected probe:



For more information on the features and controls of the Topology view, see Chapter 18, “Topology View.”

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.

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

Accessing Diagnostics from Mercury Business Availability Center

Note: Before you can access the Diagnostics views from Mercury Business Availability Center, you must configure Mercury Business Availability Center with the details for the Mercury Diagnostics Server. 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 37, “Viewing Diagnostics Data in Mercury Business Availability Center.”

Accessing Diagnostics from LoadRunner

Note: Before you can use Mercury Diagnostics with LoadRunner, you must ensure that you have configured LoadRunner with the information for the Diagnostics Server, according to the instructions 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 listed transaction.

For more information, see Chapter 38, “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 configured Performance Center with the information for the Diagnostics Server, according to the instructions 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 39, “Viewing Diagnostics Data in Performance Center.”

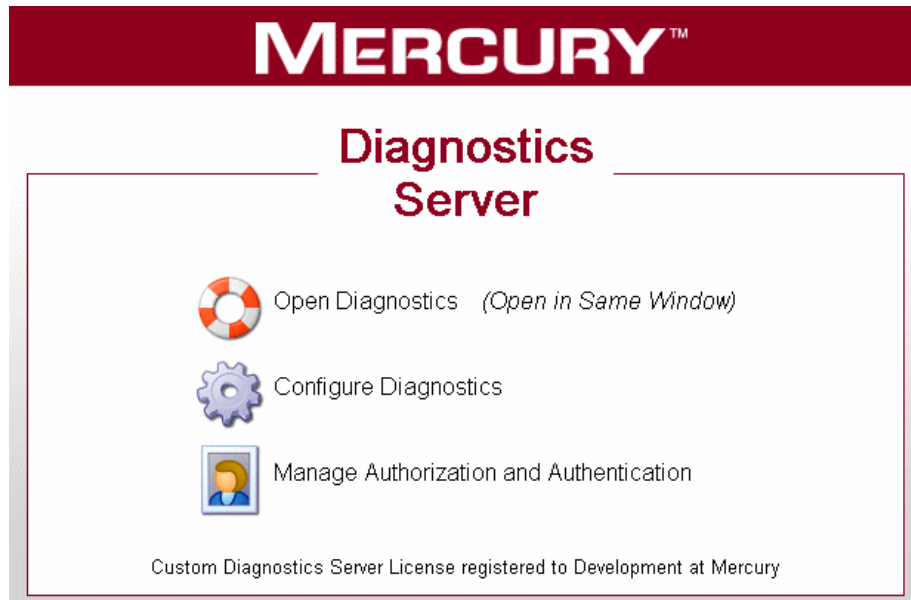
Accessing Diagnostics from the Diagnostics Server in Commander mode

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**. 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 is able to see the Diagnostics views. One of the default user names that you can use is **admin**; the default 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**.

Interpreting Diagnostics Data

The following sections explain the manner in which Diagnostics presents the performance information in its views.

About Layers and Sub-Layers

Mercury Diagnostics groups the performance metrics for the classes and methods invoked by your applications into *layers* and *sub-layers* based on the resources that the application invokes to perform the processing. These layers help 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 “Layers View” on page 239.

About Time Measurements in Diagnostics

It is important to understand how Diagnostics calculates and presents the time measurements in the Diagnostics views. When analyzing the Diagnostics metrics and comparing them with the metrics 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. This can cause the response times in a Diagnostics view and the response times on a LoadRunner / Performance Center screen to differ.
- ▶ The server side may also have multiple threads executing simultaneously. Even though these threads are run in parallel, executing simultaneously this can cause the aggregate server side time to be much larger than the time as measured by the client.
- ▶ 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.

Part II

Working with Diagnostics Views

2

Common Features and Controls in the Diagnostics Views

This chapter describes the features and controls that are common to all of the Diagnostics views.

This chapter describes:	On page:
Common Diagnostics View Controls	24
Exporting Views to HTML Reports	34
Using Table Header Controls	35

Common Diagnostics View Controls

The features and controls that are common to most of the Diagnostics views are described below using the following annotated screen image:

The screenshot shows the Mercury Diagnostics interface with several annotated components:

- View-Specific Information:** Located at the top left, it includes the breadcrumb path: `Standard Views - Probes (Portal81_raptor1) > Server Requests (CallChain for NII test) > Layers`.
- Diagnostics Viewing Time Filter:** Located at the top center, it shows the current view is for the `Last 5 minutes`.
- Diagnostics Toolbar:** Located at the top right, it contains icons for navigation and analysis, such as `Monitor and Investigate` and `Analyze Incidents`.
- View Toolbar:** Located at the far top right, it includes a `Maintenance` button.
- View Title & View Filters:** Located on the left side, it points to the breadcrumb path and the `Layers` tab in the left-hand menu.
- View bar:** Located on the left side, it points to the `Server Requests` menu item.
- Table Headers:** Located on the left side, it points to the `Layers` table below the chart.

The central chart, titled `Layers of CallChain for NII test on Portal81_raptor1 in Default Probe Group filtered by No Filter with Custom charted for Last 5 minutes`, displays latency contribution over time. The Y-axis represents latency in seconds (0 to 6 s), and the X-axis shows time from 11:42:00 to 11:43:00 on Wed 12/06/06. The chart shows three distinct peaks, with the middle peak being the highest and colored green and yellow.

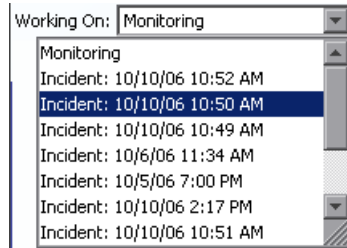
Below the chart is a table titled `Latency Contribution` with the following data:

Color	Chart?	Layer	Contribution to Parent	Latency Contributi...
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Outbound Calls	83%	4.5 s
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Database	9%	499.9 ms
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Business Tier	7%	408.4 ms
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Web Tier	0%	1.5 ms
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Directory Service	0%	127.0 μs

On the right side, the `Details` panel shows `Outbound Calls` with a `Contribution` of 4.5 s and 83%, and a `Load` of 0.09. Below this, the `Outbound Calls` section is visible.

Working On Drop-down

The **Working On** drop-down allows you to control whether the views in the Monitor and Investigate tab are in Monitoring mode or in Incident Analysis mode. In Monitoring mode, the views contain performance metrics for the current processing of your applications. In Incident Analysis mode, the views contain performance metrics for a selected incident.



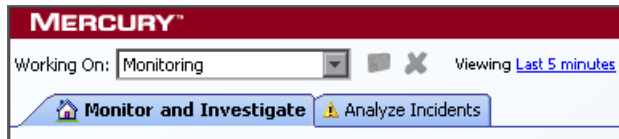
To view performance metrics for your application's current processing, select **Monitoring** from the **Working On** drop-down. Selecting **Monitoring** places Diagnostics in Monitoring mode. When in monitoring mode, the **Monitor and Investigate** tab presents the Diagnostics views with the application's performance metrics for the time period indicated in the Viewing drop-down.

To view the performance metrics for a particular incident, select the incident from the **Working On** drop-down. Selecting an incident from the **Working On** drop-down puts Diagnostics in Incident Analysis mode. When in Incident Analysis mode, the **Monitor and Investigate** tab presents the Diagnostics views with the performance metrics for the time of the selected incident.

For more information on Incident Analysis see Chapter 8, "Performing Incident Analysis."

Monitor and Analyze Tabs

Diagnostics displays its views on two tabs: the Monitor and Investigate tab and the Analyze Incidents tab. Diagnostics displays different information in these tabs depending on whether you have selected Monitoring mode or Monitor and Investigate mode. The mode is determined by your selection from the **Working On** drop-down.

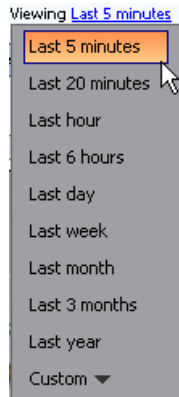


When in monitoring mode, the **Monitor and Investigate** tab presents the Diagnostics views with the application's performance metrics for the time period indicated in the Viewing drop-down. The **Analyze Incidents** tab is not used when in Monitoring mode and instead contains instructions for creating a new incident.

When in Incident Analysis mode, the **Monitor and Investigate** tab presents the Diagnostics views with the performance metrics for the time of the selected incident. The **Analyze Incidents** tab contains all of the entities and their metrics that you have included in the incident. For more information on Incident Analysis, see Chapter 8, "Performing Incident Analysis."

Diagnostics Viewing Time Filter

Using the **Viewing Time** filter you can set the time period that Diagnostics uses to display the performance metrics in all of its views. You may select one of the default time ranges, or enter a custom time range.



The time range that you select from the **Viewing Time** filter is used in all of the Diagnostics views unless you have locked a view for a specific time period. You may override the **Viewing Time** filter time period for selected views by locking a view at a specific time period using the view time range filter in the view title as described in “Filtering the View by Time” on page 48.

When you select a time period from the **Viewing Time** filter, the information in the graphs, the graph entity tables, and the Details pane of the Diagnostics Views are updated to correspond to the selected time range resolution. Diagnostics continues to update the views so that the most recent information for the selected time range resolution is always displayed.

Setting a Custom Time Range

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:

The image shows a dialog box for setting a custom time range. It consists of two calendar views side-by-side, both for November 2006. The left calendar is titled "Start Date and Time:" and the right is titled "End Date and Time:". In the "Start Date and Time:" calendar, the date 2 is highlighted in orange. Below this calendar are three dropdown menus for time selection: "12", "00", and "AM". In the "End Date and Time:" calendar, the date 2 is also highlighted in orange. Below this calendar are three dropdown menus for time selection: "11", "55", and "PM". At the bottom of the dialog are two buttons: "OK" and "Cancel".

After selecting the start and end dates, click **OK** to instruct Diagnostics to refresh the current view.

Chart Update Frequency

When you view the performance metrics, note that the charted metrics are updated at different rates, depending on the selected time range resolution. 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 is approximately 5 seconds. However, after an hour into the scenario, the update frequency decreases to approximately 4 minutes.

Understanding Time Range Resolution

The actual time period included in the displayed metrics is 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 to include only information from the previous 5 minutes.

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.

To focus on performance for an exact 20-minute period, you can use the zoom features, as described in “Controlling the Appearance of Charted Metrics” on page 49.

Using the Viewing Time Filter in Incident Analysis Mode

When you are in Incident Analysis mode, Diagnostics sets the **Viewing Time** filter to the time of the selected incident. This time is used for the Diagnostics views displayed in the **Monitor and Investigate** tab and in the Analyze Incident tab. You may set the **Viewing Time** filter so that you can see the metrics included in the incident at a more current time period without leaving Incident Analysis mode.

Zooming and Scrolling Controls

The zooming and scrolling controls allow you to manipulate the charted metrics on the graphs that appear in dashboard and detail layout views.



These controls are described in “Controlling the Appearance of Charted Metrics” on page 49.

Breadcrumb Trail

The breadcrumb 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 navigation to retrace your steps in your performance analysis. Clicking a level in the breadcrumb 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 current Diagnostics view.

For information on drilling down from one view to another, see “Drilling Down to Other Diagnostics Views” on page 86.

Diagnostics Toolbar

The **Diagnostics Toolbar** contains the following buttons:



Turn On/Off Active Alert Notifications. 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 7, “Alert Notification.”



Show Help. 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 going to display 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.



View Toolbar

The **View Toolbar** contains the following buttons:



Save This View. If you have customized the Diagnostics view, 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 10, “Customizing Diagnostics Views.”



Export to HTML. This selection allows you to save the currently displayed metrics to a report that can be recalled for viewing or sharing with others. For more information, see “Exporting Views to HTML Reports” on page 34.

View Bar

The **View Bar** contains view groups which in turn contain the views that you use to analyze the performance of your applications. The views are organized into view groups to make it easier for you to locate them.

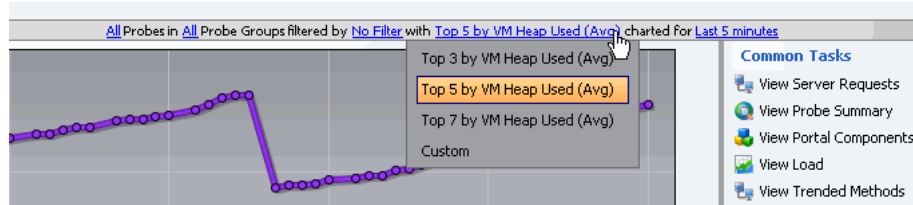
Predefined view groups that are installed with Diagnostics contain views are intended to assist you in diagnosing problems in particular situations or for specific environments. The **Standard Views** view group contains views that are useful in most situations. These views are described in Part III, “Understanding Diagnostics Standard Views.”

The other view groups contain views that are useful in more specific situations. These views are described in Part IV, “Understanding Diagnostics Specialized Views.” 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.

You may save customized versions of the predefined views or create original views using the functions of the View bar. The My Views group contains your customized views. The processes for maintaining customized views and view groups are described in Chapter 10, “Customizing Diagnostics Views.”

View Title and View Filters

The view title contains the filters that you can use to control the information that is displayed in the Diagnostics views. The filters that are included in the view title vary depending on the current view.



The view titles are formatted into a sentence structure so that they are easy to read and understand. The view filters are drop-downs within the text of the view title. Changing the selected value for a filter causes Diagnostics to update the view to display the information based upon the new filters.

Summary or Detail Layout View Specific Information

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

For information about the dashboard views, see “Dashboard Layout Views” on page 13.

For information about the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

For information about the status views, see Chapter 17, “Status View.”

Table Headers

The table headers in the Status view and detail layout 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, see “Using Table Header Controls” on page 35.

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 screen. 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.

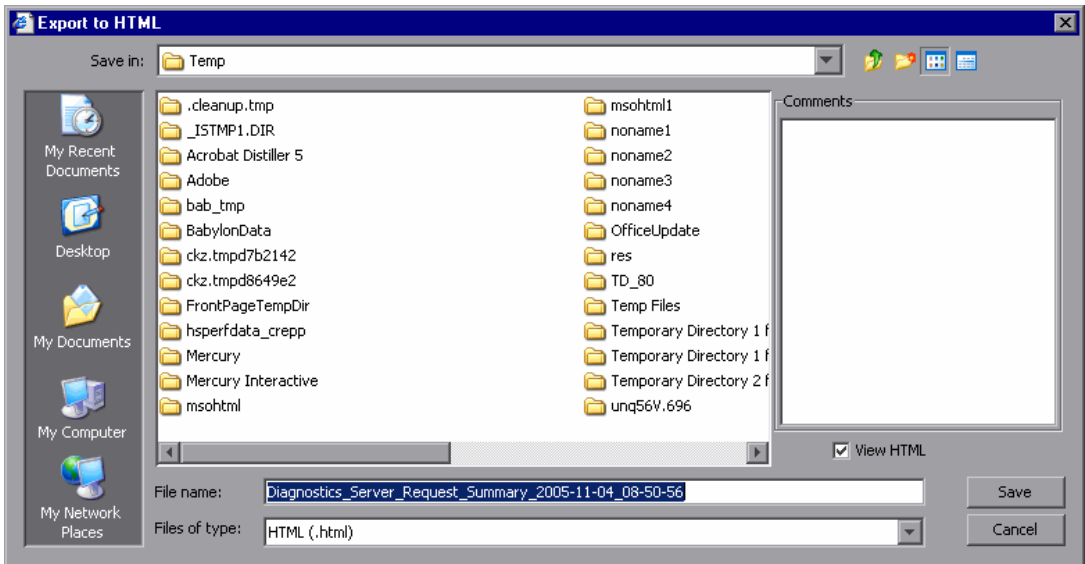
Exporting Views to HTML Reports



To save the performance metrics displayed in a Diagnostics view, click 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.

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, for example, instructions or questions.
- 4** To view the newly created 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

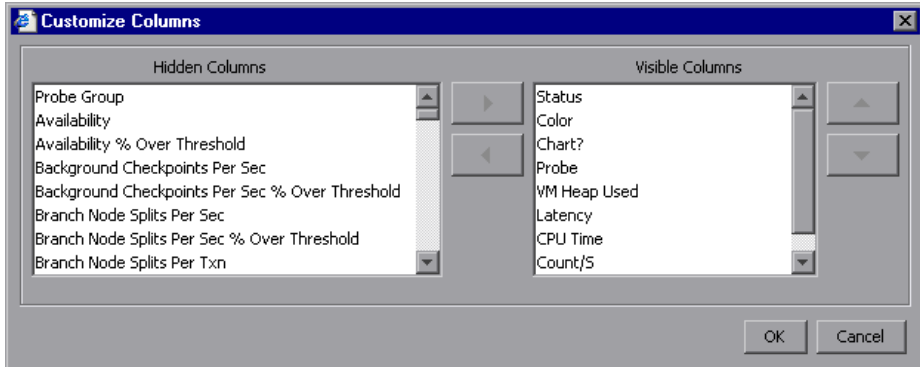
The table headers that appear in the Status view and in the detail layout provide controls that enable you to specify which columns are to appear in the table and the order in which the columns are to be displayed, as well as specifying which columns are to determine the sort order for the rows in the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

Customizing the Table Columns

The default columns that appear in the Status view and in the graph entity table of the detail layout 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.

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 layout views, and would like to save the customization for future use, save the view as a custom view. If you do not save the changes as a custom view, they are lost when you close the Diagnostics UI.

For information on saving custom views, see “Saving a Customized View” on page 142.

Sorting Rows in the Table

Most of the Diagnostics tables are sorted in 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:

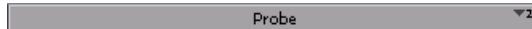
Status	Color	Chart?	Probe	VM Heap Used	Latency	Count/S
--------	-------	--------	-------	--------------	---------	---------

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:

Latency

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 is to 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 is to 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.

3

Viewing Diagnostics Data in Detail Layout

Diagnostics presents the performance metrics for your applications in a variety of views with different layouts including: detail layouts, dashboard layouts, status layouts, call profile layouts, and topology layouts. The views with detail layout share many common features and controls. The sections that follow introduce these common features and controls.

This chapter describes:	On page:
About the Diagnostics Detail Layout	42
Working With View Titles and View Filters	44
Controlling the Appearance of Charted Metrics	49

About the Diagnostics Detail Layout

There are many common elements that appear when data is presented in the detail layout. These are highlighted in the following image:



- **View Title and View Filter.** The *view title* contains the *view filters* that you use to control the type and scope of the information that is displayed in a Diagnostics view. For more information on the view filters, see “Working With View Titles and View Filters” on page 44.
- **Graph.** The *graph* contains trend lines or stacked areas that represent the performance metrics. For more information about the graph in the detail layout, see Chapter 5, “Working with Detailed Layout Graphs.”
- **Graph Legend.** The *graph legend* displays the line patterns used to represent metrics on the graph. For more information on using the graph legend, see Chapter 5, “Working with Detailed Layout Graphs.”

- ▶ **Graph Entity Table.** The *graph entity table* lists the entities associated with a particular view. For example, the entities displayed in the Probes view are probes, and in the Load view they are layers. For more information about working with and customizing the graph entity table, see Chapter 6, “Working with the Graph Entity Table.”
- ▶ **Details Pane.** The *Details pane* presents the metrics and custom attributes associated with the entity selected in the Graph Entity Table. From the Details pane, you control which metrics Diagnostics charts in the graph and which metrics are included in the active incident in Incidents Analysis. You may also set thresholds that determine at what point you want to be warned that the value of a performance metric may be of concern. For more information about the Details pane, see Chapter 4, “Working with the Details Pane.” For more information on investigating alert conditions for thresholds that have been exceeded, see “Investigating a Threshold Violation” on page 63.

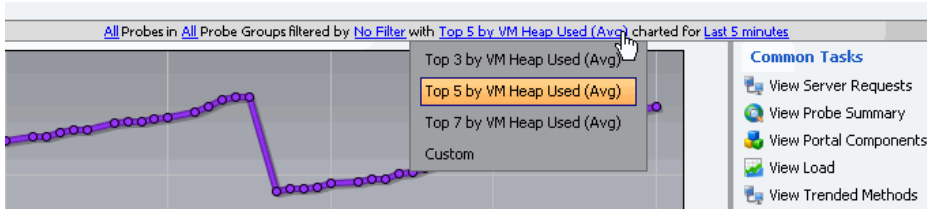
Resizing Panes in the Detail Layout

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.

Working With View Titles and View Filters

The view filters, a set of drop-down menus that are included in the view title of many of the Diagnostics views, are used to control the information displayed in the graph, graph entity table, and Details pane. The filters that appear in each view title vary according to the displayed view. To set a filter, click a link, and select a value from the drop-down menu.

Below is an example of the view title and the view filters from the Probes view. The title includes the Probe Type **All**, Probe Group **All**, Custom Filter **No Filter**, Graph **Top 5 by VM Heap Used (Avg)**, and Time Range **Last 5 M\minutes** menus. The **Graph** filter menu is open displaying the available options.



The time range (locked/unlocked) and probe group filter settings apply as you navigate to all drilldown views. Filter settings at lower level views, after drilldown, do not apply as you navigate back up to higher level views.

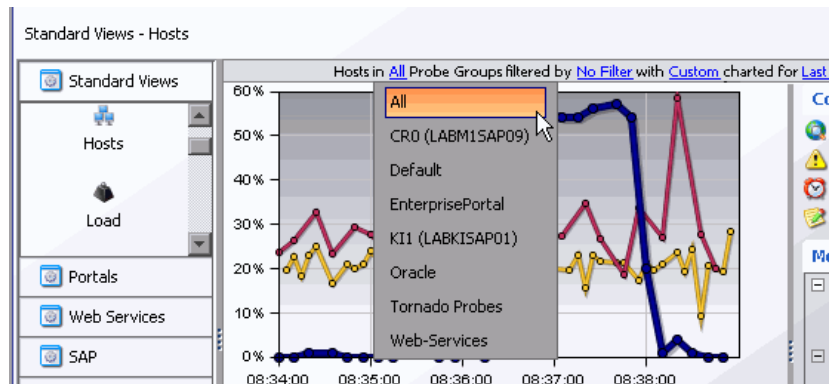
Filtering the View by Type

The **Type** filter appears in some of the Diagnostics views. This filter varies according to the type of entities in the detail layout. The following is an example of the **Type** filter menu for the Probes view:



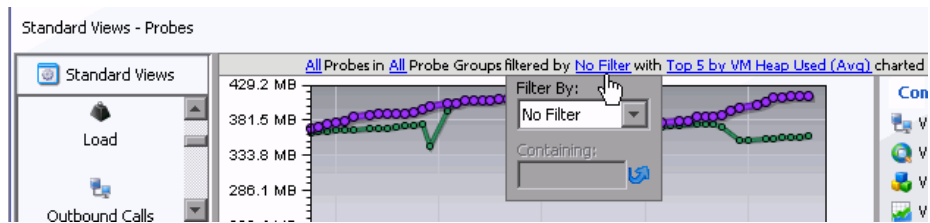
Filtering the View by Probe Group

You use the **Probe Group** filter display metrics for a specific probe group, or for all probe groups that are defined for the monitored application. The following is an example of a **Probe Group** filter menu:



Filtering the View by Custom Filters

You use the **Custom** view filter to display entities that match a specific value of an entity's attribute. For example, you can filter by Probe Groups containing Production. This filter varies according to the type of entities in the detail layout. The following is an example of the **Custom** filter menu:



To select a Custom Filter:

- 1 Click the link to open the **Custom** filter menu.
- 2 Choose a filter from the **Filter By** menu.
- 3 Enter a filter value. To view a list of applicable filter values, check the values in the graph entity table and the Details pane.
- 4 Apply the filter by clicking **Enter** or the Apply Change icon.



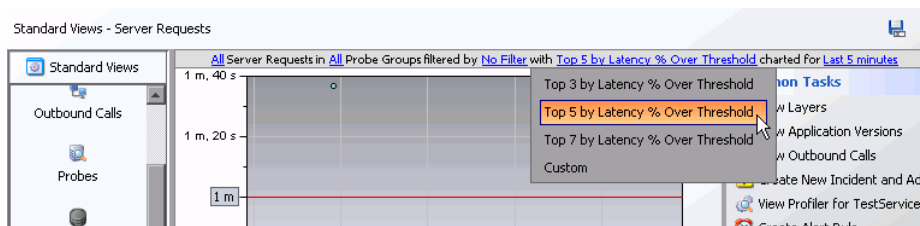
The following image shows an example of the **Custom** filter after a filtering attribute has been selected.



- 5 To remove a custom filter, select the **NoFilter** attribute and click ENTER.

Filtering the Graph Metrics

You use the **Graph** filter to determine which entities from the graph entity table should have their metrics charted on the graph. The choices in the **Graph** filter vary according to the type of entities in the detail layout. These filters are based on the values of metrics or thresholds, and the filter is reapplied whenever the view is refreshed. For this reason, the graph entities and metrics will frequently change. 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 graph entity 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 graph entity table become fixed and change only when you chart different entities, or when you select another option from the **Graph** filter menu. For more information about selecting entities in the graph entity table to be charted on the graph, see “Adding and Removing Entities from the Graph” on page 82.

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

Filtering the View by Time

If you do not modify this filter, the view time is determined by the time period set initially in the Diagnostics global **Viewing Time** filter, as described in “Diagnostics Viewing Time Filter” on page 27.

In this view, you use the **charted for** filter to lock a view at a specific time range so it will override any changes to global Diagnostics **Viewing Time** filter.

To lock the time range for a view, select the **charted for** filter in the view title and select **Lock time for this view** from the filter menu as shown below:



The value of the **charted for** filter in the view title indicates that the time range for the view is locked.

To unlock the time range for a view select the **charted for** filter in the view title and select **Use global time for this view** from the filter menu:



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

Controlling the Appearance of Charted Metrics

Diagnostics enables you to control the appearance of the graphs by zooming in and out and pausing the current time.

Zooming on the Graph

Diagnostics provides two different types of zooming: *magnification zooming* and *resolution zooming*. It automatically determines which type of zooming to use based on the selected time range and *zoom range*.

Understanding Magnification Zooming

Magnification zooming provides a magnified view of the data points. Diagnostics does not retrieve any additional information when portraying a magnified view of the performance metrics. There are two cases when Diagnostics uses a magnification zoom:

- ▶ when the graph has been plotted using data that had been aggregated into the highest resolution data points
- ▶ when you select a zoom range that spans a time period that requires too much data to be retrieved for an efficient resolution zoom

For example, 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 layout 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. In this situation, where the graph data 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 provides 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.



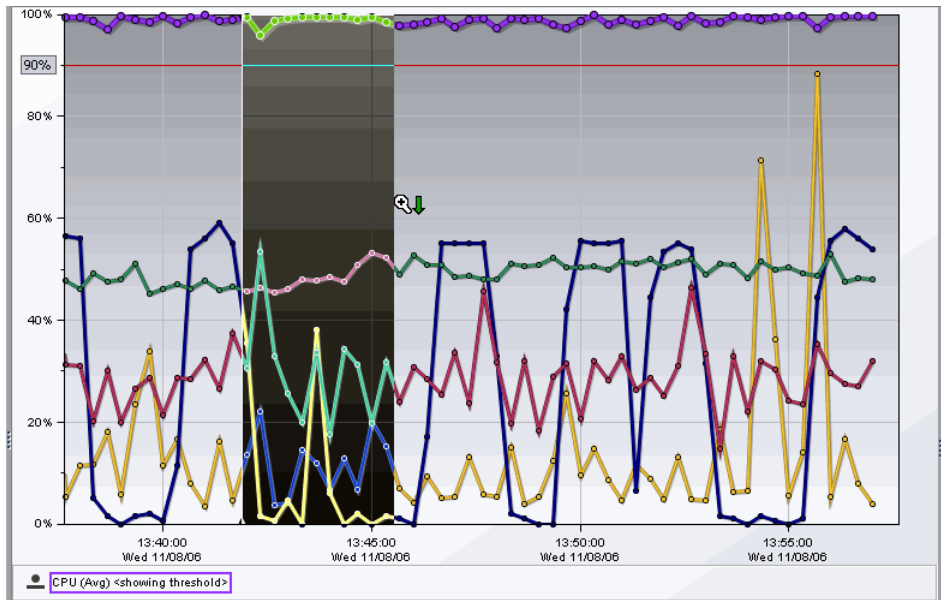
- For a description of magnification zooming, see “Understanding Magnification Zooming” on page 49.



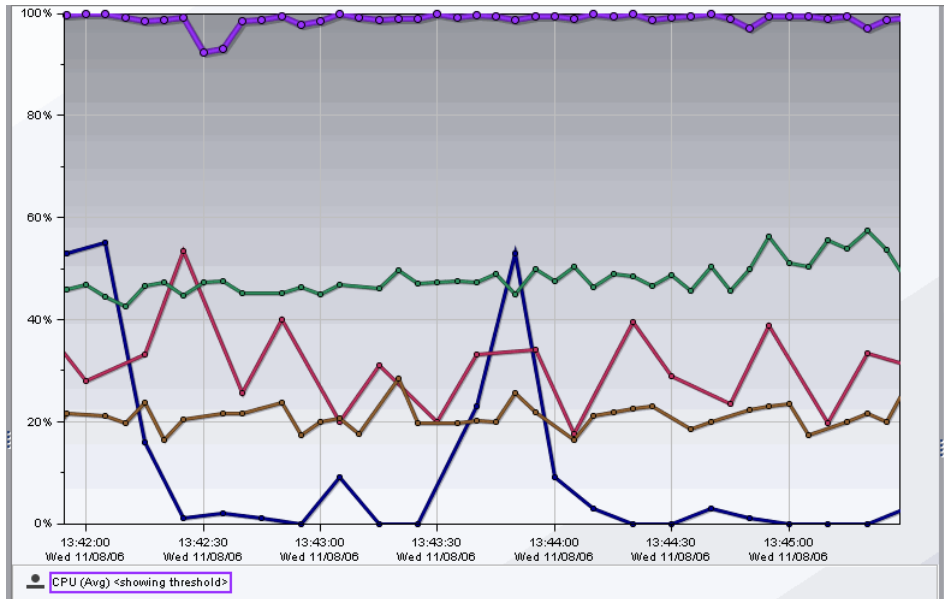
- For a description of resolution zooming, see “Understanding Resolution Zooming” on page 50.

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 **13:42** through **13:45**.

- ▶ 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 **13:42** and after **13:45**.

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 the graph.

- ▶ If Diagnostics zoomed using a resolution zoom, the selection in the **Time Range** view filter is set to the specific time range for which Diagnostics retrieved in order to display the zoom range that you indicated.

Zooming Out of Charted Metrics



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 lower resolutions and wider periods of time. When Diagnostics has zoomed back to the time ranges listed in the **Viewing Time Range** filter, it adjusts the selection in the **Viewing Time Range** filter to correspond to the time range charted in the graph as a result of zooming out.

Pausing and Panning the Diagnostic Views

The **Pause** and **Pan** buttons at the top of the Diagnostics interface allow you to stop Diagnostics from updating the views so that you can analyze the data that is currently displayed before it is replaced by more current data.



- ▶ Click **Pause** to stop charting new data in the graph. The graph remains unchanged when this button is selected.



- ▶ Click **Pan Left** to smoothly scroll the graph from the current time period to the time period with the same duration that immediately preceded the one displayed.



- ▶ Click **Pan Right** to smoothly scroll the graph from the current time period to a later time period than the one that is currently charted.



- ▶ Click **Play** 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**.

4

Working with the Details Pane

When you select an entity in the Graph Entity Table, the Details Pane displays performance metrics and custom attributes for the selected entity.

This chapter describes:/In this lesson, you will learn about:	On page:
About the Details Pane	57
Updating Custom Attributes	60
Charting a Metric in the Graph	62
Charting a Metric in an Incident	62
Setting Metric Thresholds	63
Investigating a Threshold Violation	63
Configuring Metrics	65

About the Details Pane

Using the Details pane, you can:

- ▶ control the metrics that Diagnostics charts in the graph
- ▶ specify the metrics included in an incident
- ▶ set thresholds for the metrics

The Details pane consists of two sections. The top portion of the table is a list of metrics and their values, grouped by metrics categories. Beneath the list of metrics is the control area where you can set a threshold for a selected metric. The metrics displayed in the table can be Java metrics, system metrics, or JMX metrics.

The screenshot shows the 'Details' pane for 'MSPetShop.NET'. The table below lists various metrics and their values, with icons for threshold violation indicators and charting controls.

Metric Categories	Threshold Violation Indicators	Threshold Icon
MSPetShop.NET		
Default Name	MSPetShop.NET	
Host IP Address	10.168.4.187	
Host Name	merlot.mercury.cc.il	
Name	MSPetShop.NET	
Probe Type	.NET	
Count	1	⚠️ 📊
Exceptions	0	⚠️ 📊
Throughput	12 / hr	⚠️ 📊
Timeouts	0	⚠️ 📊
Custom Attributes		
Latency (Selected Metric)		
% Over Threshold	788,689%	
Avg	71.0 ms	⚠️ 📊
CPU (Avg)	0.0 μs	⚠️ 📊
CPU (Total)	0.0 μs	⚠️ 📊
Contribution to Probe Group	3%	
Load	0.0	⚠️ 📊
Max	71.0ms	⚠️ 📊
Min	71.0 ms	⚠️ 📊
Standard Deviation	0.0 μs	⚠️ 📊
Total	71.0 ms	⚠️ 📊
Probe		
VM Heap Committed (Avg)	281.2 MB	⚠️ 📊
VM Heap Used (Avg)	281.2 MB	⚠️ 📊
Latency (Avg) (Selected Metric in Control Area)		
Threshold:	9.0 μs	⚠️ 📊

Annotations in the image point to the following elements:

- Metric Categories:** Points to the left column of the table.
- Threshold Violation Indicators:** Points to the yellow warning triangle icons.
- Threshold Icon:** Points to the small icon next to the threshold value in the control area.
- Selected Metric:** Points to the 'Latency' row in the table.
- Selected Metric in Control Area:** Points to the 'Latency (Avg)' section at the bottom.
- Threshold for Selected Metric:** Points to the '9.0 μs' value in the control area.
- Add to Current Incident Control:** Points to the warning triangle icon in the control area.
- Metric Charting Control:** Points to the charting icon in the control area.

Selecting a Row from the Graph Entity Table

The Details pane displays the metrics for the entity that you selected from the graph entity table.

The header of the first category in the Details pane displays the name of the selected entity. The image above displays the performance metrics for the entity MSPetShop.NET.

Viewing Metrics in the Details Pane Area


Diagnostics groups the entity attributes in the Details pane into categories to make it easier to understand the performance characteristics of the selected entity and to make it easier for you to find an attribute in the listed metrics. You can expand or collapse the list of metrics in a category.

In the example above, the category MSPetShop.NET contains more details about the probe entity that was selected from the graph entity table, the metric category **Latency** contains the metrics that depict the latency for the processing on the probe.

Reviewing the Details and Settings for a Metric

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

The metric details that can be included in the control area are:

- ▶ **Metric Name:** This is displayed for all metrics listed in the Details pane.
In the image above, the name of the metric in the control area is **Latency (Avg)**. Because the selected entry in the table, **Avg**, is part of the **Latency** category the metric is actually **Latency (Avg)**.
- ▶ **Threshold:** If the selected metric can have an associated threshold, a **Threshold** icon  appears in the Details pane, along with the current threshold value.

In the image above, the threshold for **Latency (Avg)** is set to 9.0 ms.

Updating Custom Attributes

The Custom Attributes category in the Details pane lists attributes associated with the selected entity. The custom attributes that are listed in a view vary depending on the current view. Most views have an **Alias** attribute and a **Category Name** attribute. Using the **Alias** attribute you can assign names to entities so that they are more recognizable in the Diagnostics views. The **Category Name** attribute allows you to assign a name to an entity that can be used to filter and group the information that is displayed in a Diagnostics view. Diagnostics Inbound Web services views include a **Web Service Alias** attribute that allows you to change the displayed name of a Web service.

Setting Custom Attributes Values

You can assign values to the attributes in the Custom Attributes category of the Details pane by selecting the row with the attribute and entering the values directly into the field in the row. The value that you enter is applied to all occurrences of the entity throughout Diagnostics.

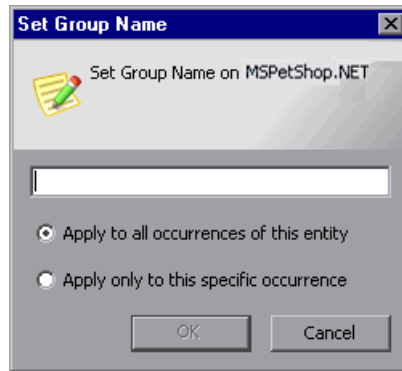
To alter the scope of the change so the change applies to either the selected occurrence or to all occurrences, you use the pop-up dialogs to assign values to the custom attributes.

You need to have **change** permissions to set custom attribute values. For more information about user permissions, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

Note: The scope for custom attributes is limited to the occurrences of the entity for a given Diagnostics Server in Mediator mode. Occurrences of the entity that are captured by probes reporting to a different Diagnostics Server in Mediator mode will not be impacted by the custom attributes.

To define the scope of the custom attributes:

- 1 Select the row for the custom attribute that you want to update from the Details pane.
- 2 Click the button that appears at the end of the selected custom attributes row. Diagnostics displays a dialog box similar to the following:



- 3 Enter the value for the attribute.
- 4 Select the scope for the attribute, **Apply to all occurrences of this entity** or **Apply only to this specific occurrence**.
- 5 Click **OK** to accept and apply your changes.

Charting a Metric in the Graph

When the **Metric Charting** icon appears in a row in of the Details pane, the metrics in the row can be charted in the graph.



When the icon does not have check mark, the metric is not charted in the graph. You can cause the metric to be charted by clicking the **Metric Charting** icon in the row or by right-clicking on the row and selecting **Start Charting this Metric**.



When the icon includes a check mark, the metric is charted in the graph. You can remove the metric from the graph by clicking the **Metric Charting** icon in the row or by right-clicking on the row and selecting **Stop Charting this Metric**.

Charting a Metric in an Incident



The rows that contain metrics that can be included in a Diagnostics Incident are marked with the Incident Analysis icon. The Incident Analysis icon behaves differently depending on whether you are in Incident Analysis mode or Monitoring mode.

Including a Metric in a New Incident

When you are looking at a Diagnostics view in Monitoring mode, clicking the Incident Analysis icon causes Diagnostics to create a new incident that includes the entity selected from the graph entity table with the attribute selected from the Details pane charted in the graph.

Including a Metric in an Active Incident

When you are looking at a Diagnostics view on the Monitoring and Investigate tab while in Incident Analysis mode, clicking the Incident Analysis icon adds and removes the attribute selected from the Details pane from the incident.



When the Incident Analysis icon includes a check mark, clicking the icon causes Diagnostics to remove the selected attribute from the active incident.



When the Incident Analysis icon does not include a check mark, clicking the icon causes Diagnostics to add the selected attribute to the active incident. This metric will then be charted in the active incident.

Setting Metric Thresholds



When you select a metric that has a threshold value, a **Threshold** box appears below the Details pane in the metric control area. You can adjust the threshold value by typing over the existing value and clicking the **Apply Change** button.

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.

If the value of the selected metric exceeds the new threshold value, a visual threshold violation indicator is immediately shown in the Details pane.

Investigating a Threshold Violation

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 threshold, Diagnostics displays threshold violation indicators in the Details pane, in the status column of the graph entity table, and in the cell of the graph entity table where the metric is displayed.

In the Details pane, a threshold violation is indicated by outlining the cell that contains the metric value with the color that is appropriate for the severity of the violation. The row in the table that contains the category name for the metric that violated the threshold is highlighted with the same color as well.

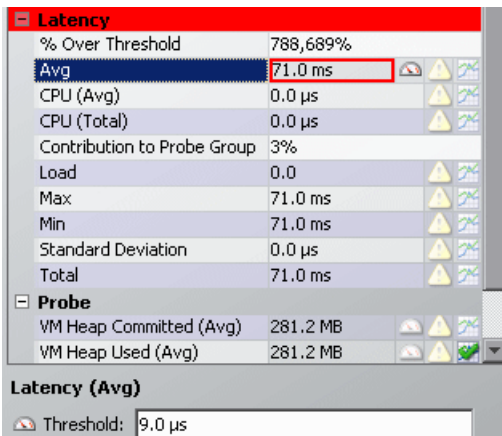
The color of the threshold violation indicator determines its severity:

- ▶ **None.** Normal - There has been no threshold violation or there is no threshold defined for the metric.
- ▶ **Yellow.** Warning - The component is occasionally but not consistently exceeding the defined threshold.
- ▶ **Red.** Critical - The component is consistently exceeding defined thresholds.

It is important to note that the threshold violation indicators persist longer than the actual threshold violation. This means that even after the metric is no longer exceeding the threshold, Diagnostics continues to indicate that a threshold violation has occurred. Once a metric has exceeded its threshold, the threshold violation indicator is displayed until there have been less than three threshold violations for five minutes.

Threshold violation indicators can progress in severity from no violation, to yellow, and then to red as the metric's performance deteriorates. However, the violation indicators do not regress in the same way. Once a red threshold violation indicator has been issued, it will continue to be displayed as red until the metric value has returned to normal levels.

In the following example, the cell containing the value of the **Latency (Avg)** metric is outlined in red and the category for the metric has a red background because the metric value, **15.9 s**, exceeded the threshold of **14.0 s**.



Configuring Metrics

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

Configuring the CPU Time Java Metric

The **CPU Time** metric appears in the Details pane 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 `<probe_install_directory>\etc\capture.properties` must be set to **true**
- ▶ The **cpu.timestamp.collection.method** property in `<probe_install_directory>\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.

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

5

Working with Detailed Layout Graphs

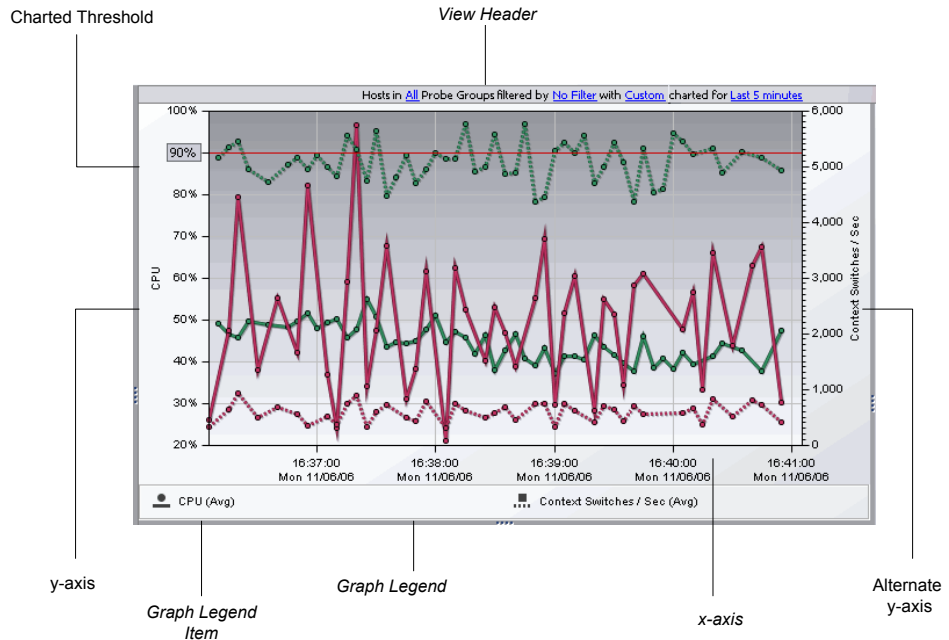
The graph in a detailed layout contains trend lines or stacked areas that represent the performance metrics.

This chapter describes:/In this lesson, you will learn about:	On page:
About Detailed Layout Graphs	68
Displaying Entities and Metrics in the Graph	71
Viewing Multiple Metrics on a Graph	71
Viewing the Data Behind a Charted Metric on the Graph	73

About Detailed Layout Graphs

You use the view filters, graph entity table, and Details pane to control the information that appears in the graph.

The following image provides an overview of the common graph navigation and display controls:



- ▶ **View Title.** The *view title* describes the view using the current settings for the view filters. You can update the view filters to change the information that is displayed in the view as described in “Working with Detailed Layout Graphs” on page 67.
- ▶ **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 **Last 5 Minutes** was selected from the **Time Range** filter, and that each interval across the axis represents a one 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 CPU utilization, and that CPU utilization is charted in percentage points.
- ▶ **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 Context Switches per second and that each tick mark on the axis represents 100 context switches.

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 an alternate way of removing the metric from the graph instead of using the controls in the Details pane. For more information about charting metrics from the Details pane, see “Charting a Metric in the Graph” on page 62.

- ▶ **Move to New Chart.** When more than one metric has been charted in the graph, the menu contains an option to move the metric to a new graph. This can allow you to make a graph less cluttered so that you can see the information more clearly. Selecting this option causes Diagnostics to open a new graph for the metric. All trend lines, stacked areas, and thresholds for the indicated metric are moved from the original graph to the new graph.
- ▶ **Move to Chart with...** When more than one graph appears in the graph area, the menu contains an option to move a metric that appears on one graph to another graph. Selecting this option causes Diagnostics to move all trend lines, stacked areas, or threshold indicator for the selected metric from the original graph to the graph indicated in the selected menu option.

Note: You will not be able to move a charted metric to a graph that is already using the y-axis and the alternate y-axis when the metric to be moved is not compatible with the y-axis.

- ▶ **Display Threshold.** This option is only active when the metric can have a metric threshold assigned.
 - ▶ 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 previously charted, the threshold for that metric is removed from the graph because only one threshold can be charted on a graph.
 - ▶ Selecting this option when the threshold is already charted causes the threshold line to be removed from the graph.

Displaying Entities and Metrics in the Graph

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

- ▶ Using the **Chart** column in the graph entity table. See “Adding and Removing Entities from the Graph” on page 82.
- ▶ From the **Graph** filter in the View Filters area. See “Filtering the Graph Metrics” on page 47.
- ▶ From the **Details** pane. See “Charting a Metric in the Graph” on page 62.

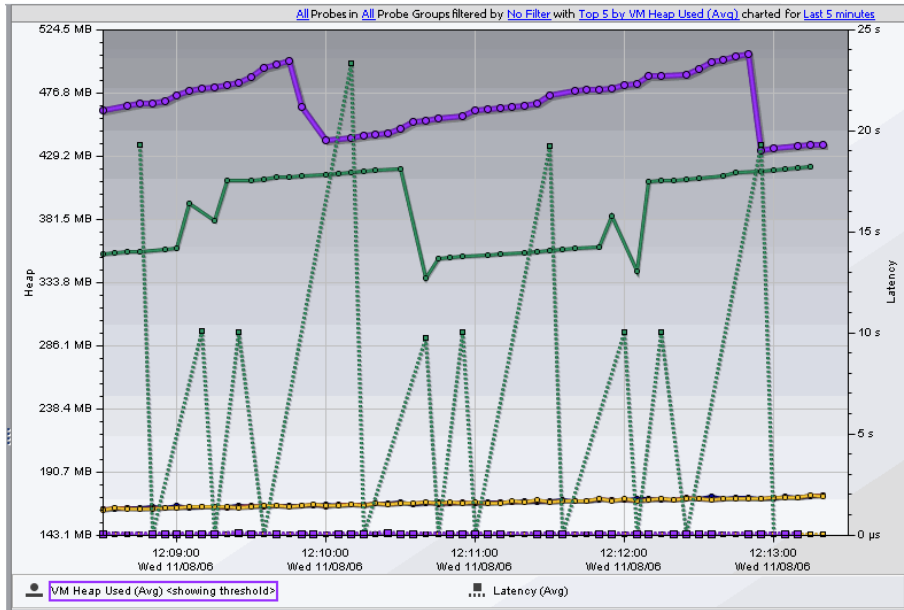
Viewing Multiple Metrics on a Graph

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 graph entity table. Each metric selected from the Details pane is charted for each entity selected from the graph entity table, so long as a valid metric value exists for the entity.

Chapter 5 • Working with Detailed Layout Graphs

In the following image, both the Heap and Latency metrics are charted for the four probe entities selected from the graph entity table. The pattern used to chart metrics for each entity are shown in the graph legend.



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 are charted in the graphs.

Identifying the Entity in the Graph Entity Table for a Charted Metric

From the graph, you can associate a trend line or stacked area with the entity in the graph entity 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 graph entity table. At the same time, the Details pane is updated to display all of the metrics for the selected entity.

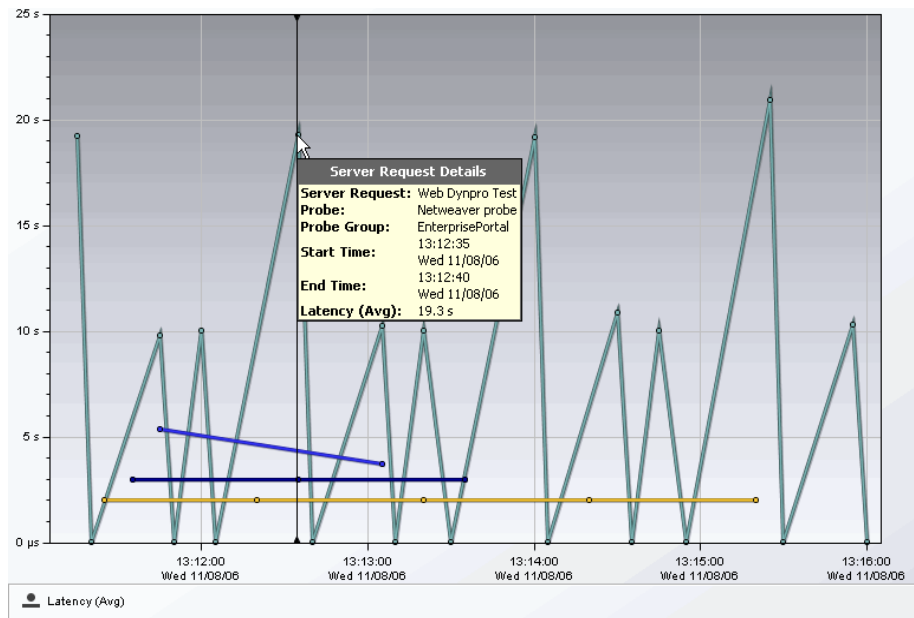
When you select a trend line in the graph, all of the trend lines for the metrics charted for the entity are also highlighted.

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, Diagnostics displays a tooltip, containing details for the underlying data point, as shown in the following example:



The information in the tooltip varies depending on the detail layout view that Diagnostics has displayed. The tooltips can include:

- **Title Bar.** The title bar identifies the type of entity to which the data point belongs. In the example above, the title bar is **Server Request Details**, which indicates that the tooltip contains the details for a metric displayed in the Server Requests 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 **Server Request**, and the name of the entity is **Web Dynpro Test**.
- **Aggregation Period Start Time.** The **Start Time** entry contains the starting time of the aggregation of the metrics for the data point.
- **Aggregation Period End Time.** The **End Time** entry contains the time that metrics for the data point were last gathered.
- **Metric Name.** The metric name identifies the name and the value for the charted metric. In the example above, the metric is **Latency (Avg)** and its value is **19.3 s**.

6

Working with the Graph Entity Table

The detail layout graph entity table lists the entities for which metrics have been gathered for the specified time period. For example, on the Probes view, the entities are probes and on the Load view, the entities are layers.

This chapter describes:	On page:
About the Graph Entity Table	78
Understanding the Columns in the Graph Entity Table	79
Customizing the Graph Entity Table	81
Working with the Graph Entity Table and the Graph	82
Searching for an Entity in the Graph Entity Table	83
Viewing Additional Entity Information	84
Performing Common Tasks for a Selected Entity	85

About the Graph Entity Table

From the graph entity table you can select the entities that are to be charted on the graph. The graph entity table is highly customizable allowing you to control the columns that appear in the table and the sort order for the rows in the table. The following image provides an overview of the common navigation and display controls that are included the graph entity table:

Status	Color	Chart?	Server Request	Probe	Latency % Over Thresh...	Latency	Throughput	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"/>	/patient/login.do	AMKILAB03	-99%	557.5 ms	0.013	
		<input checked="" type="checkbox"/>	/patient/editprofile.do	AMKILAB03	-99%	358.5 ms	0.007	
		<input type="checkbox"/>	TraderService::buy	T-LC7	-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	

- **Graph Entity Table Header.** From the graph entity 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 Graph Entity Table” on page 81.


For more information about sorting the rows in the graph entity table, see “Sorting Rows in the Table” on page 37.

- **Entity Rows.** The entities in the table correspond to those presented in the graph. The entity shows you the details of the performance of your application and enables you to drill down to more performance details.

The entities that are listed in the table and the order in which they appear changes as Diagnostics updates the information in the view.

Understanding the Columns in the Graph Entity Table

Several of the columns that appear in the graph entity table are the same for most of the predefined detail layouts. Other columns may be specific to a particular view. The columns common to most of the predefined detail layout views are:

- 
Status. This indicator shows how the listed entity or any of its child entities is performing relative to the thresholds that you set for their metrics.

For example, a probe can have a critical status even when none of the probe's own metrics have exceeded their thresholds. The probe's status will become critical when any of the server requests for that probe become critical.

The color of the indicator stands for the severity of the threshold violation. If you hold the mouse pointer over the status indicator in a row of the table, Diagnostics displays a tooltip with a description of the current status:

Status Indicator Color	Description
Green	Good - The entity 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 entity 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 entity 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.

If the metrics that are exceeding the threshold are displayed in the row of the graph entity table, they will be highlighted with a threshold violation indicator.

It is important to note that the threshold violation indicators persist longer than the actual threshold violation. This means that even after the metric is no longer exceeding the threshold, Diagnostics continues to indicate that a threshold violation has occurred. Once a metric has exceeded its threshold, the threshold violation indicator is displayed until there have been less than three threshold violations for five minutes.

Threshold violation indicators can progress in severity from no violation, to yellow, and then to red as the metric's performance deteriorates. However, the violation indicators do not regress in the same way. Once a red threshold violation indicator has been issued, it will continue to be displayed as red until the metric value has returned to normal levels.

For information about setting thresholds, see “Setting Metric Thresholds” on page 63.

- ▶ **Color.** This column contains a colored block. When the **Chart** check box for an entity in the graph entity 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 the entity's metrics appear in the graph with the same color.
- ▶ **Chart.** This column contains a check box to select the entities whose metrics should appear in the graph. When the entity is selected, Diagnostics charts a trend line or stacked area for each of the entity's metrics selected in the Details pane. For more information about charting the metrics for an entity in the graph entity table, see “Adding and Removing Entities from the Graph” on page 82.

- ▶ **Metric Columns.** The metric columns that Diagnostics displays in the graph entity table vary depending on the displayed view and table customizations. If the value of a metric in a metric column has exceeded the threshold that you defined for the metric, the cell in the metric column that contains the offending metric is displayed with a threshold violation indicator. The threshold violation indicator appears as a colored outline, either yellow or red. This threshold violation indicator corresponds to threshold violation indicator for the metric in the Details pane.

For information about setting and understanding thresholds, see “Setting Metric Thresholds” on page 63.

- ▶ **Info.** This column indicates that the entity has alert rules or comments. One or more of the following icons can appear in this column:



- ▶ **Active Alert Rule.** Indicates that an alert notification rule has been created for the selected entity and the rule is active.



- ▶ **Disabled Alert Rule.** Indicates that an alert notification rule has been created for the selected entity but the rule is currently disabled.



- ▶ **Custom Comments.** Indicates that comments have been created for the selected entity.

For more information about Alert Notification Rules, see Chapter 7, “Alert Notification.” For more information on entering comments for an entity, see “Managing Comments for an Entity” on page 88.

Customizing the Graph Entity Table

The graph entity table headers contain controls that allow you to specify which columns Diagnostics is to display in the table, and the column order. 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 35.

Working with the Graph Entity Table and the Graph

From the graph entity table, you can:

- control the entities for which Diagnostics charts metrics in the graph
- find more information about the table entities and metrics

Identifying Charted Metrics for an Entity in the Graph Entity Table

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

Adding and Removing Entities from the Graph

The **Chart** column on the graph entity 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**. You may then repopulate the graph by clicking the **Chart** check box for the entities whose metrics you would like to have charted. You can also use the **Graph** filter in the **View Title** to let Diagnostics pick significant sets of entities to chart.

When you click the **Chart** check box for any entities in the graph entity table, Diagnostics changes the **Graph** filter in the **View Title** to **Custom** to indicate that you have manually selected the entities. For more information about the **Graph** filter, see “Filtering the View by Custom Filters” on page 46.

Searching for an Entity in the Graph Entity Table

You can search for an entity in the graph entity table.

To find an entity in the graph entity table:

- 1 Press CTRL + F. Diagnostics displays the **Search For:** pop-up above the graph entity table column headers as shown in the following image:

Status	Color	Chart?	Server Request	Probe	Latency % Over Thresh...	Latency	Throughput	Info
		<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 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 graph entity table.
- 4 If no match is found, Diagnostics changes the color of the text in the **Search For:** pop-up to red, and adds the **(no match)** string, following your search criteria.
- 5 To exit the search, press Esc.

Viewing Additional Entity Information

The columns in the graph entity table contain a subset of the attributes and performance metrics that Diagnostics has gathered for the listed entities. However, the information in the table 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 graph entity table.

Viewing Tooltips for the Graph Entity Table Cells

Many of the cells in the rows of the graph entity 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 Graph Entity Table” on page 79.
- ▶ 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 U. ▼ 1	Latency	CPU (Avg)
		<input checked="" type="checkbox"/>	Netweaver probe	386.3 MB	6.8 s	459.1 ms
		<input checked="" type="checkbox"/>	ALMAGRO	385.0 MB	7.2 ms	3.6 ms
		<input checked="" type="checkbox"/>	WV01MedDes	187.0 MB	35.6 ms	
		<input checked="" type="checkbox"/>	x2		8.0 ms	8.0 ms
		<input checked="" type="checkbox"/>	Ce		6.7 ms	3.1 ms
		<input type="checkbox"/>	M8		26.0 ms	5.0 ms
		<input type="checkbox"/>	Te		62.1 ms	4.5 ms
		<input type="checkbox"/>	Ja		15.2 ms	8.6 ms
		<input type="checkbox"/>	Testservice.webservice.NET	95.0 MB		
		<input type="checkbox"/>	WAS_server1	92.2 MB	514.9 ms	2.3 ms

Probe Details

Probe: ALMAGRO
Probe Group: Tornado Probes
Group Name: Group dug
Host: almagro.lab.performant.com
Probe Type: Java

- ▶ The tooltip for the columns that contain metric values display the same value that is displayed 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 Details Pane

When you select an entity in the graph entity table, Diagnostics displays the metrics for the selected entity in the Details pane. For more information about the Details pane, see Chapter 4, “Working with the Details Pane.”

Drilling Down On Entities in the Graph Entity Table

Most of the Diagnostics views provide a way for you to drill down to other Diagnostics views to see more detailed performance metrics for an entity selected in the Details pane. The Probe view provides a way for you to open the Diagnostics Profiler for the selected entity. For more information on drilling down on an entity see “Drilling Down to Other Diagnostics Views” on page 86.

Performing Common Tasks for a Selected Entity

Diagnostics provides access to the navigation and configuration tasks that are available for an entity in the table in two ways: by right-clicking on the row in the graph entity table to display the drop-down menu, and using the Common Task area above the Details pane on the right side of the Diagnostics views.

The following image shows the right-click menu that is displayed for a probe entity in the Probe view.

Status	Color	Chart?	Probe	VM Heap U. ▾ 1	Latency	CPU (Avg)
		<input checked="" type="checkbox"/>	ALMAGRO	444.5 MB	10.1 ms	6.0 ms
		<input checked="" type="checkbox"/>	Network		3.9 ms	210.0 ms
		<input checked="" type="checkbox"/>	WL91		37.3 ms	
		<input checked="" type="checkbox"/>	Test5		23.3 ms	4.4 ms
		<input checked="" type="checkbox"/>	MSPe		6.5 ms	3.0 ms
		<input type="checkbox"/>	CallCl			
		<input type="checkbox"/>	x230		520.3 μs	538.8 μs
		<input type="checkbox"/>	Test5		14.8 ms	4.7 ms
		<input type="checkbox"/>	JavaT		61.2 ms	2.1 ms
		<input type="checkbox"/>	WAS		501.8 ms	2.3 ms
		<input type="checkbox"/>	Portal		3.5 ms	0.0 μs
		<input type="checkbox"/>	WL81		859.4 ms	7.8 ms
		<input type="checkbox"/>	T-LC7		92.4 ms	8.2 ms
		<input type="checkbox"/>	t-lc8		340.6 ms	5.1 ms
		<input type="checkbox"/>	t-lc8		339.7 ms	3.3 ms
		<input type="checkbox"/>	snipp		115.1 μs	53.3 μs
		<input type="checkbox"/>	LABKI		113.8 ms	81.9 ms
		<input type="checkbox"/>	orcl			

Task	Latency	CPU (Avg)
View Server Requests	3.9 ms	210.0 ms
View Probe Summary	37.3 ms	
View Portal Components	23.3 ms	4.4 ms
View Load	6.5 ms	3.0 ms
View Trended Methods	520.3 μs	538.8 μs
Create New Incident and Add	14.8 ms	4.7 ms
View Profiler for ALMAGRO	61.2 ms	2.1 ms
Create Alert Rule...	501.8 ms	2.3 ms
View/Edit Comments...	3.5 ms	0.0 μs
Delete Comments	859.4 ms	7.8 ms
Delete ALMAGRO	92.4 ms	8.2 ms

When you select a row in the graph entity table, Diagnostics updates the Common Tasks menu so that it contains the tasks that are appropriate for the selected entity.

These tasks are included in the entity task menus, and are described in the sections that follow:

- “Drilling Down to Other Diagnostics Views” on page 86
- “Creating a New Incident and Adding the Entity” on page 87
- “Drilling Down to the Profiler for a Probe Entity” on page 87
- “Managing Alert Rules for an Entity” on page 88
- “Managing Comments for an Entity” on page 88
- “Deleting an Entity” on page 88

Drilling Down to Other Diagnostics Views

Many of the Diagnostics views allow you to drill down on the entities listed in the graph entity table so that you can analyze their performance in greater detail. As you drill down from one view to the next, the bread crumb at the top of the view keeps track of the path that you have taken so that you can navigate backwards.

Note: Not all views have drill-down navigation options.

There are three ways to drill-down on an entity in the graph entity table:

- Right-click on a row in the table. Diagnostics displays the pop-up menu for the entity listing the available drill-down navigation options.
- Double-click a row in the table to drill down directly into the default lower level detail layout view. This is a shortcut that allows you to drill down without stopping at the entity pop-up menu. The default drill-down option is bolded in the entity pop-up menu.
- Select the drill-down navigation option from the Common Tasks area.

Creating a New Incident and Adding the Entity

Selecting **Create New Incident and Add** from the entity pop-up menu or from the Common Tasks menu causes Diagnostics to create a new incident including the selected entity and all of the metrics that were charted for the entity. Diagnostics switches from Monitoring mode to Incident Analysis mode and displays the Analyze Incidents tab with the newly created incident. For more information on Incident Analysis see Chapter 8, “Performing Incident Analysis.”

Drilling Down to the Profiler for a Probe Entity

Selecting **View Profiler for...** from the entity pop-up menu or from the Common Tasks menu on the Probe view causes Diagnostics to open the Diagnostics profiler for the Probe. If the entity is a J2EE Probe, Diagnostics opens the J2EE Diagnostics Profiler in a new window in the same browser. If the entity is a .NET Probe, Diagnostics opens the .NET Diagnostics Profiler in a new browser.

Managing Alert Rules for an Entity

You can create and edit alert rules for an entity listed in the graph entity table by right-clicking on the entity and selecting one of the alert menu options from the pop-up menu. The options displayed in the menu vary depending on whether alert rules exist or not. The same menu options are also available on the Common Tasks menu. For more information on alert rules see Chapter 7, “Alert Notification.”

Managing Comments for an Entity

You can create and edit comments for an entity listed in the graph entity table by right-clicking on the entity and selecting one of the comment menu options from the pop-up menu. The options displayed in the menu vary depending on whether comments exist or not. The same menu options are also available on the Common Tasks menu.

Deleting an Entity

Diagnostics automatically purges a Probe from the Diagnostics data after three months have passed since data was last received from the Probe. There are times when you may want to remove a Probe or get rid of old Probe metrics sooner than the three month automatic purge. The **Delete <entity>** in the entity pop-up menu in the graph entity table provides a way for you to delete a Probe’s metrics from Diagnostics.

Deleting Old Data for a Probe

If you have redeployed an application or made other changes to its configuration that are likely to significantly alter the applications performance, you may want to purge the old data that the Probe monitoring the application collected so that the old performance metrics do not distort the current application’s performance characteristics.

When you delete a Probe that is actively sending data to Diagnostics, the old data for the Probe is deleted and the Probe disappears from the graph entity table. However, the next time that the Probe sends its Metrics, the Probe will reappear in the graph entity table with only the new metrics that were just received.

Deleting a Deactivated Probe

When you delete a Probe that is no longer sending data to Diagnostics, the old data for the Probe is deleted and the Probe disappears from the graph entity table. You can no longer reference the Probe in Diagnostics.

7

Alert Notification

This chapter describes:	On page:
About Alert Notification	91
Configuring Alert Notification	92
Working with Alert Notification Rules	97
Reviewing Alert Notification Events	103

About Alert Notification

Diagnostics enables you to set threshold values for metrics so that it can issue a threshold alert when the metric values have exceeded the threshold.

Diagnostics indicates that a threshold alert condition has occurred in the following ways:

- ▶ By changing the color of the status indicator to red or yellow when displayed in the Status view, the Alert Rules view, and the graph entity table of the views with the detail layout.
- ▶ By adding a red or yellow border to the cells of the graph entity table and the details pane that contain the metric values that have exceeded their thresholds.

For more information on setting thresholds, see “Setting Metric Thresholds” on page 63.

In addition to the threshold 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 when an entity's status becomes critical. Alert notification rules can be created for when critical status conditions are encountered for probes, probe Groups, hosts, and server requests. Diagnostics can trigger alert notifications when the status of the entity changes to critical and also when the entity becomes unavailable for a period of five minutes.

Note: Alert notifications for a given entity are sent once, at the time that the entity's status becomes critical. 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.

Note: Changes to the Alert Notification configuration only apply to alert notification rules that you create after you have submitted the configuration change. To reconfigure alert rules that already exist, you must edit each alert rule.

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)	
<input type="button" value="Submit"/> <input type="button" value="Reset All"/>			

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 Diagnostics 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 impacts only the alert notifications for entities that are using this 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 graph entity table and details pane 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 93.
- 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 93.
- 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. When you maintain an alert rule, you will not be able to select the SNMP alert notification option until you have configured these 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 93.
- 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 values to take effect.

Configuring SMTP E-mail Alert Notifications

SMTP alert notifications are disabled until you configure the SNMP alert notification properties. When you maintain an alert rule, you will not be able to select the SNMP alert notification option until you have configured these 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 93.
- 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. These values will be used to configure each alert rule that you create.

Note: You should set the SMTP Default E-mail Address to an alias address or e-mail distribution list that you can easily control from your e-mail server. This will save you from having to update each alert rule to a new e-mail address each time the person to receive the alerts changes.

- 3 Click **Submit** to enable SMTP alert notifications. It can take up to 30 seconds for the new property values to take effect.

Turning Alert Notification Messages On and Off in All Views



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 external alert notification is sent.

Note: Active Alert Notifications are turned off by default and they will not be displayed unless an external alert notification has actually been sent.

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.

Understanding the Scope of Alert Rules

The status of lower level entities is propagated up to the higher level entities. For this reason, you do not have to set an alert on each entity where you have set a threshold. A single high-level alert notification rule on a high-level entity can be used to notify you of a variety of conditions in the performance of lower-level entities.

For example, a critical alert notification rule on a probe group is triggered any time a probe or a server request in that probe group enters critical status. This is because the status of the probe group becomes critical when any of its probes or their server requests become critical. Similarly, an alert on a probe can be triggered by the threshold violation of a probe metric; but it can also be triggered when any metric on any of the probe's server requests crosses a threshold. In these cases, the alert notification provides the details for the metric that triggered the alert, even if that metric is actually on a lower-level entity.

Maintaining Alert Notification Rules

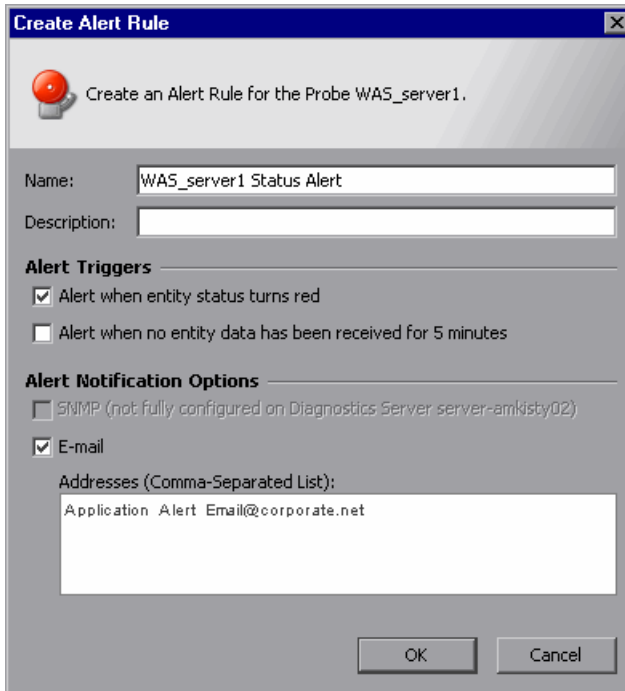
You can create, edit, and delete the alert notification rules using the right-click pop-up menu on the views that list probes, probe groups, hosts, and server requests. Only one alert notification rule can be created for each entity.

Note: You cannot create an enabled alert notification rule until you have configured either the SNMP properties or the e-mail properties as described in “Configuring Alert Notification” on page 92.

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 graph entity table of a view with a detail layout. Diagnostics displays the pop-up menu for the selected entity.
- 2 Open the Create Alert Rule dialog by selecting the **Create Alert Rule** menu option. If an alert rule has already been created for the selected entity, a menu option for **View/Edit Alert Rule** is displayed instead of **Create Alert Rule**.

Diagnostics displays the dialog as shown in the following example:



- 3** 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 want to enter a name such as Prod_010 CPU Check. However, you may want to use a more generic name for the alert rule if there is more than one metric with a threshold value that could trigger a single alert notification. A specific name that includes the name of the metric could be misleading in this case.

- 4** Enter a short **Description** for the rule.

- 5** 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 becomes critical and the indicator is turned red.

An entity's status becomes critical whenever one of its metric thresholds has been exceeded, or when any of the metric thresholds have been exceeded on its sub-entities.

- ▶ 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.

Note: If you do not select at least one Alert Trigger, the alert notification rule is created but is disabled.

- 6** 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 at least one Alert Notification Option, the alert notification rule is created but is disabled.

To maintain alert notifications rules:

- 1** Right-click on a Probe, Probe Group, Host, or Server Request listed in a Status view on the graph entity table of a detail layout 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 with the **Edit Alert Rule** menu option.

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

If an alert rule has not yet been created for the selected entity, a menu option for **Create Alert Rule** is displayed instead of **View/Edit Alert Rule**.

The Edit Alert Rule dialog is the same as the Create Alert Rule dialog. For more information see the instructions for creating alert rules on page 98.

To delete alert notifications:





- 1** Right-click on a Probe, Probe Group, or Host listed in a Status view or the graph entity table of a detail layout 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 with the **Delete Alert Rule** menu option.

- 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 by 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:

Alert Rules				
Entity Name	Entity Type	Alert Name	Last Fired	Info
CallChain.NET	Probe	CallChain.NET Status Alert	11/13/06 15:39:24.000	
foo	Host	balboa.lab.performant.com Status Alert	11/13/06 15:39:34.000	
t-1c10.lab.performant.com	Host	t-1c10.lab.performant.com Status Alert	11/13/06 15:44:54.000	
WAS_server1	Probe	WAS_server1 Status Alert		

In this example, the Info column indicates that there are four alert notification rules that have been defined: two for a hosts and two for probes. All of the alert notification rules are currently enabled. In addition, the example shows that one of the entities has comments associated with it.

Understanding the Columns in the Alert Rules View

The columns in the Alert Rules view provide the same information and navigation options as their counterparts in the graph entity table of the views with a detail layout.

When you right-click on the rows in the Alert Rules view, Diagnostics displays the same pop-up menu that appears for the listed entity in the views with a detail layout.

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 layout 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. If an alert notification has been issued for a listed alert rule the pop-up menu for the entity contains the **View Threshold Violation** option.

Note: The Last Fired column contains 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 indicates that the entity has alert rules or comments. One or more of the following icons can appear in this column:



- ▶ **Active Alert Rule.** Indicates that an alert notification rule has been created for the selected entity and the rule is active.



- ▶ **Disabled Alert Rule.** Indicates that an alert notification rule has been created for the selected entity but the rule is currently disabled.



- ▶ **Custom Comments.** Indicates that comments have been created for the selected entity.

Note: If an alert notification rule has fired, you can double-click the row representing the alert rule to be taken to a view that displays the threshold violation.

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, the table of events in this view 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 graph entity table of the views with a detail layout.

The following is an example of the Alert Events view.

Alert Events						
Entity Name	Alert Name	Timestamp	Violating Metric	Threshold	Value	Status
t-lc10.lab.perform...	t-lc10.lab.perform...	11/13/06 15:44:5...	CPU (Avg)	31 %	31.6 %	
foo	balboa.lab.perform...	11/13/06 15:39:3...	CPU (Avg)	90 %	90.1 %	
CallChain.NET	CallChain.NET Stat...	11/13/06 15:39:2...	VM Heap Used (Avg)	100 B	76.3 MB	

6.5.19.611

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 layout 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.
- **Violating Metric:** This column contains the name of the metric that violated its threshold and triggered the alert event.
- **Threshold:** This column contains the threshold value for the metric 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.** This indicator shows how the listed entity or any of its child entities is performing relative to the thresholds that you set for their metrics.

For example, a probe can have a critical status even when none of the probe's own metrics have exceeded their thresholds. The probe's status will become critical when any of the server requests for that probe become critical.

The color of the indicator stands for the severity of the threshold violation. If you hold the mouse pointer over the status indicator in a row of the table, Diagnostics displays a tooltip with a description of the current status:

Status Indicator Color	Description
Green	Good - The entity 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 entity 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 entity 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.

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 layout 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 layout view that is appropriate for the entity type depicted in the selected row. The detail layout view contains the selected entity in the graph entity table, and the graph contains the charted metrics that triggered the alert.

You can also navigate to the threshold violation by double-clicking on an alert event row.

8

Performing Incident Analysis

This chapter describes how you can use Monitor and Investigate and Analyze Incidents tabs to analyze captured performance incidents.

This chapter describes:	On page:
About Incident Analysis	108
Monitoring Performance Versus Analyzing Incidents	109
Create New Incidents	110
Renaming an Incident	111
Deleting an Incident	112
Using the Analyze Incidents View	112

About Incident Analysis

Diagnostics can capture the performance of your application when you see a performance anomaly and store the information as an *incident*. An incident includes all of the performance metrics that you can see in the Diagnostics views, but it is limited to just the time period for which the incident was captured. With the information captured in an incident, you can investigate threshold violations and other performance issues using the Analyze Incidents view and the other Diagnostics views. Incidents provide you with the metrics for the specific slice of time when the performance of your application was not as you expected, so that you can understand what is causing the performance issue.

You control when Diagnostics captures an incident. When you see a performance anomaly that you want to analyze you can use the controls in the Diagnostics views to create an incident that includes the performance metrics for the specified time period.

Once an incident has been captured, you can ask Diagnostics to display the incident in the Diagnostics views and the Analyze Incidents view. When you are working with an incident, you can modify the incident to control the entities and metrics are included in the Analyze Incidents view so that you can visually compare and correlate the metrics from multiple entities of different types that provide the insight into the performance issue needed to determine an appropriate response.

Monitoring Performance Versus Analyzing Incidents

The **Monitor and Investigate** tab and the **Analyze Incidents** tab behave differently depending on your selection from the **Working On** drop-down.

Monitoring Mode

When you select Monitoring from the **Working On** drop-down, Diagnostics is in *Monitoring mode* and the views displayed in the **Monitor and Investigate** tab contain the real-time performance metrics for the time period specified in the **Viewing Time** filter. The **Analyze Incidents** tab contains instructions for creating a new incident

Incident Analysis Mode

When you select an incident from the **Working On** drop-down, Diagnostics is in *Incident Analysis mode*. As soon as you select an incident, Diagnostics opens the Analyze Incidents view in the **Analyze Incidents** tab for the selected incident. The view is displayed with the entities and metrics that you specified shown in the tables and graph.

The controls in the views on the Monitor and Investigate tab and the Analyze Incident tab for the most part work the same in Incident Analysis mode as they do in Monitoring mode. You should be aware of the following views and controls:

- The **Viewing Time** filter contains an additional option, **Use Time of Incident** that allows you to return to the incident time after you have changed the viewing time using the **Viewing Time** filter, the **Pan**, **Pause** and **Play** buttons, or by zooming.



- ▶ When you are in Incident Analysis mode you can use the **Pan**, **Pause** and **Play** buttons to display performance metrics for periods of time that precede or follow the metrics that are charted in the graph. When you click **Play**, Diagnostics stays in Incident Analysis mode, however it begins to show the current values for the performance metrics charted in the graph and listed in the tables of the views. You can return to the time for the captured incident by clicking the **Use Time of Incident...** option in the **Viewing Time** filter.
- ▶ The Status view and the Alert Events view always show the most up to date information for the performance of your applications no matter if you are in Incident Analysis mode or in Monitoring mode. These views are not impacted by your choices in the **Viewing Time** filter and the **Working On** drop-down.

Create New Incidents

When you see a performance issue while you are monitoring your application's performance in Diagnostics, you can trigger an incident capture using the controls of the graph entity table and the details pane. When Diagnostics creates a new incident it creates a default incident name based on the starting date and time of the period for which the incident was captured. Diagnostics also sets up the Analyze Incident view to display the entities and metrics that are appropriate for the trigger that caused the incident to be captured.

To create a new incident from the graph entity table:

- 1 From the graph entity table, select the row for the entity that you want to include in the Analyze Incidents view of the new incident.
- 2 Select the **Create New Incident and Add** menu option from either the right-click pop-up menu for the entity or the Common Tasks menu.

Diagnostics creates a new incident for the time period displayed in the view. The new incident is automatically selected from the **Working On** drop-down as Diagnostics switches to Incident Analysis mode. The Analyze Incidents view is displayed with the selected entity listed in the graph entity-metric table and the same metrics charted in the graph as were charted on the view where you captured the incident.

To capture a new incident from the details pane:

- 1** From the details pane, select the row for the metric that you want to include in the Analyze Incidents view of the new incident.
- 2** Click **Incident Analysis** for the selected row or select the **Create New Incident and Add** menu option from the right-click pop-up menu for the metric.



Diagnostics creates a new incident for the time period displayed in the view. The new incident is automatically selected from the **Working On** drop-down as Diagnostics switches to Incident Analysis mode. The Analyze Incidents view is displayed with the metric you selected charted in the graph and the entity to which the metric belongs listed in the graph entity-metric table.

Renaming an Incident

You can rename an incident listed in the **Working On** drop-down to make it easier to locate in the drop-down or easier to recognize the performance issue that the incident represents.

To rename an incident:

- 1** Select the incident to be renamed from the **Working On** drop-down.

Diagnostics switches to Incident Analysis mode and displays the metrics for the selected incident in the tab that was open when you made your selection.

- 2** Click **Rename Incident** to the right of the **Working On** drop-down.

Diagnostics displays the Rename Incident dialog with the current name of the selected incident.

- 3** Enter the new incident name.
- 4** Click **OK** to rename the incident.

Deleting an Incident

You can delete an incident listed in the **Working On** drop-down so that an incident that you have already resolved or an incident that you are no longer interested in does not clutter the list in the drop-down.

To delete an incident:

- 1 Select the incident to be renamed from the **Working On** drop-down.

Diagnostics switches to Incident Analysis mode and displays the metrics for the selected incident in the tab that was open when you made your selection.

- 2 Click **Delete Incident** to the right of the **Working On** drop-down.

Diagnostics deletes the selected incident so that it will not longer be listed in the **Working On** drop-down.

Using the Analyze Incidents View

This section includes:

- “About the Analyze Incidents View” on page 113
- “Description of the Analyze Incidents View” on page 114
- “Accessing the Analyze Incidents View” on page 115
- “Customizing the Analyze Incidents View” on page 116
- “Working With Entity-Metric Pairs” on page 116
- “Maintaining Incident Notes” on page 119

About the Analyze Incidents View

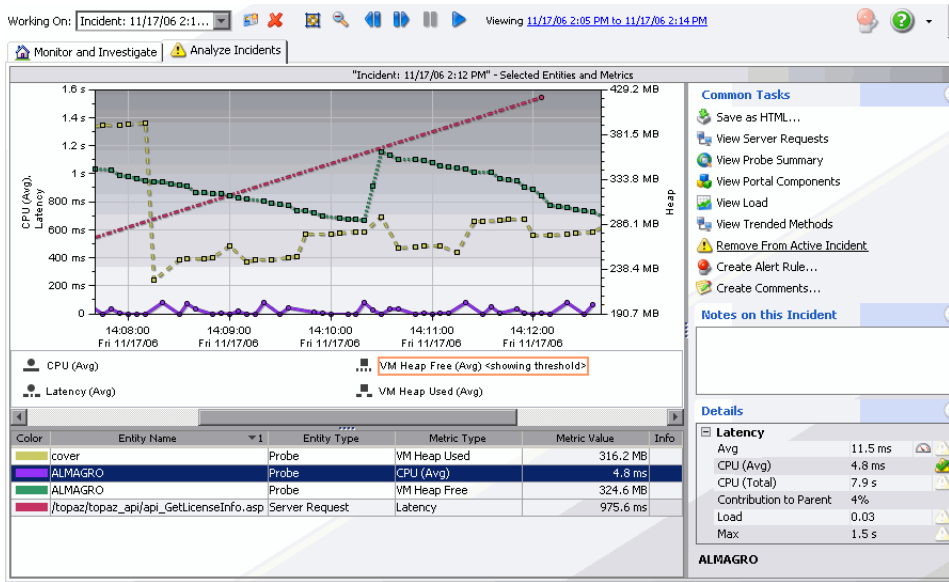
The Analyze Incidents View on the **Analyze Incidents** tab displays the entities and metrics from a captured incident that you have selected to be included in this view. This view provides a way for you to compare, contrast, and correlate entities and metrics of different types that would normally only be shown in separate views in the graph and graph entity table of one view.

Note: The Analyze Incidents view is only displayed in the **Analyze Incidents** tab when Diagnostics is in Incident Analysis mode. When Diagnostics is in Monitoring mode, the **Analyze Incidents** tab contains instructions for creating a new incident.

The Analyze Incidents view has the layout of a Diagnostics detail view. For information about the layout and controls in the views with the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

Description of the Analyze Incidents View

The following image is an example of the Analyze Incidents view:



Graph

By default, the graph in Analyze Incidents contains the charted metrics for the entity-metric pairs listed in the graph entity-metric table. These metrics are the metrics that you have included in the incident.

Graph Entity-Metric Table

The graph entity-metric table lists each entity-metric pair that you selected to be included in the Analyze Incident view. Each entity included in the view will be listed once for each metric that has also been included in the view.

Notes

The **Notes on this Incident** area allows you to record information about the reason that you created the incident and the steps that you have taken and the progress that you have made in understanding the cause of the performance captured in the incident.

Details Pane

The details pane in the Analyze Incidents view lists the metrics for the selected row of the graph entity-metric table.

Accessing the Analyze Incidents View

To access the Analyze Incidents view, Diagnostics must be in Incident Analysis mode. Diagnostics is in Incident Analysis mode when you select an incident from the Working On drop-down or when you create a new incident from a view while in Monitoring mode. The Analyze Incidents view is displayed on the Analyze Incidents tab when Diagnostics is in Incident Analysis mode. For more information on creating a new incident see “Monitoring Performance Verses Analyzing Incidents” on page 109.

If you open the Analyze Incidents tab when Diagnostics is in Monitoring mode, the Analyze Incidents view is not displayed. Instead, a text message is displayed with instructions for creating a new incident.

To access the Analyze Incidents view using the Working On drop-down:

Select an incident listed in the **Working On** drop-down.

- If you were looking at the views in the Monitor and Investigate tab, Diagnostics continues to display the same view in that tab. However, the time range displayed in the tab changes to that of the selected incident. You can click the Analyze Incidents tab at any time to see the Analyze Incidents view for the selected incident.
- If you were looking at the Analyze Incidents view Diagnostics continues to display the same view. However, the performance metrics displayed in the tab change to those that were selected for the view for the selected incident.

Customizing the Analyze Incidents View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Analyze Incidents view. For more information about the ways you can control how performance metrics are presented in this view, see “Viewing Diagnostics Data in Detail Layout” on page 41.

Working With Entity-Metric Pairs

As you are investigating and analyzing an incident you will discover entities and metrics that help shed some light on the application’s behavior which you will want to see displayed in the Analyze Incidents view. You will also find that some of the entities and metrics that are shown in the view are no longer helpful and should no longer be included. Diagnostics provides the following ways for you to manipulate the entities and metrics that are included in the Analyze Incidents view.

Adding a Metric to the Analyze Incidents View

When Diagnostics is in Incident Analysis mode, you can chose additional metrics to be charted in the graph of the Incidents Analysis view from the views in the Monitoring and Investigate tab or from the Analyze Incidents view.

To add a metric to the Analyze Incidents view:

- 1** Select an incident from the **Working On** drop-down so that Diagnostics is in Incident Analysis mode.
- 2** From the details pane in either the Analyze Incidents view or a view displayed in the Monitor and Investigate tab, select the row for the metric that you want to include in the Analyze Incidents view of the active incident.
- 3** Click **Incident Analysis** for the selected row or select the **Add to Active Incident** menu option from the right-click pop-up menu for the metric.



Diagnostics displays the Analyze Incidents view. The selected metric is charted in the graph and an entry for the selected entity-metric pair is listed in the graph entity-metric table. In the details pane the **Incident Analysis** button for the metric is changed to indicate that the metric is included in the view.



Adding an Entity to the Analyze Incidents View

When Diagnostics is in Incident Analysis mode, you can chose additional entities to be included in the Incidents Analysis view. When you include an entity from a Diagnostics view, each of the metrics for the entity that were charted, are also charted in the graph of the Incidents Analysis view. You can select an entity to add to the Incidents Analysis view from the views displayed on the Monitoring and Investigate tab.

To add a metric to the Analyze Incidents view:

- 1** Select an incident from the **Working On** drop-down so that Diagnostics is in Incident Analysis mode.
- 2** In the Monitor and Investigate tab, navigate to the view that contains the target entity.
- 3** From the graph entity table, select the row for the entity that you want to include in the Analyze Incidents view for the active incident.
- 4** Select the **Add to Active Incident** menu option from either the right-click pop-up menu for the entity or the Common Tasks menu.

Diagnostics displays the Analyze Incidents view. The selected entity is included in the view so that each of the metrics that were charted for the view on the Monitor and Investigate tab are also charted in the graph and an entry for each entity-metric pair is listed in the graph entity-metric table. In the details pane the **Incident Analysis** button for each of the charted metrics is changed to indicate that the metric is included in the view.



Deleting a Metric from the Analyze Incidents View

When Diagnostics is in Incident Analysis mode, you can remove metrics from the Incidents Analysis view so that they are no longer charted in the graph or listed as an entity-metric pair in the graph entity-metric table. You can delete a metric from the views in the Monitoring and Investigate tab or from the Analyze Incidents view.

Note: When deleting metrics from the view, the entity will no longer be included in the view when the last metric for the entity has been deleted. Entity-metric pairs are listed in the graph entity-metric table in the Analyze Incidents view. The metrics that are listed in the Details table are the same for each entity-metric pair for the same entity. When you delete the last metric for a given entity in the view. The entity is no longer listed in the view.

To delete a metric from the Analyze Incidents view:

- 1** Select an incident from the **Working On** drop-down so that Diagnostics is in Incident Analysis mode.
- 2** From the details pane in either the Analyze Incidents view or a view displayed in the Monitor and Investigate tab, select the row for the metric that you want to remove from the Analyze Incidents view of the active incident.
- 3** Click **Incident Analysis** on the selected row or select the **Remove from Active Incident** menu option from the right-click pop-up menu for the metric.



Diagnostics continues to display the view with which you were working. In the Analyze Incident view, the selected metric is no longer charted in the graph and the entry for the selected entity-metric pair is deleted from the graph entity-metric table. In the details pane the **Incident Analysis** button for the metric is changed to indicate that the metric is no longer in the view.



Deleting an Entity from the Analyze Incidents View

When Diagnostics is in Incident Analysis mode, you can remove entities so that they are no longer included in the Incidents Analysis view. When you remove an entity from a Diagnostics view, each of the metrics for the entity that were charted, are also removed from the Incidents Analysis view. You can select an entity to remove from the Incidents Analysis view from the views displayed on the Monitoring and Investigate tab and from the Incident Analysis view.

- 1** Select an incident from the **Working On** drop-down so that Diagnostics is in Incident Analysis mode.
- 2** In the Monitor and Investigate tab, navigate to the view that contains the target entity. You do not need to do this if you are in the Analyze Incidents view.
- 3** From the graph entity table, select the row for the entity that you want to remove from the Analyze Incidents view for the active incident.
- 4** Select the **Remove From Active Incident** menu option from either the right-click pop-up menu for the entity or the Common Tasks menu.

Diagnostics displays the Analyze Incidents view. The selected entity has been removed from the view along with each of the metrics that were charted for the view.

Maintaining Incident Notes

You can enter notes and comments about an incident in the Notes on this Incident pain located above the Details pain on the Analyze Incidents view. Any text that you enter here is saved when you leave the view, and is redisplayed the next time that you work with the incident in the Analyze Incidents view.

9

Instance Trees

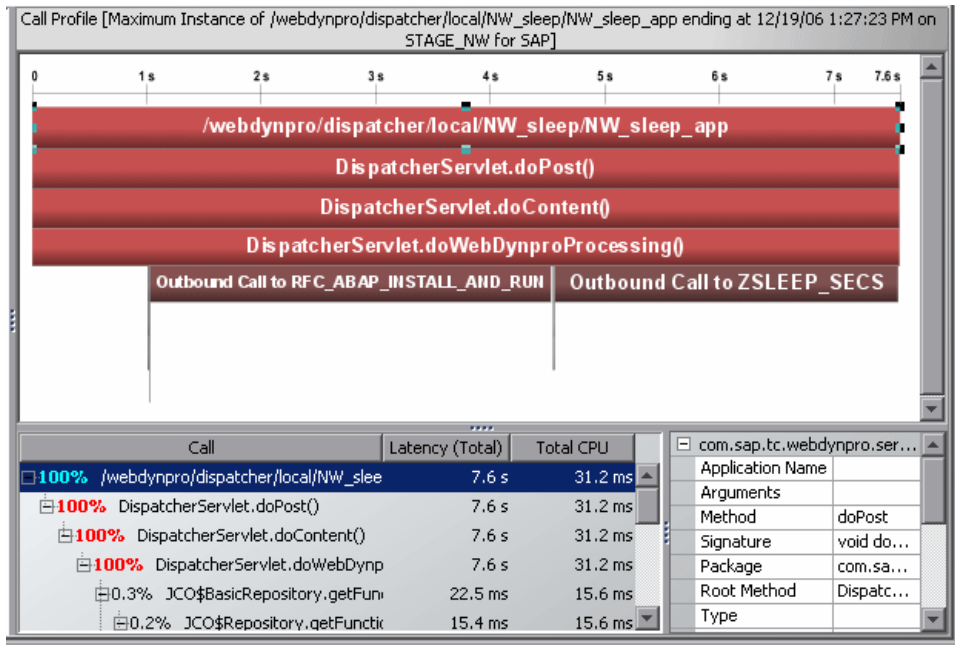
This chapter provides a detailed description about instance trees and how they are used.

This chapter describes:	On page:
Introducing Instance Trees	122
Solving Problems with Instance Trees	124
Cross-VM Trees	128
When Instance Trees Are Not Sufficient	130
Aggregate Trees	131

Introducing Instance Trees

Mercury Diagnostic probes work by adding instrumentation before and after important methods in your application to provide a simplified program trace that includes elapsed times for performance optimization and problem diagnosis.

The program trace is presented in a call profile visualization as shown in the following example:



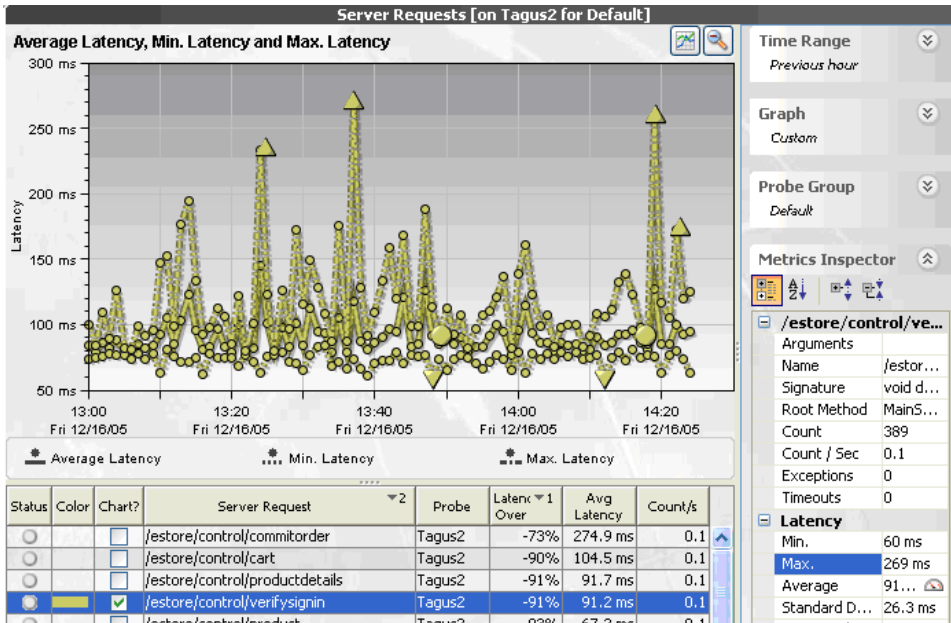
Diagnostics does not record every captured trace because of storage considerations. Instead, at 5-minute intervals, Diagnostics uses an algorithm to select the most interesting instance trees for each server request. The selected instance trees are stored to disk and the trees that were not selected 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
- ▶ the *median* instance tree (in other words, a tree for a server request invocation that took an average amount of time). This is actually only a very close approximation of the actual median instance.
- ▶ a *randomly selected end-to-end complete trace* of the entire cross-VM tree when the server request makes external calls (across two or more virtual machines) such as RMI. The cross-VM tree, includes the trees of the remote system if the remote system was also instrumented.

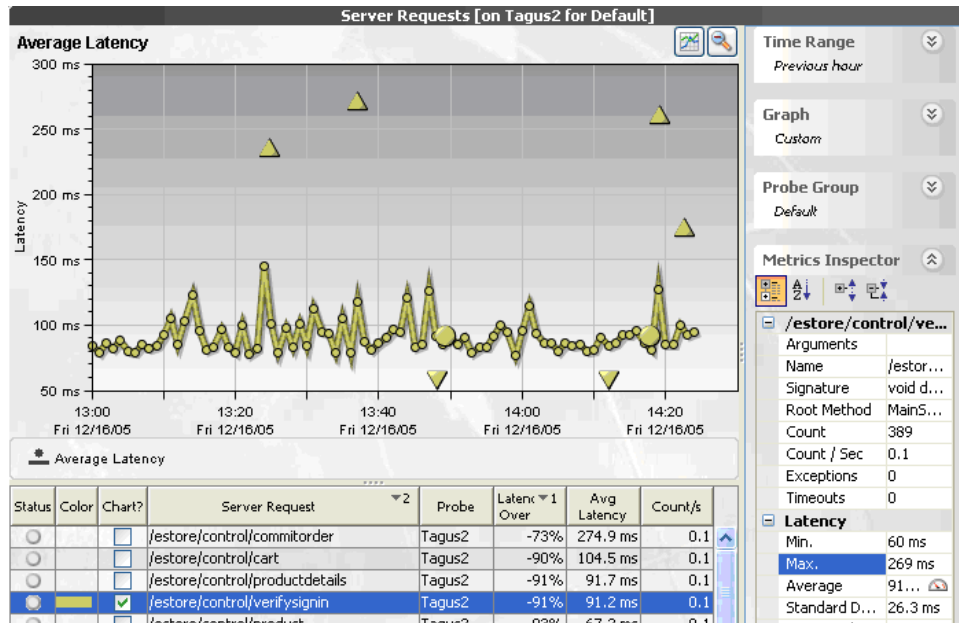
Solving Problems with Instance Trees

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:






The graph shows 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. It shows the charted trend for the Average Latency without trend lines for the minimum and maximum latency:

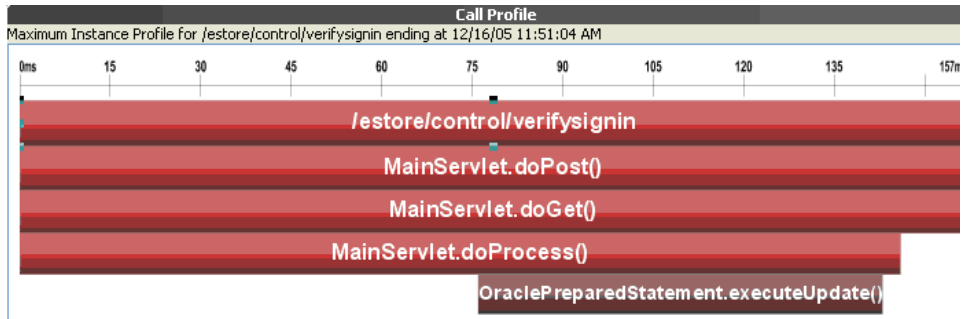


The trend line for this metric includes see small icons to indicate where the fastest, slowest, and median instance trees were recorded:

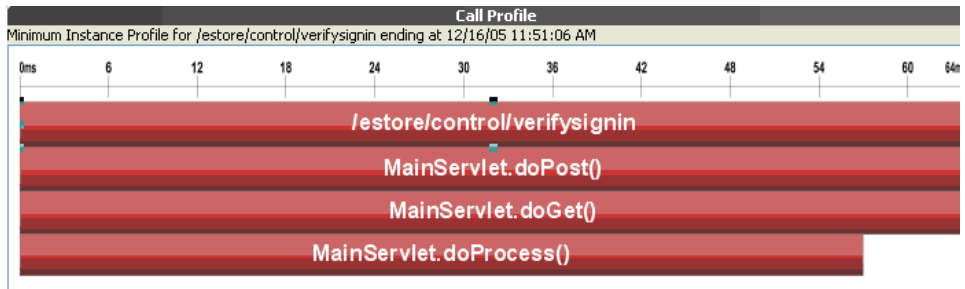
	A median instance tree
	A slow instance tree
	A fast instance tree

To understand why `/estore/control/verifysignin` is sometimes 3 times slower than average (or 5 times slower than the best case), compare the captured instance trees against one another to see what is different about each case:

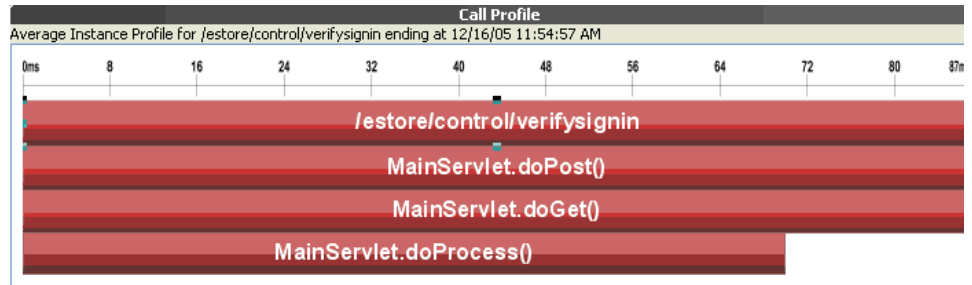
The maximum (slowest) instance tree call profile looks as follows:



The minimum (or fastest) instance tree call profile looks as follows:



The average (or median) instance tree call profile looks as follows:



Comparing these three call profiles, it becomes apparent that the database method (**OraclePreparedStatement.executeUpdate**) is slowing down the server request in the slowest instance. Note that there is not much difference between the fastest and average calls in terms of application behavior.

Of course, 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:

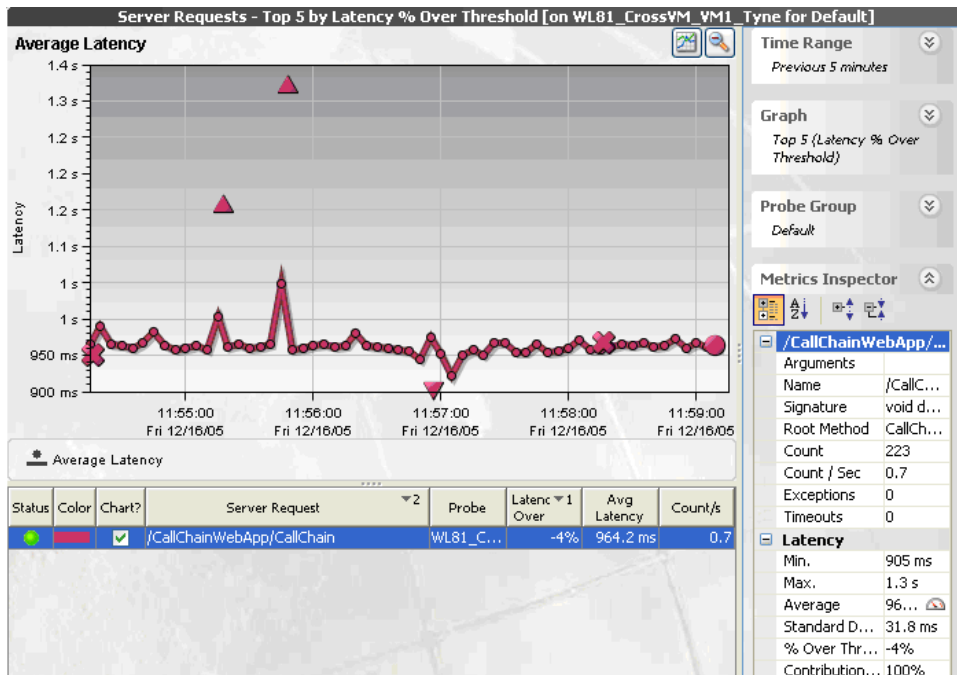
- ▶ The EJB is cached in the application server memory and the retrieval is nearly instantaneous. This would be the minimum call profile.
- ▶ The EJB must be retrieved from the database. Depending upon usage statistics, this could be the average call profile.
- ▶ The server request must wait a significant amount of time for a pooled database connection prior to requesting the EJB data. This could be the maximum call 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 x-icon, indicates where a Cross-VM call tree was recorded. The color of the icon is the same color as the server request trend line with which it is charted. When you click the icon, a complete cross-system call profile opens:



The red nodes in the profile are the precessing that took place on the origination system, and the blue nodes are the processing that took place on the child system, tyne.lab.performant.com.

This call tree indicates that over half of the elapsed time for this server request was caused by processing in the child system.

When Instance Trees Are Not Sufficient

For some applications, three types of instance trees do not provide sufficient information to make it possible to distinguish between 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 each with completely different performance behaviors.

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, maximum, minimum, and average trees for **/controller** are insufficient to diagnose application problems—the call profile for each action processed could be significantly different.

For cases like this you should customize the probe instrumentation for the **/controller** servlet slightly so that the probe can pick up the equivalent of the 'action' parameter as part of the Diagnostics server request. This requires a minor change to the instrumentation that Mercury Customer Support can help you make.

With this customized instrumentation, you would be able to 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

Early versions of Diagnostics used aggregate trees instead of instance trees. In fact, the Mercury Diagnostics Profiler still supports both types of trees. The following are 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 could show a cache being both hit and missed when in reality, for any particular invocation, only one of the two could have occurred. This can be confusing and even lead to misunderstandings when reports are passed on to those who do not understand Diagnostics.

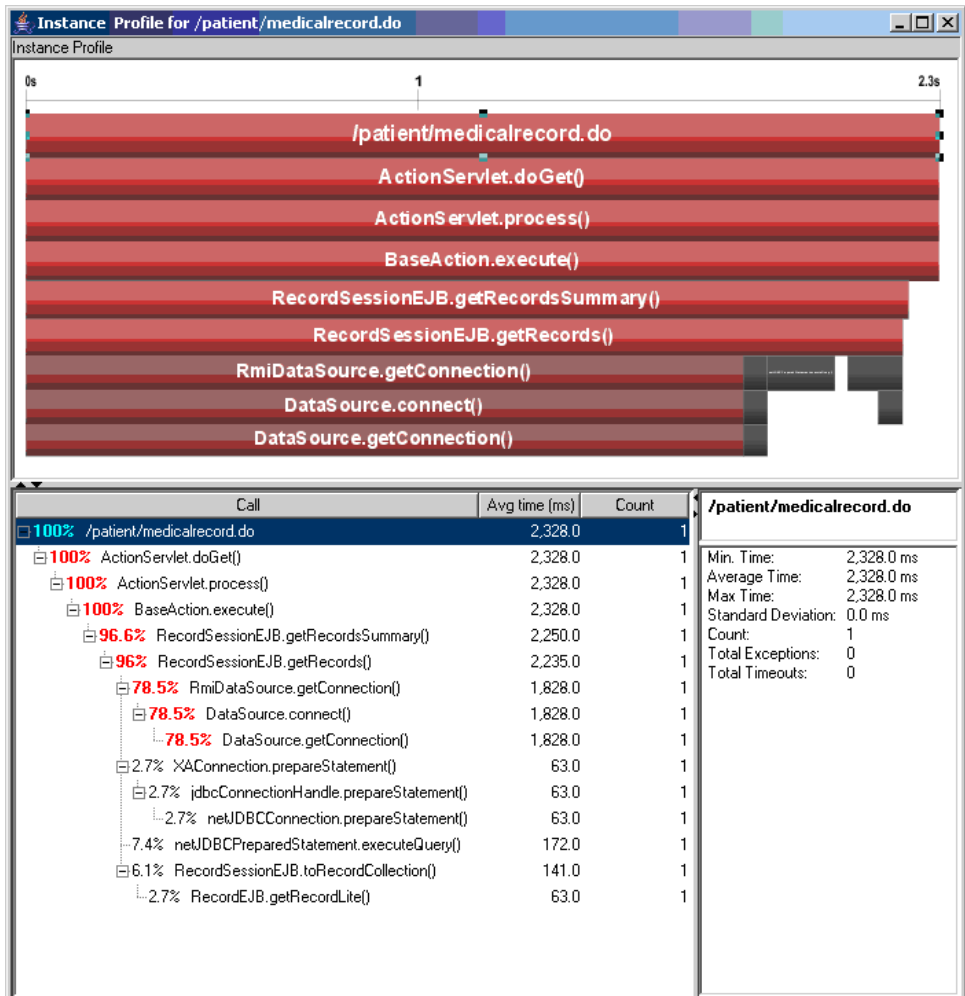
Another disadvantage is that while aggregate trees do well for general performance tuning (such as what occurs during a load test), they tend to hide information about relatively infrequent slow cases. Unfortunately, it is often these cases that cause problems for the customer.

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.

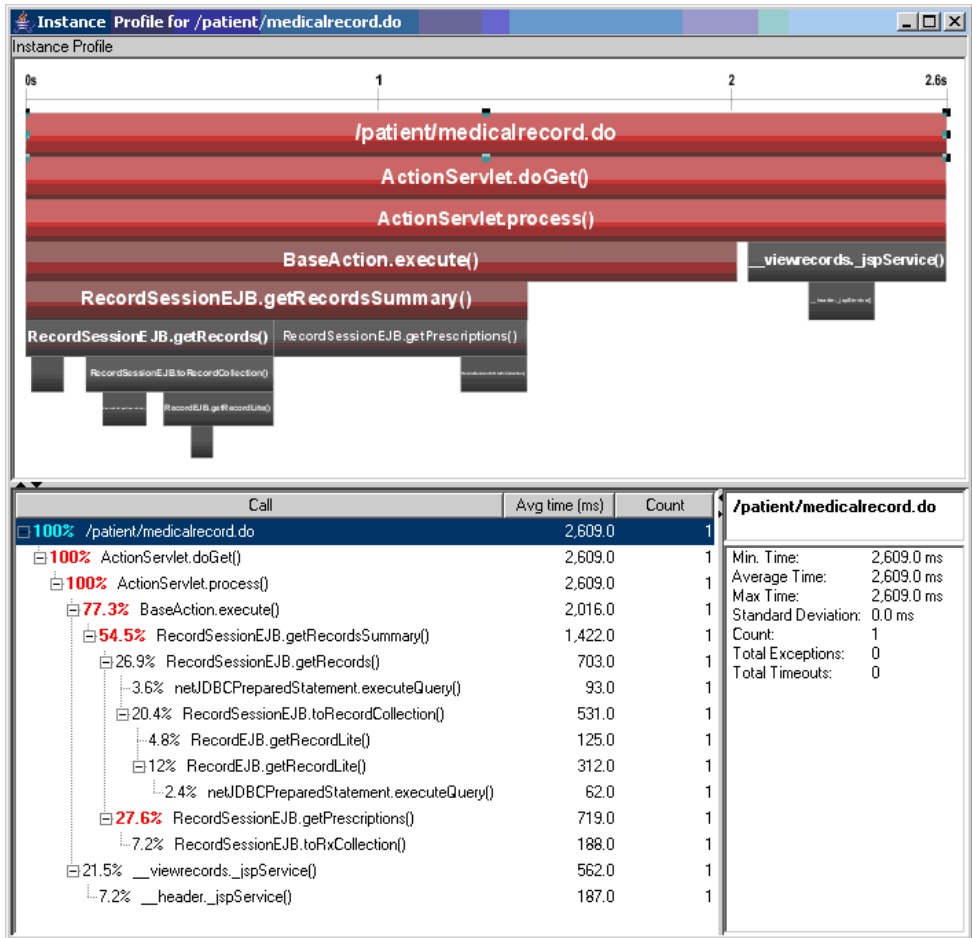
Example Instance Tree #1

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?)



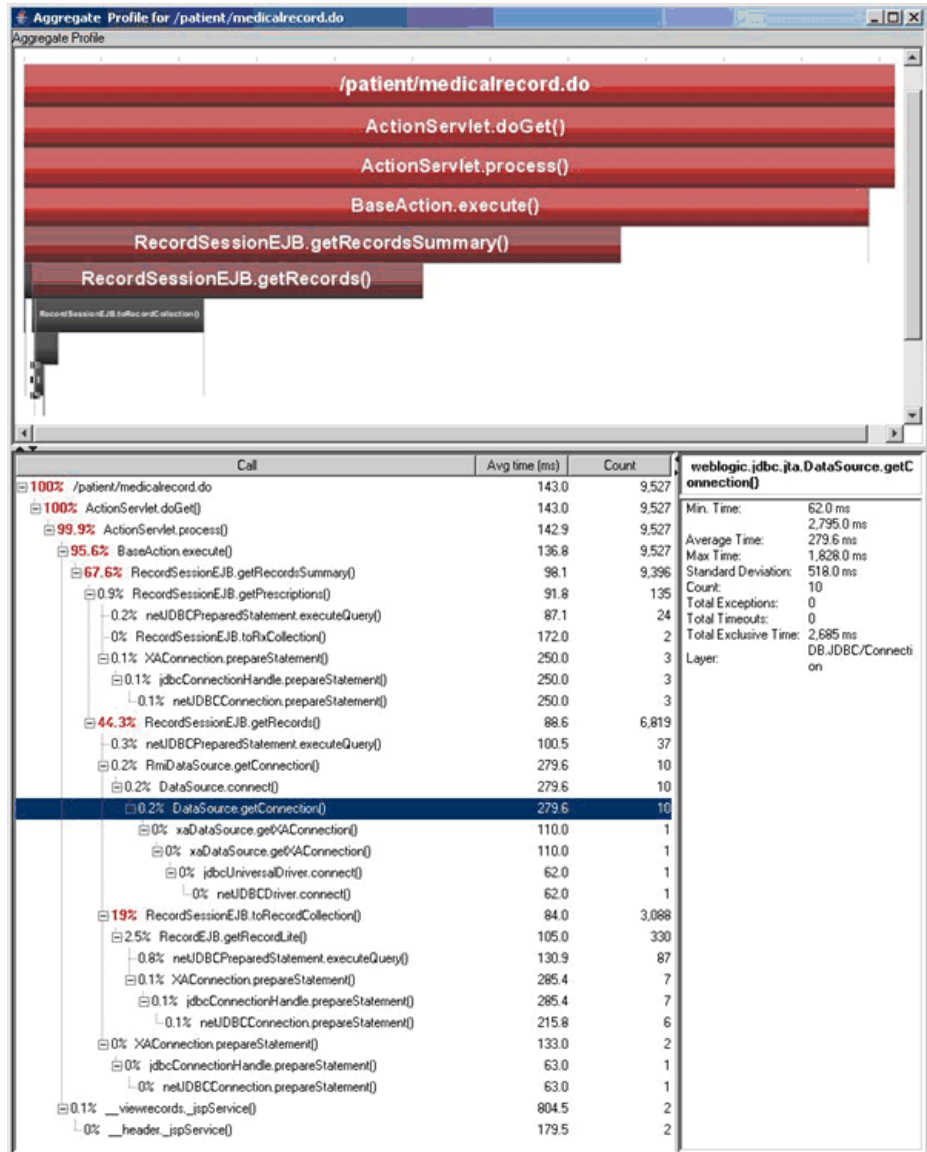
Example Instance Tree #2

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 for the same server request:



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**).

10

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	138
Understanding Diagnostics Custom View Retention	138
Creating View Groups	139
Hiding View Groups	140
Saving a Customized View	142
Creating a New View	144
Renaming a View	148
Deleting a View	149
Modifying a Custom View	150
Sharing Custom Views	150
Upgrading Custom Views From Previous Diagnostics Versions	151

About Mercury Diagnostics Custom Views

Diagnostics lets 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 Part II, “Working with 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 Diagnostics 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 you access it.

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

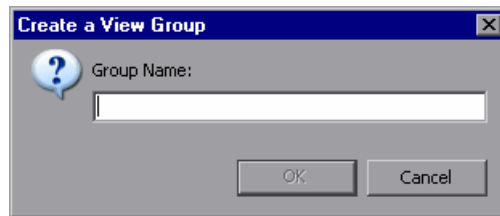
Custom views must be saved in a custom view group, such as **My Views**, which is installed with the product, or in a custom view group that you created. Custom views cannot be stored in any of the view groups installed with the product except for **My Views**.

To create a view group:

- 1 Right-click the View bar inside any view group, and select **Create a View Group** from the pop-up menu.

Note: Be sure to hold the pointer over an open space on 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**.

Diagnostics creates the new view group and opens it in the View bar.

Hiding View Groups

If there are View Groups listed in the View bar that you do not wish to see in the list any longer you may hide them so that Diagnostics will not display them in the View bar. You may unhide View groups that have been hidden so that they are once more available in the View bar.

Note: You may only hide groups that do not contain any views. To hide a group with views, first delete the views.

To hide a view group:

- 1 Select the View group that you want to hide from the View bar.
- 2 Right-click the View bar inside of the selected view group, and select **Hide this View Group** from the pop-up menu.

Note: Be sure to hold the pointer over an open space on the View bar, and not over the icon for a view.

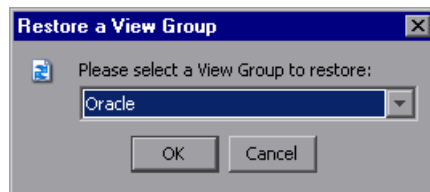
Diagnostics hides the view group so that it is no longer visible in the view group displayed in the View bar and opens the View group that preceded the view group that you hid.

To unhide a view group:

- 1 Right-click the View bar inside of any view group, and select **Restore a View Group** from the pop-up menu.

Note: Be sure to hold the pointer over an open space on the View bar, and not over the icon for a view.

Diagnostics displays the Restore a View Group dialog as shown in the following image:



- 2 Select the view group that you want to unhide and click **OK**

Diagnostics unhides the selected view group and displays it at the end of the list of view groups in the view bar.

Saving a Customized View

As you work with a view, you may find that your customization to 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 into the My Views view group or into a view group that you have created. 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.

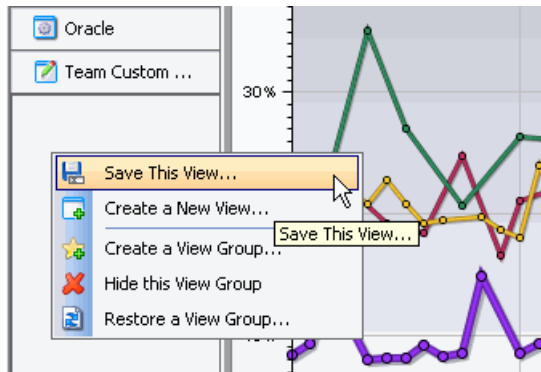
To save a view in a custom view group:

- 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.

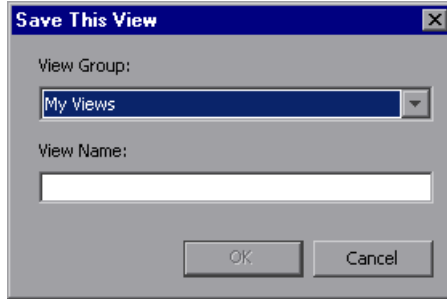


- 2 Right-click the View bar inside the custom view group, and select **Save This View**, or click the **Save This View** button in the view toolbar.



Note: Be sure to hold the pointer over an empty area in the View bar, and not over the icon for a view.

Diagnostics displays the **Save This View** dialog.



- 3 Select a view group from the list of view groups that can contain custom views.
- 4 Type a name for the **View Name**, and click **OK**.

The view that is currently displayed is saved as a custom view, 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.

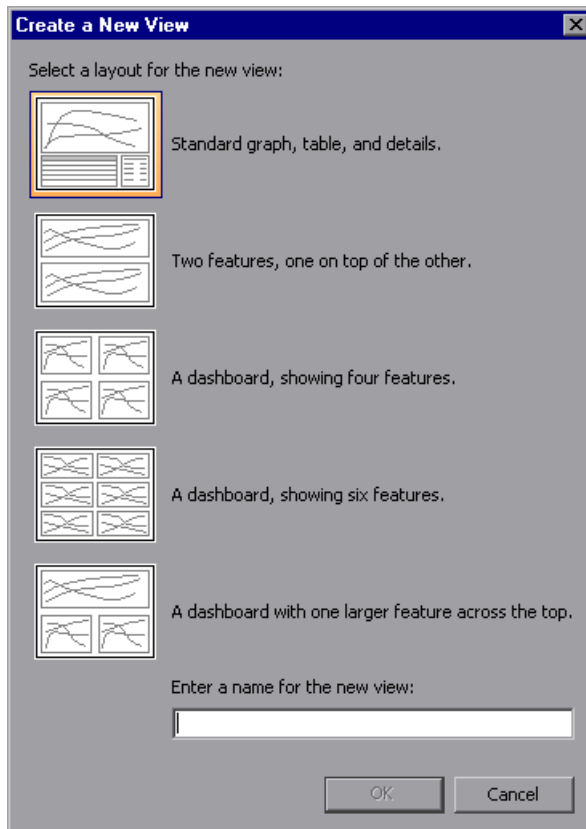
To create a view:

- 1 Select the custom view group where you want to store the new view from the view bar .

Note: Custom views can only be stored in the My Views view group, or in a view group that you have created.

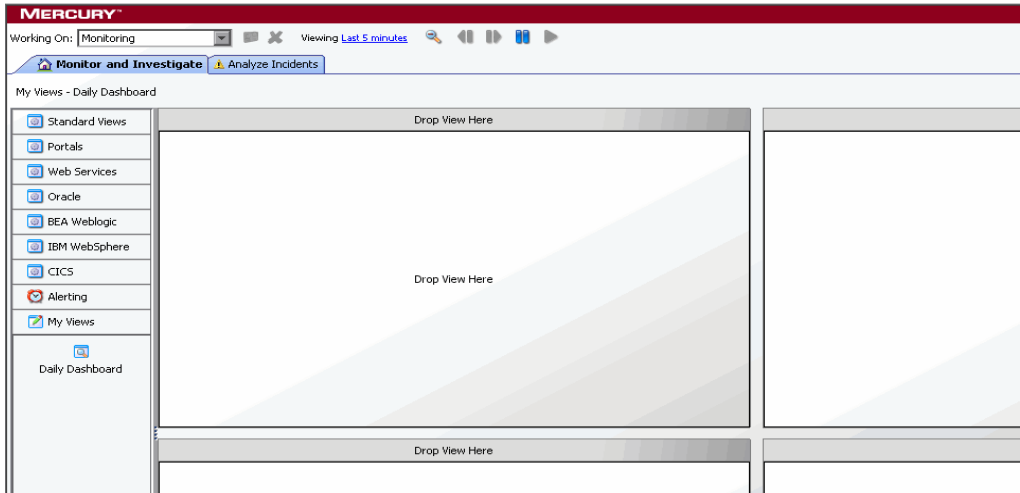
- 2 Right-click the View bar inside the custom view group, and select **Create a New View** from the pop-up menu.

Diagnostics opens the **Create a New View** dialog box opens.



- 3 Select the icon for the layout of the new view. The selected icon is highlighted. Enter a name for the new view, and click **OK**.

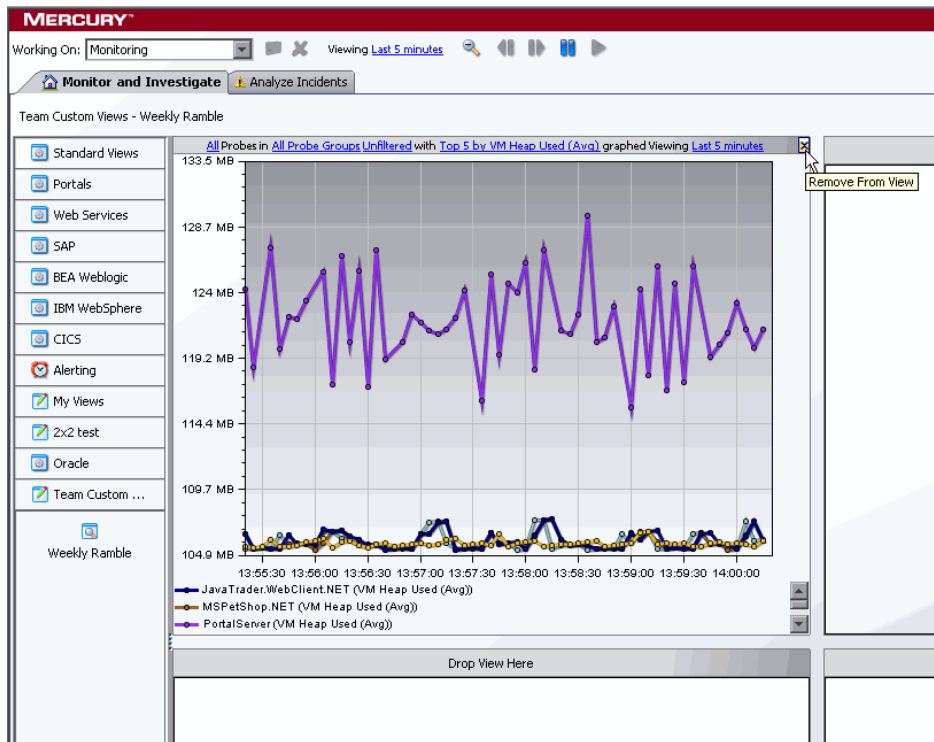
An icon for the new view is added to the view group, and the selected view layout is displayed on your screen with **Drop View Here** placeholders for each of the views that make up the new view. The example below shows the view layout for the four-feature dashboard.



4 Select the views that you want to appear in the new custom view.

- 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

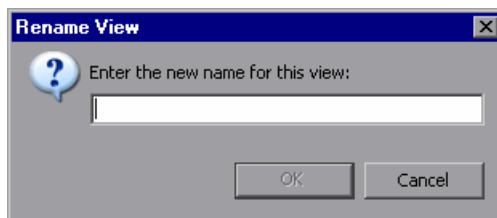
You can rename a view that you have saved in a custom view group.

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 Select 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** from the pop-up menu.

The Rename View dialog box opens.



- 3 Enter a new name for the view, and click **OK**.

Diagnostics renames the view and updates the view bar to display the new name.

Deleting a View

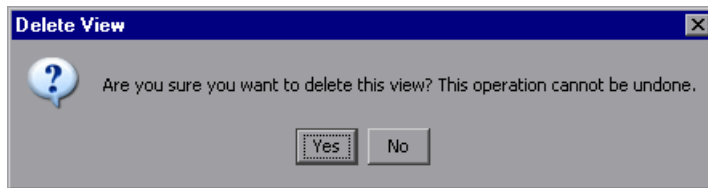
You can delete a view that you have saved in a custom view group.

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 Select 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** from the pop-up menu.

Diagnostics displays the Delete View verification dialog.



- 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.

Upgrading Custom Views From Previous Diagnostics Versions

When you open the custom views that were created in Diagnostics 4.0, 4.1 or 4.2 for the first time in Diagnostics 6.5, 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, 4.1, or 4.2 to 6.5, 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 150.

Part III

Understanding Diagnostics Standard Views

11

Hosts View

This chapter explains how to work with the Diagnostics Hosts view.

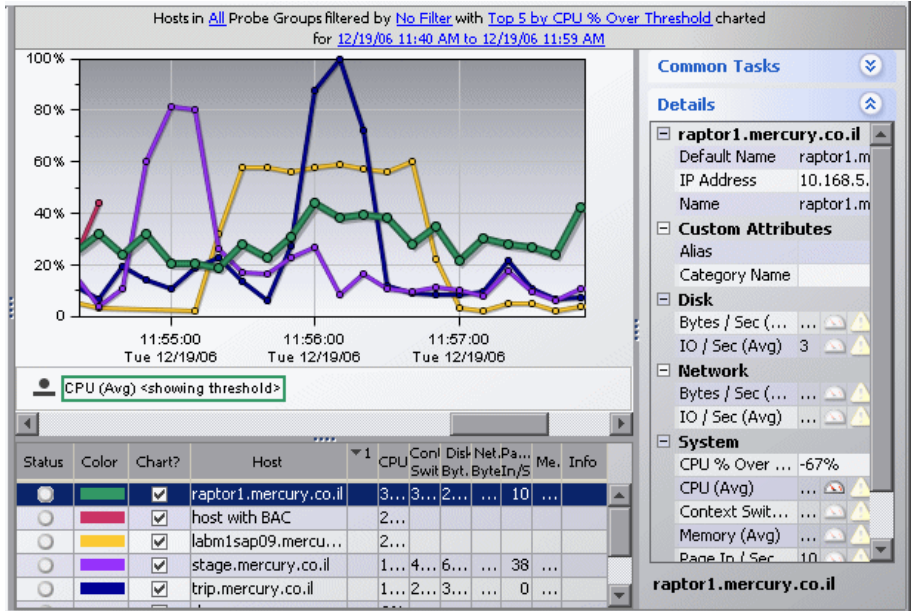
This chapter describes:	On page:
Using the Hosts View	156
Description of the Hosts View	156
Accessing the Hosts View	157
Customizing the Hosts View	158
Interpreting the Hosts View	158
Drilling Down from the Hosts View	160
Maintaining an Alert Rule or Comments	160

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 layout view. For information about the layout and controls in the detail layout views, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

Description of the Hosts View

The following image is an example of the Hosts view:



Graph

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

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.

Graph Entity Table

Diagnostics lists the hosts that pertain to the context shown in the bread crumbs in the graph entity 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.

Details Pane

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

Accessing the Hosts View

There are two primary 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. The time period depends on the mode of integration, as described on the previous page.

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 on a Host” on page 210.

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 Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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.

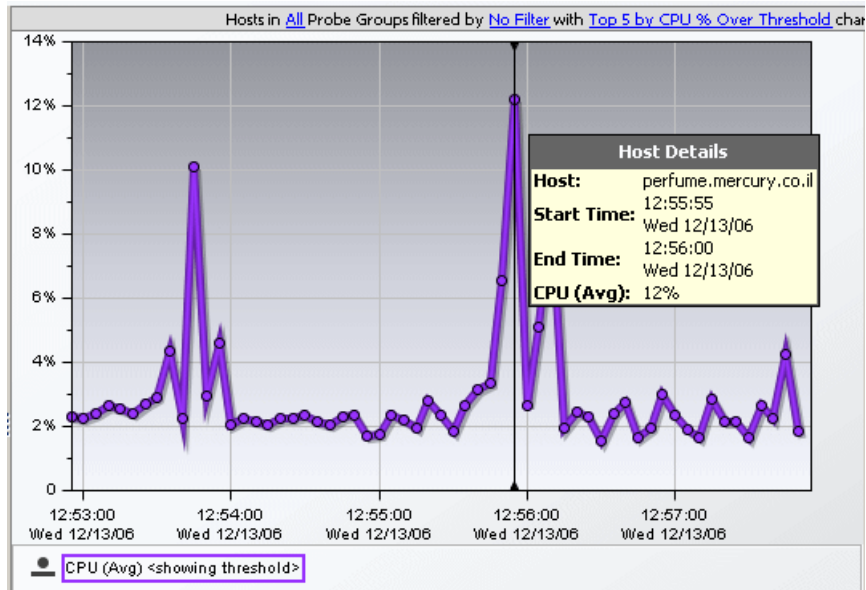
Displaying Host Details from the Charted Metrics

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.
- ▶ **CPU (Avg).** 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 Hosts View

You can drill down from a host in the Hosts View by:

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

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 Chapter 14, “Probes View.”

Maintaining an Alert Rule or Comments

When you right-click a row in the graph entity 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. These options are also available in the Common Tasks Pane. For information on alert notification rules, see Chapter 7, “Alert Notification.” For information on entity comments, see “Maintaining an Alert Rule or Comments” on page 160.

12

Load View

This chapter explains how to work with the Diagnostics Load view.

This chapter describes:	On page:
Using the Load View	161
Description of the Load View	162
Accessing the Load View	163
Customizing the Load View	164
Interpreting the Load View	164
Drilling Down to a Layer in the Graph Entity Table	167

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 layout view.

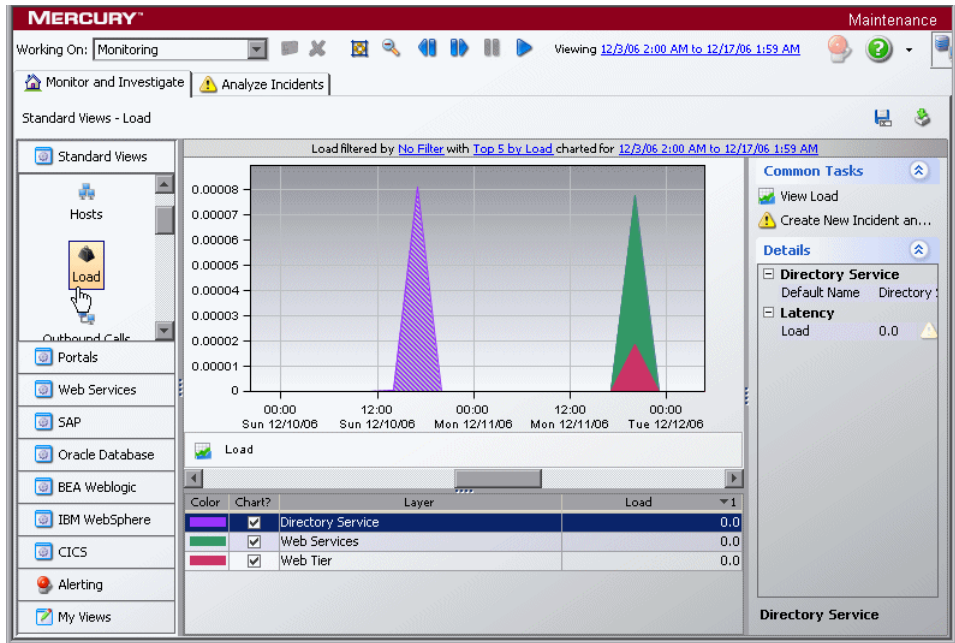
For information about Diagnostics layers, see the *Mercury Diagnostics Installation and Configuration Guide*. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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.

Graph Entity Table

The graph entity 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.


Details Pane

The **Details pane** in the Load view lists the metrics for the selected row of the graph entity 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 layout 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 Graph Entity Table” on page 180.
- ▶ Instructions for drilling down to the Load view for a particular probe in the Status view can be found in “Drilling Down on a Probe” on page 209.

Customizing the Load View

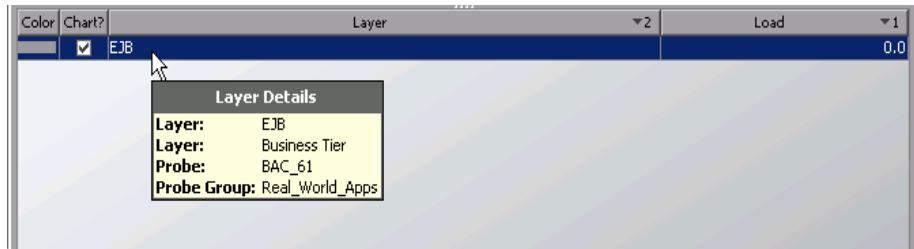
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Load view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 Graph Entity Table

You can view more information about a layer listed in the graph entity 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:

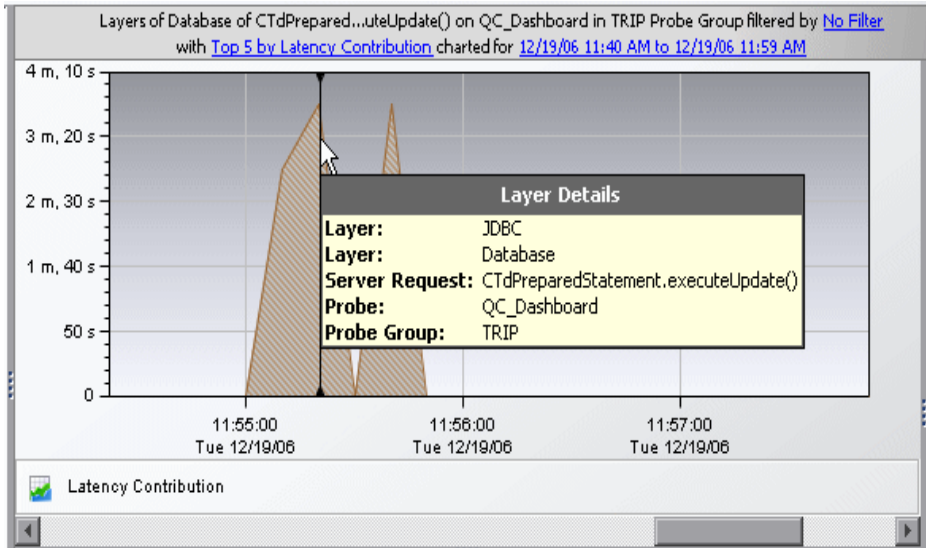


The information included in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Load view. In the example above, the Load view was accessed by drilling down from a probe in the Probes view through a layer in a higher-level Load view. The tooltip displays the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

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

When you double-click a row in the graph entity 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 graph entity table, Diagnostics displays a menu with **View Load** as the only drilldown 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.

13

Outbound Calls View

This chapter explains how to work with the Diagnostics Outbound Calls view.

This chapter describes:	On page:
About the Outbound Calls View	170
Accessing the Outbound Calls View	170
Description of the Outbound Calls View	170
Drilling Down from the Outbound Calls View	170

About the Outbound Calls View

The Outbound Calls view monitors server activity that is distributed between multiple servers. This view displays performance metrics for the outbound calls made from within your monitored environment. This view has the layout of a typical Diagnostics view. For information about the layout and controls of a typical Diagnostics view, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

Types of Outbound Calls Monitored by Diagnostics

Diagnostics monitors the following types of outbound calls:

- **Web Service.** Web services are also monitored in the Web service views. For more information, see “Web Services Views” on page 271.
- **RMI.** Calls between Java servers.
- **RFC (SAP / R3).** Calls between SAP servers.
- **CICS.** Calls within an IBM environment.
- **JMS.** Calls between Java servers and message servers.

Accessing the Outbound Calls View

You can access the Outbound Calls view from the Standard Views group on the View bar or by drilling down from the originating server request in the the Server Requests view.

For information about drilling down to the Outbound Calls view for a particular server request in the Server Requests view, see “Drilling Down from a Server Request in the Graph Entity Table” on page 194.

To access the Outbound Calls view using the View bar:

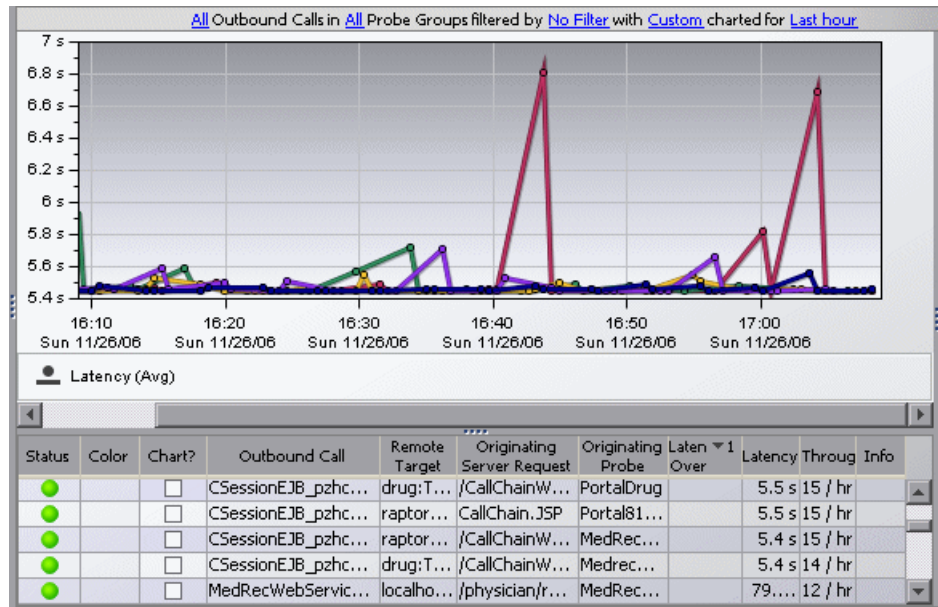
- 1 Open the **Standard Views** group on the View bar.
- 2 Click **Outbound Calls** in the **Standard Views** group.



The Outbound Calls view is displayed with the default settings so that the five calls that have the highest latency during the selected time range are displayed on the graph.

Description of the Outbound Calls View

In Diagnostics, outbound Web service calls are displayed as remote calls within a server request. The following image is an example of the Outbound Calls view:



By default, the Outbound Calls view graph displays the five calls that have the highest average latency during the selected time range. Diagnostics displays the average latency for each call using a trend line.

The Outbound Calls graph entity table includes the following columns:

- ▶ **Outbound Call.** The name of the outbound call. The name represents the entry point to the remote target.
- ▶ **Remote Target.** The destination of the outbound call.
- ▶ **Originating Server Request.** The server request containing the outbound call.
- ▶ **Originating Probe.** The probe from which this outbound call was made.

When you hold your mouse pointer over the call in the **Outbound Call** column of the graph entity table, you can view a tooltip that includes information about the selected call including the type of outbound call (**Call Type**).

For information about the types of outbound calls that Diagnostics monitors, see “Types of Outbound Calls Monitored by Diagnostics” on page 170.

Drilling Down from the Outbound Calls View

When you right-click a row in the graph entity table, Diagnostics displays a menu for the selected outbound call with the following drilldown options:

- ▶ **View Server Requests.** Diagnostics displays the originating server request in the Server Requests view. For more information about the Server Requests view, see Chapter 16, “Server Requests View.”
- ▶ **View Profiler for <probe name>.** Diagnostics opens the Profiler for the probe from which this outbound call was made. For information on the Mercury Diagnostics Profilers, see Part V, “Using the Mercury Diagnostics Profiler for J2EE” or Part VI, “Using the Mercury Diagnostics Profiler for .NET.”

14

Probes View

This chapter explains how to work with the Diagnostics Probes view.

This chapter describes:	On page:
Using the Probes View	174
Description of the Probes View	175
Accessing the Probes View	176
Customizing the Probes View	177
Interpreting the Probes View	177
Drilling Down from a Probe in the Graph Entity Table	180

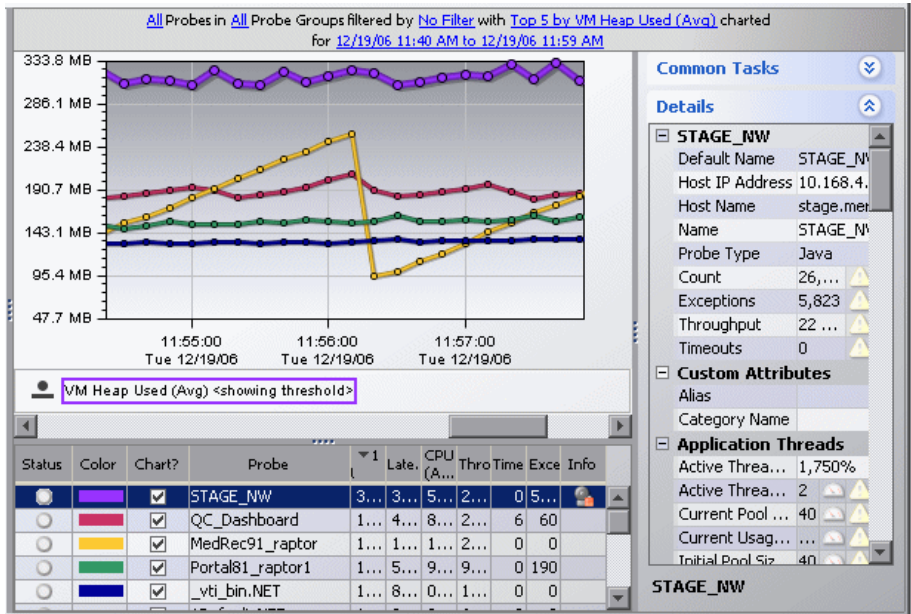
Using the Probes View

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 layout view. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 the 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.

When Diagnostics is integrated with Performance Center or LoadRunner, the x-axis of the graph shows the elapsed time since the beginning of the current scenario. In all other cases, the x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the heap usage amount in megabytes (mb) or other related units, if more appropriate.

Graph Entity Table

The graph entity table in the Probes view lists all of the probes that pertain to the context shown in the breadcrumbs 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 view filter specifies the **Time Range** and **Probe Group** for the metrics displayed in this table.

Details Pane

The Details pane in the Probes view lists the metrics for the selected row of the graph entity 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, from a dashboard view that contains a monitoring version of the Probes view, from the Hosts screen, or from the Topology 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 on a Host” on page 210.

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 Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 Graph Entity Table

You can view details for a probe listed in the graph entity 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 H. Used	Latency	CPU (Avg)	Through...	Timeouts	Exceptions	Info
●	■	<input checked="" type="checkbox"/>	NetFaultThrower2.NET	212.1 MB						
●	■	<input checked="" type="checkbox"/>	NetFaultThrower.NET	212.1 MB						
●	■	<input type="checkbox"/>	OrderWebService.NET							
●	■	<input checked="" type="checkbox"/>	ContactVendor.NET							
●	■	<input checked="" type="checkbox"/>	LibrarySite.NET							
●	■	<input checked="" type="checkbox"/>	LibraryWebService.M							
●	■	<input type="checkbox"/>	WAS6_Spurs							

Probe Details

Probe: NetFaultThrower.NET

Probe Group: NET

Host: skasabi-cr-il.mercury.global

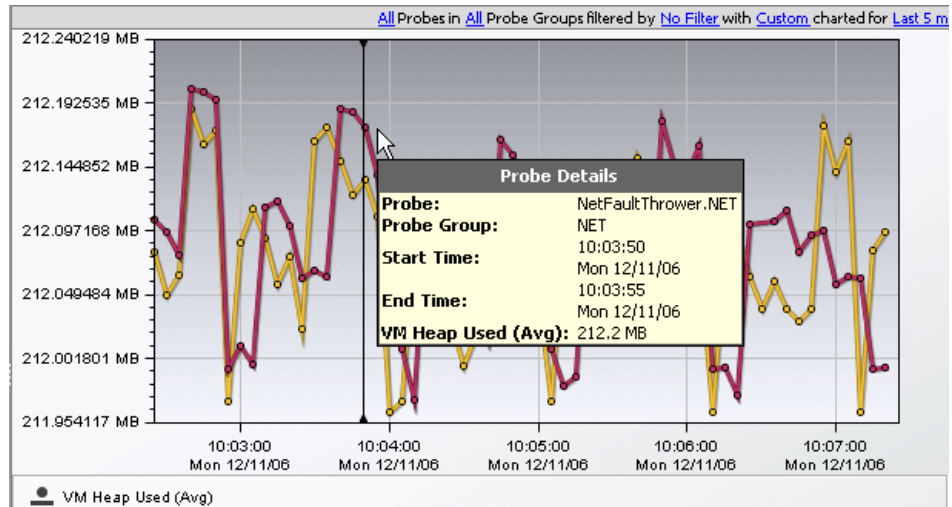
Probe Type: .NET

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 a trend line 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.

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 Graph Entity Table

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

When you double-click a row in the graph entity 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 graph entity 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 Chapter 16, "Server Requests View."
- ▶ **View Probe Summary.** The Probe Summary is a dashboard view that is made up of detail layout views. For information on dashboard views, see "Dashboard Layout Views" on page 13.
- ▶ **View Portal Components.** For information on the Portal Components view, see "Portal Components View" on page 264.
- ▶ **View Load.** For information on the Load view, see "Load View" on page 161.
- ▶ **View Trended Methods.** For information on the Trended Methods view, see "Trended Methods View" on page 233.
- ▶ **View Profiler for <probe name>.** For information on the Mercury Diagnostics Profilers, see Part V, "Using the Mercury Diagnostics Profiler for J2EE" or Part VI, "Using the Mercury Diagnostics Profiler for .NET."
- ▶ **Add/Edit Comment.** For information on entity comments, see "Maintaining an Alert Rule or Comments" on page 160.
- ▶ **Add/Edit Alert Rule.** For information on alert notification rules, see Chapter 7, "Alert Notification."

15

SQL Statements View

This chapter explains how to work with the SQL Statements view.

This chapter describes:	On page:
Using the SQL Statements View	181
Accessing the SQL Statements View	184
Description of the SQL Statements View	185

Using the SQL Statements View

Important: The SQL Statements view should only be used when you are working with a Diagnostics probe from version 6.5 or later.

The SQL Statements view displays performance metrics for SQL statements executed by certain methods within your monitored environment. This view is presented in detail layout. For more information about views presented in detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

The SQL statements in this view are aggregated according to probe group. This means that if the same SQL statement is called from two different probe groups it will be displayed twice. However, if the same statement is called from two different probes within the same probe group, it will be displayed only once.

Note: This behavior is significantly different from most other Diagnostics views. In other views, activity that takes place on different probes, is displayed separately.

SQL Trending Threshold

Trending of the SQL statements begins only after an SQL statement exceeds the predefined latency threshold. After the SQL statement exceeds this threshold, trending continues, even if it falls below the latency threshold.

In the following cases, SQL trending continues indefinitely:

- ▶ The .NET Probe in all deployments
- ▶ The J2EE Probe in a deployment where Diagnostics is integrated with LoadRunner or Performance Center

If the J2EE Probe is running in a deployment where Diagnostics is integrated with Mercury Business Availability Center or where Diagnostics is running in standalone mode (no integration), SQL trending continues until the application monitored by the J2EE Probe is restarted.

The default SQL trending latency threshold is one second. You can change this threshold in the property files for the Diagnostics components. There are two different properties that can define the SQL trending latency threshold:

- ▶ **minimum.sql.latency** in `<J2EE_probe_install_dir>\etc\dispatcher.properties`
- ▶ **sql.latency.trim** in `<Diag_server_install_dir>\etc\server.properties` file

The one that you must use depends on the type of probe you are using and the way that you Deployed Diagnostics.

The following table describes in which file to define the threshold depending on your deployment and the type of probe.

	.NET Probe	J2EE Probe
Diagnostics Standalone	server.properties file	dispatcher.properties file
Integrated with Mercury Business Availability Center	server.properties file	dispatcher.properties file
Integrated with LoadRunner or Performance Center	server.properties file	server.properties file

Note: Lowering the SQL trending latency below one second, might add significant overhead to the Diagnostics system.

To define the SQL trending latency threshold in the `dispatcher.properties` file:

- 1 Open the `<J2EE probe_install_dir>\etc\dispatcher.properties` file.
- 2 Adjust the latency threshold by setting the property shown in the following example:

```
minimum.sql.latency = 1s
```

To define the SQL trending latency threshold in the server.properties file:

- 1 Open the **server.properties** file of the Diagnostics Server in Mediator mode to which the probe is connected
(`<Diag_mediating_server_install_dir>\etc\server.properties` file).
- 2 Adjust the latency threshold by setting the property shown in the following example:

```
sql.latency.trim=1000ms
```

Note: The threshold is changed for all probes connected to this Diagnostics Server in Mediator mode.

Accessing the SQL Statements View

You can access the SQL Statements view from the Standard Views group on the View bar.

To access the SQL Statements view using the View bar:

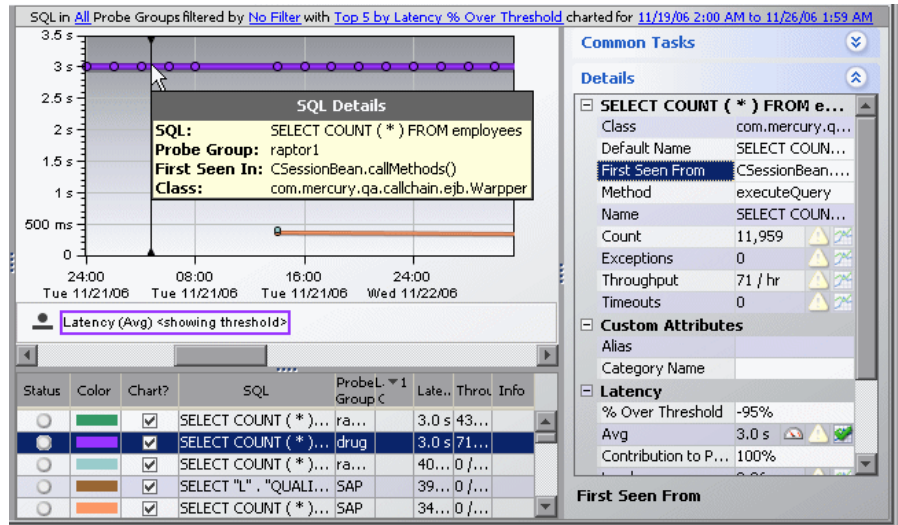
- 1 Open the **Standard Views** group on the View bar.
- 2 Click **SQL Statements** in the **Standard Views** group.



The SQL Statements view is displayed with the default settings so that the five statements that have the highest latency during the selected time range are displayed on the graph.

Description of the SQL Statements View

The following image is an example of the SQL Statements view:



Graph

By default, the SQL Statements view graph displays the five statements that have the highest average latency during the selected time range. Diagnostics displays the average latency for each statement using a trend line.

When Diagnostics is integrated with Performance Center or LoadRunner, the x-axis of the graph shows the elapsed time since the beginning of the current scenario. In all other cases, the x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the average latency in seconds and milliseconds.

Graph Entity Table

The graph entity table in the SQL Statements view lists all of the SQL statements according to the relevant filters defined in the view filters.

Details Pane

The Details pane in the SQL Statements view lists the metrics for the selected row of the graph entity table. This includes the **First Seen From** metric, which displays the name of the method and in which the SQL statement was first identified.

For more information about the Details pane, see “Viewing Metrics in the Details Pane Area” on page 59.

Tooltip

When you hold your mouse pointer over the statement in the **SQL** column of the graph entity table, you view a tooltip that includes information about the selected statement. You can view this same information by holding your mouse pointer over the trend line for a charted metric. This information includes the name of the method in which the SQL statement was first identified (**First Seen In**).

When you hold your mouse pointer over a particular point in the graph, you can view the start time and end time of the aggregation period represented by the selected data point.

Viewing the Entire SQL Statement

You can view the entire SQL statement in a separate dialog box by double-clicking the SQL statement in the graph entity table. Alternatively you can right-click the selected SQL statement in the graph entity table and selecting **View entire SQL statement** from the menu or you can select the **View entire SQL statement** option from the Common Tasks menu.

16

Server Requests View

This chapter explains how to work with the Diagnostics Server Requests view.

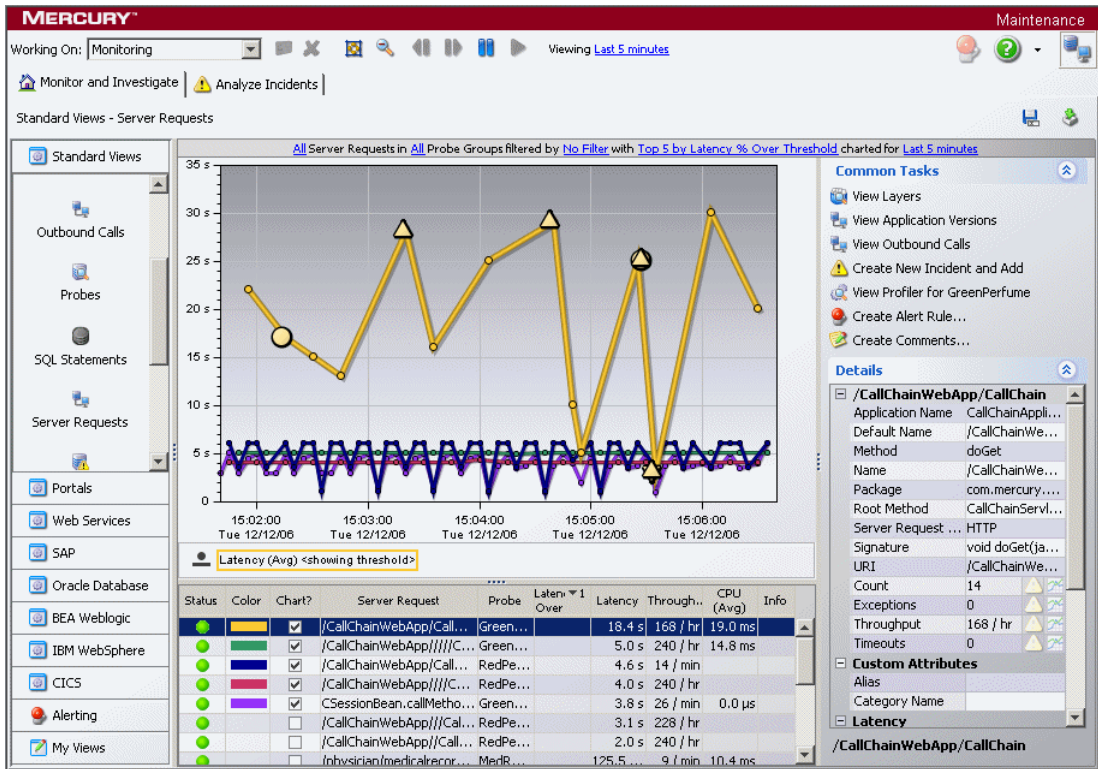
This chapter describes:	On page:
Using the Server Requests View	187
Accessing the Server Requests View	189
Customizing the Server Requests View	190
Interpreting the Server Requests View	191
Drilling Down from a Server Request in the Graph Entity Table	194
Drilling Down to an Instance Tree for a Server Request	195

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 layout. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

By default, the Server Requests view displays the average latency for the five server requests that have the highest latency percent over threshold values.

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



Accessing the Server Requests View

You can access the Server Requests view from the **Standard Views** group on the View bar, from a dashboard view that contains a monitoring version of the Server Requests view, or from some other detail layout by drilling down on an entity.

To access the Server Requests 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 graph displays the average latency for the five server requests that have the highest latency percent over threshold values .

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 layouts:

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. You can also drill to the originating server request for an Outbound Call and drill to the server requests for a probe from Topology.

- ▶ To drill down to the Server Requests view for a particular transaction in the Transactions view, see “Drilling Down from a Transaction in the Graph Entity Table” on page 232.
- ▶ To drill down to the Server Requests view for a particular probe in the Probes view, see “Drilling Down from a Probe in the Graph Entity Table” on page 180.
- ▶ 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 208.

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 “Viewing Diagnostics Data in Detail Layout” on page 41.

Displaying CPU Time for Server Requests

By default, CPU time for server requests is displayed on the Server Requests view. Both average and total CPU time appear in the Details pane, and CPU(Avg) is a default column in the graph entities table in the detail layout.

You configure displaying the CPU time for server requests in the probe. In the Java probe, set the **property `cpu.timestamp.collection.method` in `dynamic.properties`**. In the .NET probe, set the XML element **`cputime` in `probe_config.xml`**.

Setting the value of these properties to **true** causes Diagnostics to report the CPU time for each server request in the Server Request view. Setting the value of these properties to **false** causes Diagnostics to stop reporting 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 Details pane. To configure displaying the CPU column in the graph entity table, modify the graph entity table as described in “Customizing the Graph Entity Table” on page 81.

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.

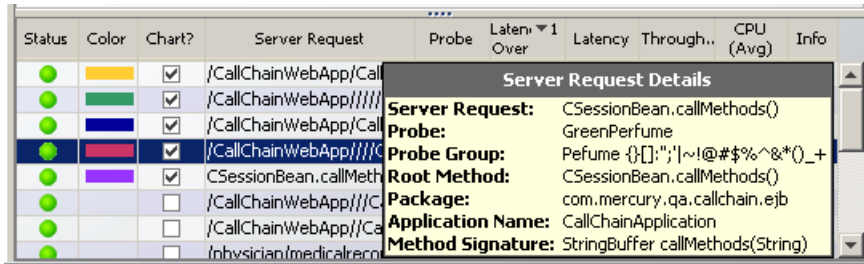
Depending on the app & workload, user may see an average of zero latency if they look at average response time on the probe. This is caused by the low-resolution java timer (the java timer can't measure things that take less than 10ms on many platforms, including Windows). The Server Requests that typically fall into this category include static content (e.g. GIFs, JPEGs) and Database and housekeeping threads in an application. If this is an important issue for a given customer, one way to get non-zero response times is to turn on the native (JNI) timestamps for high resolution timings.

Note: Under certain situations you may see an average latency of zero reported in the Server Requests view. This happens because on many platforms, including Windows, the low-resolution java timer cannot measure time intervals that take less than 10ms. The Server Requests that typically fall into this category include static content such as GIFs and JPEGs and Database and housekeeping threads in an application.

If you need to see the latency at lower resolutions you can get non-zero response times by turning on the native (JNI) timestamps for high resolution timings.

Displaying Server Request Details from the Graph Entity Table

You can view more information about a server request listed in the graph entity 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:

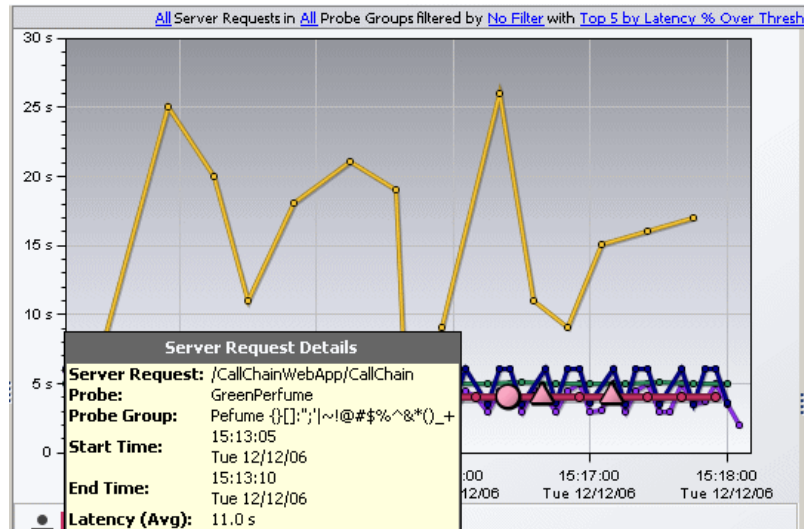


The **Server Request Details** tooltip displays the following information:

- ▶ **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 signature of the root method.
- ▶ **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.
- **Latency (Avg).** The average latency for the server request during the aggregation period.

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 Graph Entity Table

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

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

When you right-click a row in the graph entity 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. This Layers view displays a breakdown of the latency contribution for each layer in the server request. For more information on the Layers view, see “Layers View” on page 239.
- ▶ **View Application Versions.** This option is relevant for WebLogic version 9 or later.
- ▶ **View Outbound Calls.** For more information, see “Outbound Calls View” on page 169.
- ▶ **Create New Incident and Add, or Add to Active Incident.** For more information, see “Create New Incidents” on page 110.
- ▶ **View Profiler for <probe name>.** For information on the Mercury Diagnostics Profilers, see Part V, “Using the Mercury Diagnostics Profiler for J2EE” or Part VI, “Using the Mercury Diagnostics Profiler for .NET.”
- ▶ **Create/Edit/Delete Alert Rule.** For information on alert notification rules, see Chapter 7, “Alert Notification.”
- ▶ **Create/Edit/Delete Comments.** For information on entity comments, see “Maintaining an Alert Rule or Comments” on page 160.

Drilling Down to an Instance Tree for a Server Request

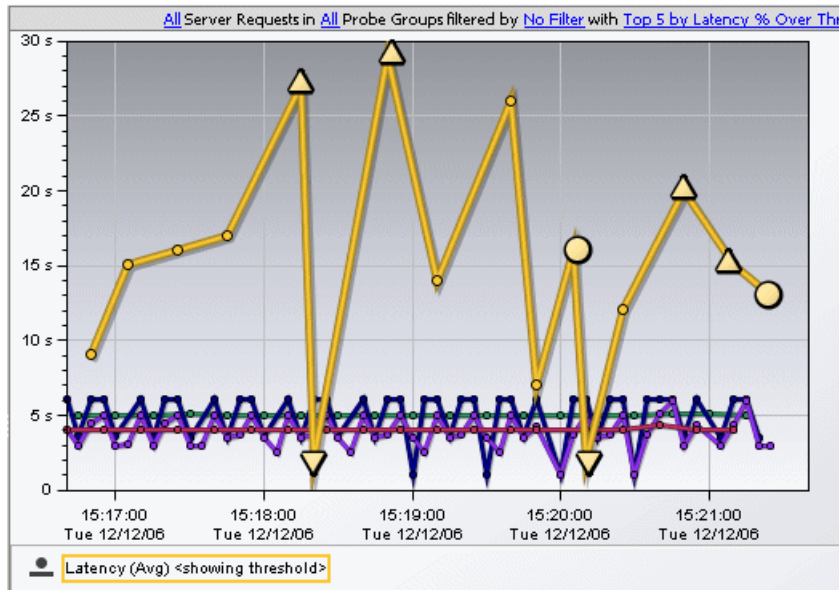
Each node of a trend line in the Server Requests view represents data aggregated from all server requests in a specific time interval. In addition to this, the Server Requests view displays information about specially selected individual server request instances. You can drill down from the graph in the Server Requests view to see a call profile for any of these instances. 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 instance trees see Chapter 9, “Instance Trees”. For more information on the Call Profile view, see Chapter 22, “Call Profile View.”

When you select a server request in the Server Request view, by either clicking a trended metric for the server request in the graph or by selecting the row for the server request from the graph entity 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 and for the Cross-VM calls.

In the following example, the yellow 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 server request. The markers are placed on the graph to indicate the total latency and the end time of 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.**

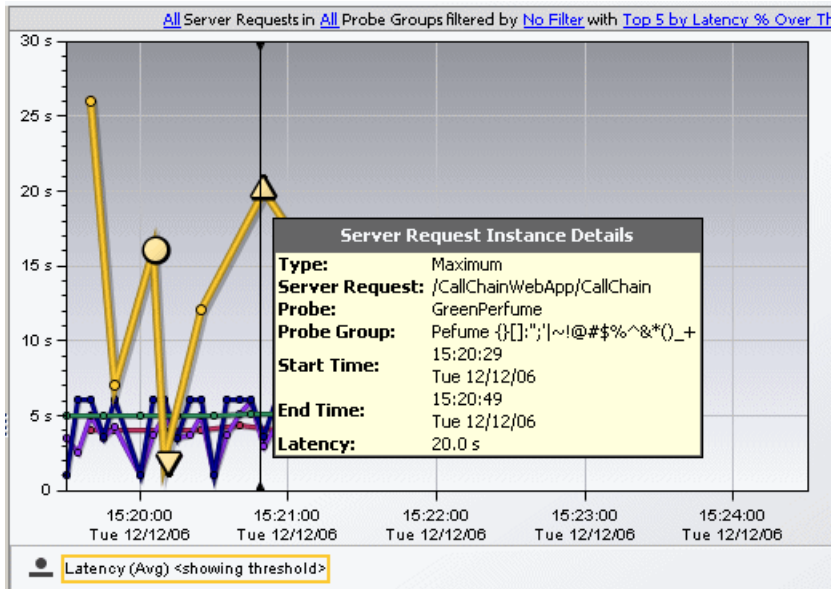
The average instance tree for a server request represents an instance tree that represents the average latency for the time period. This depicts a typical invocation of the selected server request and resulting root method call, and can be useful for comparison to instances that perform poorly.



► **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.

You can view more information about the server request represented by an instance tree marker by holding the mouse pointer over the marker until the Server Request Instance Details tooltip is displayed, as shown in the following example:



The instance tree marker tooltip displays the following information:

- **Type.** The type of instance tree marker that has been selected.
- **Server Request.** The name of the selected server request, as displayed 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.
- ▶ **Start Time.** The start time for the server request invocation represented by the selected instance tree.
- ▶ **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 to the Instance Tree in the Call Profile view

There are two ways to drill down to 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 and select either "View Call Profile" or "View Call Profile in New Window". Opening the call profile in a new window 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 Chapter 22, "Call Profile View."

17

Status View

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

This chapter describes:	On page:
About the Status View	202
Accessing the Status View	204
Customizing the Status View	205
Interpreting the Status View	206
Drilling Down from the Status View	208

About the Status View

The *Status* view is a table that contains the probe groups that you defined when you installed the probes. Within each probe group in the Status view probe group table there are two sub-tables. The first is the *Probe* table, which lists all of the probes in the probe group, and the second 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 highlights the features and controls of the view.

The screenshot shows the Status View interface with the following components labeled:

- Probe Group Status:** Points to the top-level status bar.
- Probe Group:** Points to the 'Default' probe group in the tree.
- Probe:** Points to the 'WL81-W5' probe in the table.
- Host:** Points to the 't-lc8.lab.performant.com' host in the table.
- Status View Title:** Points to the main title 'Status for Last 5 minutes baselined against Last 6 hours'.
- Baseline Control:** Points to the 'Last 6 hours' link in the title.
- Probe Group Headers:** Points to the header row of the probe group table.
- Host Status:** Points to the 'Host' column header in the host table.
- Threshold Alert Indicator:** Points to the red box around the '65%' CPU usage value.
- Trend Indicator:** Points to the small arrows next to the CPU usage values.
- Host Table Header:** Points to the header row of the host table.
- Probe Status:** Points to the status icon (green dot) next to the probe name.
- Probe Table Header:** Points to the header row of the probe table.

Probe Group	Probe	VM Heap Used	Latency	Count/S	Timeouts	Exceptions	Info
Default	WL81-W5	42.1 MB	1.0 s	0.033	0	5	

Host	CPU	Disk Bytes/S	Network Bytes/S	Page In/S	Memory	Info
t-lc8.lab.performant.com	65%	13.6 KB	926.9 KB	0	82%	

Probe Group	Probe	VM Heap Used	Latency	Count/S	Timeouts	Exceptions	Info
Tornado Probes	WAS_server1	83.1 MB					
	ALMAGRO						

Host	CPU	Disk Bytes/S	Network Bytes/S	Page In/S	Memory	Info
chtang.performant.com	2%	17.2 KB	21.7 KB	2	44%	
almagro.lab.performant.com	12%	781.8 KB	25.3 KB	165	97%	

Status View Title and Status Baseline Control

The title of the Status view contains the baseline control. You use this control to set the time that Diagnostics uses as the baseline performance level to compare the current performance level against. The result of the comparison of the baseline performance and the current performance is shown using the *metric trend indicators* that are displayed following the metrics values in the metric columns of the Status view tables. Precise information about how the current performance of a given metric compares against its baseline performance is available in the tooltip for the metric.

To set the baseline time for the Status view, click the baseline control in the Status view title and select a baseline time period from the drop-down menu. When you have made your selection, Diagnostics sets the value displayed in the baseline control to the value you selected and displays revised trend indicators next to the appropriate metrics in the Status view tables.

Probe Group

The top level in the Status view table is the probe group. You assign probes to specific probe groups during the probe installation process.

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. This indicates how the performance of all of the probes in that probe group compares with their performance thresholds.

The probes and hosts for each probe group are listed in the tables following each probe group entry in the Status view.

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 the 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 that contain a Status view.

To access the Status view using the View bar:



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

The Status view is displayed.

To access the Status view from a dashboard view:

- Open a dashboard view where 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.

The Server Summary view is an example of a dashboard view that contains a concise Status view.

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, “Viewing Diagnostics Data in Detail Layout.”

Note: Changing the global time range has no effect on the Status view—it is always based on the last five minutes.

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 to 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 35.

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 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 the Status of Selected Entities



► **Status.** This indicator shows how the listed entity or any of its child entities is performing relative to the thresholds that you set for their metrics.

For example, a probe can have a critical status even when none of the probe's own metrics have exceeded their thresholds. The probes status will become critical when any of the server requests for that probe become critical.

The color of the indicator stands for the severity of the threshold violation. If you hold the mouse pointer over the status indicator in a row of the table, Diagnostics displays a tooltip with a description of the current status:

Status Indicator Color	Description
Green	Good - The entity 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 entity table, the cell for the metric is outlined in yellow as well.

Status Indicator Color	Description
Red	<p>Critical - The component is consistently exceeding defined thresholds.</p> <p>If the metric that has been exceeding the defined threshold is displayed in the entity table, the cell for the metric is outlined in red as well.</p>
Grey	<p>No status information available.</p> <p>Either no data has been received for the metric or no threshold has been set.</p>

For information on setting thresholds, see “Setting Metric Thresholds” on page 63.

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.

The **Probe Details** tooltip includes 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 into the details for the probes, hosts, and layers that are behind the status that you see portrayed in view.

Drilling Down on a Probe Group

You can drill down on a probe group listed in the Status view by right-clicking the row containing the probe group. Diagnostics displays a menu with the following options:

- ▶ **View Probe Group Summary.** The Probe Group Summary is a dashboard view made up of detail layout views. For information on dashboard views, see “Dashboard Layout Views” on page 13.
- ▶ **View Probes.** For information on the Probes view, see “Probes View” on page 173.
- ▶ **View Hosts.** For information on the Hosts view, see “Hosts View” on page 155.
- ▶ **View Load.** For information on the Load view, see “Load View” on page 161.
- ▶ **View Trended Methods.** For information on the Trended Methods view, see “Trended Methods View” on page 233.
- ▶ **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one. For more information, see “Alert Notification” on page 91.
- ▶ **Create/Edit/Delete Comments.** Create a comment, or edit or delete an existing one. For more information, see “Managing Comments for an Entity” on page 88.

Note: The above drilldowns are filtered by the selected probe group and display data for the last five minutes, regardless of the global time range.

Drilling Down on a Probe

You can drill down on a probe listed in the Status view by double-clicking or right-clicking on the row in the Probe table that contains the probe.

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 in the Probe view. For information on the Probe view see, “Probes View” on page 173.
- ▶ **View Server Requests.** For information on the Server Requests view, see “Server Requests View” on page 187.
- ▶ **View Probe Summary.** The Probe Summary is a dashboard view made up of detail layout views. For information on dashboard views, see “Dashboard Layout Views” on page 13.
- ▶ **View Portal Components.** For information on the Portal Components view, see “Portal Components View” on page 264.
- ▶ **View Load.** For information on the Load view, see “Load View” on page 161.
- ▶ **View Trended Methods.** For information on the Trended Methods view, see “Trended Methods View” on page 233.
- ▶ **View Profiler for <probe name>.** 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 V, “Using the Mercury Diagnostics Profiler for J2EE” or Part VI, “Using the Mercury Diagnostics Profiler for .NET.”

- ▶ **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one. For more information see Chapter 7, “Alert Notification”.

- ▶ **Create/Edit/Delete Comments.** Create a comment, or edit or delete an existing one. For more information see “Managing Comments for an Entity” on page 88.
- ▶ **Delete <entity>.** Delete the selected entity and its metrics from Diagnostics. For more information see “Deleting an Entity” on page 88.

Note: The above drilldowns are filtered by the selected probe and display data for the last five minutes, regardless of the global time range.

Drilling Down on a Host

You can drill down on 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.

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

- ▶ **View Hosts.** For information on the Hosts view, see “Hosts View” on page 155.
- ▶ **View Probes.** For information on the Probes view, see “Probes View” on page 173.
- ▶ **Create/Edit/Delete Alert Rule.** Create an alert rule, or edit or delete an existing one. For more information see Chapter 7, “Alert Notification”.
- ▶ **Create/Edit/Delete Comments.** Create a comment, or edit or delete an existing one. For more information see “Managing Comments for an Entity” on page 88.

Note: The above drilldowns are filtered by the selected host and display data for the last five minutes, regardless of the global time range.

Status Propagation

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

18

Topology View

This chapter describes the Topology 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 Topology View	214
Description of the Topology View	215
Accessing the Topology View	221
Working with the Topology Diagram	221
Drilling Down from the Topology View	226

About the Topology View

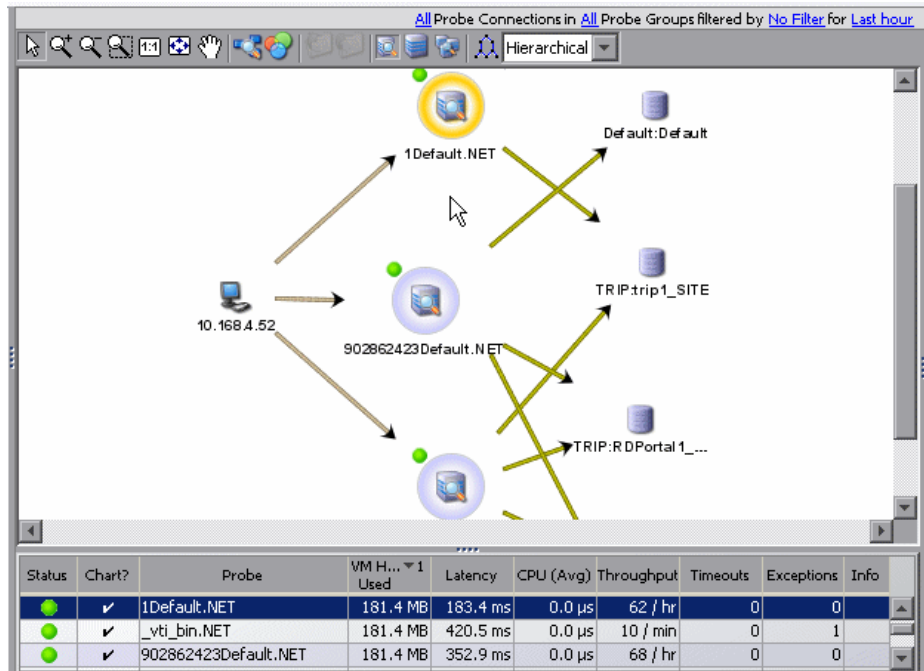
The Topology view provides a graphical representation of the way that your applications or business processes are working by generating a diagram that depicts probes, probe connections, and targets. Depending on your instrumentation, the probes generally correspond to the applications that they are monitoring. The probe connections represent inbound and outbound calls to and from an instrumented application. Targets provide information on the origin of inbound calls or the destination of outbound calls.

The Topology view lets you display topology diagrams for the probes or probe connections listed in the graph entity table. When you select one of these entities, Diagnostics displays a topology diagram that shows all of the probes, probe connections, and targets that interact in the processing for the operations that occur on the selected entity.

Note: The probe connections that are available to be depicted in this view are highly dependent on probe instrumentation.

Description of the Topology View

The Topology view has the appearance and controls of standard Diagnostics views with a detail layout. The following image is an example of the Topology view:



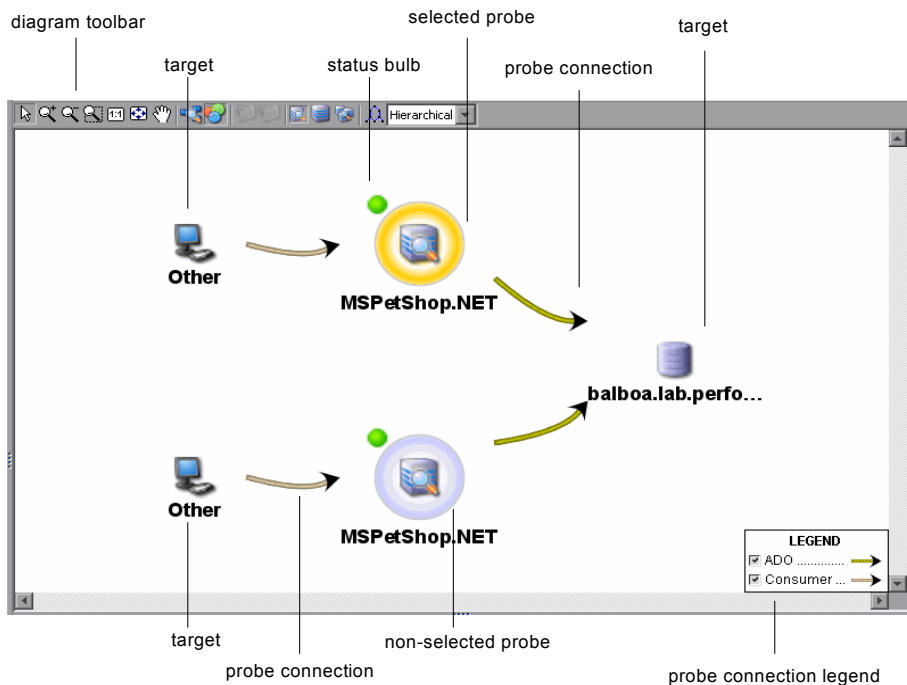
This section includes:

- “Topology Diagram” on page 216
- “Graph Entity Table” on page 220
- “Details Pane” on page 220

Topology Diagram

Diagnostics displays the topology diagram where the graph is normally displayed in views with a detail layout. The diagram includes a tool bar with controls that allow you to manipulate the diagram by panning and zooming and alter the way that the diagram is drawn by grouping the diagrammed entities or changing the type of the diagram that is drawn. Instructions for how to use the diagram tool bar and the diagrammed entities to enhance your understanding of processing in your applications are presented in “Working with the Topology Diagram” on page 221.

The objects that appear in the diagram are shown in the following screen image and described in the text that follows:



Probes

The probes nodes generally represent the applications that they are monitoring. When a diagramed probe has been selected from the probes tab in the graph entity detail table, the probe is displayed with a gold halo to indicate that it is the selected probe. The metrics for the selected probe are displayed in the details table and the row for the probe in the graph entity table is shown as selected.

The status bulb next to each probe in the diagram indicates the status of probe just as it does in the Status view and in the graph entity table of the Topology view.

Though topology by default doesn't group probes, probes can also be grouped in the diagram by hosts or by probe groups. For information about working with grouped probes in the diagram, see “Grouping Probes” on page 222.

Probe Connections

The probe connections represent aggregated inbound and outbound calls to the monitored applications. Normally, probe connections are only shown in between probes and targets; although it is possible for probe connections to be drawn between probes. The arrow on the probe connection indicates the direction of the communication.

When the probe connection points from a target to a probe, it is an inbound probe connection. The target from which the probe connection originates represents the origin of an aggregation of inbound calls. The metrics displayed in the Topology view for an inbound probe connection are measured from the callee, which is the application that is monitored by the the probe. However, the probe connection is shown from the caller's point of view in the diagram.






When the probe connection points from a probe to a target, it is an outbound probe connection. The target of an outbound probe connection represents the destination of an aggregation of outbound calls from the caller's point of view.

The color of the probe connection indicates the type of call. You can see the type of call in the details table when you select a probe connection from the diagram or from the graph entity table, or you can display the legend for the topology diagram and determine the type based on the color of the probe connection. If you select a probe connection from the diagram or from the probe connections tab in the graph entity table, the probe connection is shown in the diagram with a dashed black line on top of the color shown in the legend.

Targets

Targets, as determined through the instrumentation of your probes, represent the origin of inbound calls or the destination of outbound calls. A variety of icons are used to represent the different types of targets that Diagnostics recognizes. In the diagram above, the desktop icon represents a target type of consumer and the data store icon represents a target type of JDBC. You cannot select targets in the diagram and you cannot see metrics for targets in the other parts of the Topology view.

Every type of target corresponds roughly to a type of communication technology. The following target types are shown in the topology diagram:

Topology Target Icon	Descriptions
	Inbound and outbound Web Service arcs, for Java and .NET
	Inbound HTTP Consumer arcs, for Java and .NET
	Inbound and outbound RMI arcs, for Java
	Outbound JDBC arcs Outbound .NET ADO calls
	Outbound JMS


Topology Target Icon	Descriptions
R/3	Outbound SAP R/3
	CICS (outbound)

Diagram Toolbar

The Diagram Toolbar appears at the top of the diagram right below the view title. The toolbar allows you to control the appearance of the diagram by scrolling and zooming and by toggling the appearance of useful aids like the diagram overview box and the diagram legend.

Diagram Overview

The diagram overview gives you a view of the entire diagram to assist you in scrolling and zooming the diagram so that the information that you need is visible in the Topology view. The diagram overview is especially helpful when you are using the zoom box tool as it can help you see where the box that you zoomed in on fits in the big-picture diagram.

You can also use the diagram overview to pan and zoom the view of the diagram. The diagram overview is outlined by a black view frame which is initially not visible beyond the edges of the diagram overview. You can pan the primary diagram by dragging this view frame to reposition it within the diagram overview. You can zoom the primary diagram by resizing the view frame at the handles and dragging to the desired zoom level.



To display the diagram overview, click the **Diagram Overview** button in the diagram toolbar. Diagnostics displays the overview box in the bottom right corner of the diagram. Clicking the **Diagram Overview** button again removes the overview box. If the legend was displayed, it is removed when the diagram overview box is displayed.

Diagram Legend

The graph legend lists each of the probe connection types that have been included in the diagram that is currently displayed and shows the color that Diagnostics used to draw the probe connection of each type.



To display the diagram legend, click the **Diagram Legend** button in the diagram toolbar. Diagnostics displays the legend in the bottom right corner of the diagram. Clicking the **Diagram Legend** button again removes the legend. If the diagram overview box was displayed, it is removed when the legend is displayed.

You can use the check box that precedes each probe connection type entry in the legend to indicate which probe connection types are to be diagrammed. When a probe connection type is not checked, all of the probe connections for that type are removed from the diagram. This can be useful in a complicated diagram where the number of probe connections can make it difficult to understand.

Note: Filtering the diagrammed probe connections in the legend does not impact the probe connection metrics listed in the graph entity table.

Graph Entity Table

The graph entity table in the Topology view contains two tabs: the Probe Connections tab and the Probes tab. The graph entity table behaves like the table in any other view with a detail layout, except that the you cannot control which entities appear in the diagram using the Chart column. The Chart column indicates which entities are included in the diagram for the entity that you selected from the graph entity table. All of the entities that are included in the diagram for the entity that you selected are marked with a check mark in the Chart column. For more information on using the controls of a graph entity table see Chapter 6, “Working with the Graph Entity Table”.

Details Pane

The Details pane in the Topology view lists the metrics for the selected row of the graph entity table. For information on the details pane see Chapter 4, “Working with the Details Pane”.

Accessing the Topology View

You can access the Topology view from the Standard view group in the view bar or by drilling down on a transaction listed in the Transaction view. When accessed through the Transaction view, topology data is limited to the probes and connections that are part of the transaction in question.

Working with the Topology Diagram

When Diagnostics first displays the topology diagram for a probe or probe connection that you selected from the graph entity table, the diagram is displayed zoomed so that it fits the view. The default format for the diagram is hierarchical and the probes are not grouped by probe group or host.

Using the controls in the diagram and the diagram toolbar, you can control the format, layout, and resolution of the diagram so that you can see the topology in the way that you want to see it.

This section includes:

- “Grouping Probes” on page 222
- “Zooming” on page 223
- “Scrolling and Panning” on page 224
- “Adjusting the Diagram Layout” on page 224
- “Restoring the Diagram Layout” on page 225

Grouping Probes

When the topology diagram is first displayed the probes are not grouped, and that each probe is shown individually in the diagram. You can change the way that the diagram is drawn by instructing Diagnostics to group probes by hosts or by probe groups. Grouping probes can simplify the appearance of the diagram so that you can see the topology of your applications from a different point of view.

Grouping Probes by Hosts



To group probes by the hosts, click **Group by Probe Host** in the toolbar. The probes in the diagram are replaced with probe hosts which contain the probes that are installed on the host.

Grouping Probes by Probe Group



To group probes by probe groups, click **Group by Probe Group** in the toolbar. The probes in the diagram are replaced with probe groups which contain the probes that have been to the probe group.

Displaying Ungrouped Probes



To cause the probes to be displayed not grouped by hosts or probe groups, click **Don't Group Connected Probes** in the toolbar. The probes in the diagram are once more displayed as individual probes.

Working With Grouped Probes



When probes are grouped in the diagram, a small toggle icon is shown above the icon for the group. If you click on this icon or right click on the group icon, the group is opened so that you can see each of the probes in the group along with the probe connections to the targets.



You can collapse the expanded probe group by clicking the icon in the group header or by right clicking anywhere in the group box.



You can expand all of the groups in the diagram by clicking **Expand all Nodes** in the toolbar. Diagnostics opens all of the groups in the diagram.



You can collapse all of the groups in the diagram by clicking **Collapse all Nodes** in the toolbar. Diagnostics collapses all of the groups in the diagram.

Zooming

Diagnostics provides several controls that allow you to zoom in on the diagram to get a closer look at parts of a displayed topology diagram or to zoom out to see the bigger picture. There is a limit to how far you can zoom the diagrams in or out. When you have zoomed to one of these limits, using the zoom controls will no longer impact the appearance of the diagram.

Zooming In



To zoom in on the entire diagram to get a closer look at an area of the diagram, click **Zoom In**. Diagnostics zooms in on the Diagram keeping the same entities in the center of the diagram as it provides a closer look. You may need to use the scroll bar or panning to reposition the center of the view as you zoom in.

You can also zoom in using the mouse wheel. To use the mouse wheel to zoom in, hold down the CTRL key while rolling the wheel forward.



To zoom in on a selected area of the diagram click **Zoom Box** to put the pointer into zoom box mode. When in zoom box mode, the pointer is shaped like a magnifying glass. Click and drag the pointer to define the location and size of the area that you would like to zoom. When you release the mouse button, Diagnostics zooms in to show the selected area at maximum magnification. To return the pointer to selection mode, click **Zoom Box** again or click **Make Select Active**.



Zooming Out



To zoom out to get a more complete view of the diagram, click **Zoom Out**. Diagnostics zooms out on the Diagram keeping the same entities in the center of the diagram as it provides a broader view. You may need to use the scroll bar or panning to reposition the center of the view as you zoom out.

You can also zoom out using the mouse wheel. To use the mouse wheel to zoom out, hold down the **Ctrl** key while rolling the wheel backwards.



To zoom out all at once to the default view click **Fit to Contents**. Diagnostics zooms out so that the diagram fits the view keeping the same entities in the center of the diagram as it provides a broader view. You may need to use the scroll bar or panning to reposition the center of the view as you zoom out.

Scrolling and Panning

To reposition the diagram so that the entities that you want to see are visible in the diagram, you can scroll the diagram using the scroll bars or you can pan by putting the pointer in panning mode.

When the diagram is not completely displayed in the view, the scroll bars are activated so that you can use them to scroll to the location in the diagram that you want to see. When the vertical scroll bar is active, you can use the mouse wheel instead of the scroll bar to scroll vertically within the diagram.



To pan to a specific selected area of the diagram click **Pan** to put the pointer into pan mode. When in pan mode, the pointer is shaped like a hand. Click and drag the pointer to reposition your view of the diagram. When you release the mouse button, the diagram continues to be displayed in the position where you released. You may continue to pan as long as the pointer remains in pan mode. To return the pointer to selection mode, click **Pan** again or click **Make Select Active**.



Adjusting the Diagram Layout

The topology diagram can be drawn in several different layouts and you can customize the layouts by dragging the targets and probes to reposition them in a way that makes the probe connections easier to see.

Selecting the Diagram Layout

Diagnostics can display the diagram using several different layout formats. The default layout is hierarchical. You may find that one of the other layouts makes your diagram easier to understand.

To change to an alternate layout select the layout from the **Layout** dropdown. Diagnostics redraws the diagram using the selected layout.

Customizing the Diagram Layout

You can reposition the probes and the targets in the topology diagram so that the probe connections become more clear. To reposition a probe or probe connection, click on the node that you would like to move and drag it to the intended location. Diagnostics redraws the probe connections to a from the relocated node.

Note: Diagnostics redraws the diagram when an entity has been detected that must be added to the diagram or when an entity must be removed from the diagram. When Diagnostics redraws the diagram, it will zoom the diagram to fit the view and places it in the center of the view as though it were displaying the Diagram for the first time. This means that any zooming, panning, custom layout changes that you made are lost.



To avoid this, you can increase the time represented in the diagram using the Viewing time filter so that the diagram is not updated as frequently or you can pause the updates by clicking **Pause**

Restoring the Diagram Layout

When you have finished your investigation of the parts of the topology diagram that you highlighted by zooming, scrolling, panning and customizing the diagram, you may want to return to default layouts and resolution. Diagnostics provides several different options for restoring the diagram layout. Each impacts the settings that you have customized in different ways.

Restoring the Layout



When you have customized the diagram layout by repositioning probes and targets, you can have the diagram redrawn with the positions and proportions for the layout selected from the **Layouts** drop-down by clicking **Layout all Nodes** in the diagram toolbar. Diagnostics redraws the diagram keeping your zoom and pan settings as you set them.

Restoring Default Diagnostics Resolution



Diagnostics sizes the diagram for the best fit to your browser window when it first displays a diagram. After you have panned and zoomed in the diagram, you can return to the default resolution by clicking **Fit to Contents**. Diagnostics zooms the diagram to the best fit for your browser window and sets the center of the diagram to the center of the window. The custom layout changes that you made to the position of probes and probe connections is not impacted.

Restoring the Diagram to Primary Zoom

Diagnostics sizes the diagram for the best fit to your browser window when it first displays a diagram. There is another default setting that you may want to use when viewing the diagram. This setting sizes the layout to a resolution that displays the diagram in a manner that you may find more useful. You can restore the diagram to this primary zooming resolution by clicking **Reset Zoom**. Diagnostics zooms the diagram to the primary zoom resolution. The custom layout changes that you made to the position of probes and probe connections is not impacted.

1:1

Drilling Down from the Topology View

You can drill down on the entities listed in the graph entity table by using the right-click menu for the listed entity or using the options in the Common Tasks menu.

Drilling Down on Probes

From the Probes tab in the graph entity table, you can drill down to the Server Requests view, the Probe Summary view, the Portal Components view, the Load view, and the Trended Methods view.

Drilling Down on Probe Connections

From the Probe Connections tab in the graph entity table you can drill down to the Probes view.

19

Transactions View

This chapter explains how to work with the Transactions view.

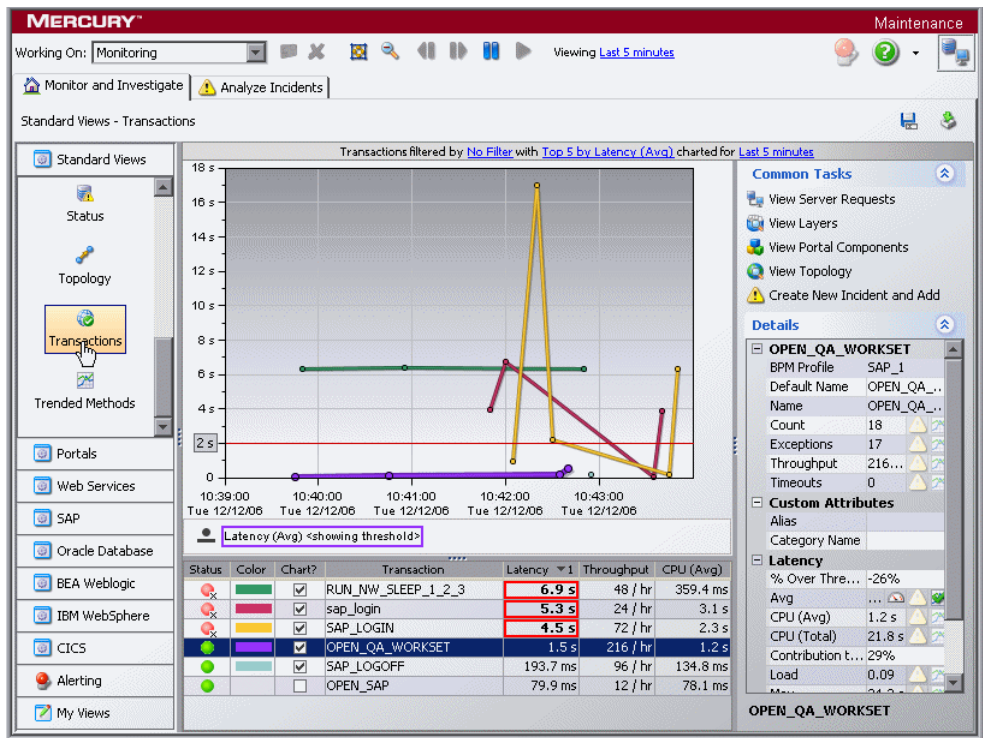
This chapter describes:	On page:
Using the Transactions View	228
Description of the Transactions View	228
Accessing the Transaction View	229
Customizing the Transactions View	230
Interpreting the Transactions View	230
Drilling Down from a Transaction in the Graph Entity Table	232

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 layout view. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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.

When Diagnostics is integrated with Performance Center or LoadRunner, the x-axis of the graph shows the elapsed time since the beginning of the current scenario. In all other cases, the x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the average latency in seconds and milliseconds.

Graph Entity Table

The graph entity 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.


Details Pane

The Details pane in the Transactions view lists the metrics for the selected row of the graph entity table. For more information, see “Working with the Details Pane” on page 57.

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 Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 Graph Entity Table

You can view details for a transaction listed in the graph entity 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	Throughput	CPU (Avg)
x	■	<input checked="" type="checkbox"/>	RUN_NW_SLEEP_1_2_3	6.2 s	60 / hr	34.4 ms
●	■	<input checked="" type="checkbox"/>	sap_login	520.2 ms	120 / hr	317.2 ms
●	■	<input checked="" type="checkbox"/>	SAP_LOGIN		7 / min	213.1 ms
●	■	<input checked="" type="checkbox"/>	OPEN_QA_WORKSET		8 / min	44.9 ms
●	■	<input checked="" type="checkbox"/>	SAP_LOGOFF		6 / min	35.2 ms
●	■	<input type="checkbox"/>	OPEN_SAP	33.7 ms	60 / hr	34.4 ms

Transaction Details

Transaction: RUN_NW_SLEEP_1_2_3

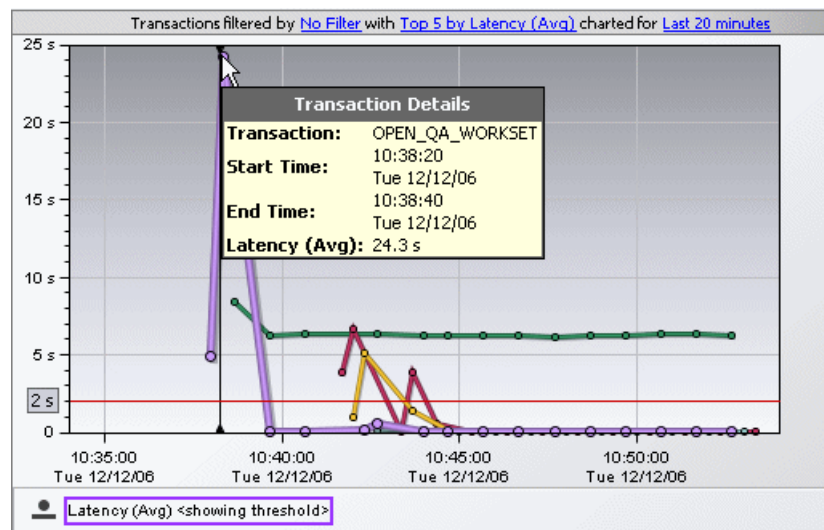
Profile: SAP_1

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 will depend on the metric that is represented by the trend line you selected.

Drilling Down from a Transaction in the Graph Entity Table

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

When you double-click a row in the graph entity 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 graph entity 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 Chapter 16, "Server Requests View."

- ▶ **View Layers**

For information on the Layers view, see Chapter 21, "Layers View."

- ▶ **View Portal Components**

For information on the Portal Components view, see Chapter 14, "Probes View."

- ▶ **View Topology**

For information on the Topology view, see Chapter 18, "Topology View."

- ▶ **Create New Incident and Add**

For information on the Incident Analysis view, see Chapter 8, "Performing Incident Analysis."

20

Trended Methods View

This chapter explains how to work with the Trended Methods view.

This chapter describes:	On page:
Using the Trended Methods View	233
Accessing the Trended Methods View	235
Description of the Trended Methods View	236

Using the Trended Methods View

You can configure specific methods to be displayed as trend lines in the Trended Methods view. The Trended Methods view displays performance metrics for these methods.

This view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

Important: By default, this view is empty. For a method to be displayed in this view, it needs to be configured.

To configure a method to be displayed in the Trended Methods view:

- 1** Open the relevant capture points file.
 - ▶ By default, the J2EE Probe capture points file is located at `<J2EE probe_install_dir>\etc\auto_detect.points`
 - ▶ For the .NET Probe, the capture points files are located in the `<.NET probe_install_dir>\etc\` directory.
- 2** In the capture points file, locate or add the point that includes the method you want to display in the Trended Methods view.
- 3** Add the following argument to the point definition:

```
layertype = trended_method
```

- 4** Restart the application server on which your probe is running.

Important:

- ▶ Configuring methods to be displayed in the Trended Methods view might add significant overhead to the Diagnostics system.
 - ▶ If you configured the probe to capture arguments for a point that is also configured for method trending, each unique argument will result in a unique trended method. This can have a significant impact on memory overhead.
 - ▶ Trended methods are not latency trimmed. This means that any frequently-executing method with low latency could cause a noticeable performance impact on the application.
-

Accessing the Trended Methods View

You can access the Trended Methods view from the Standard Views group on the View bar or by drilling down from the entities of some of the other detail layout views.

Note: The information displayed on the Trended Methods view differs, depending on how you access the view. For more information, see “Description of the Trended Methods View” on page 236

To access the Trended Methods view using the View bar:

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



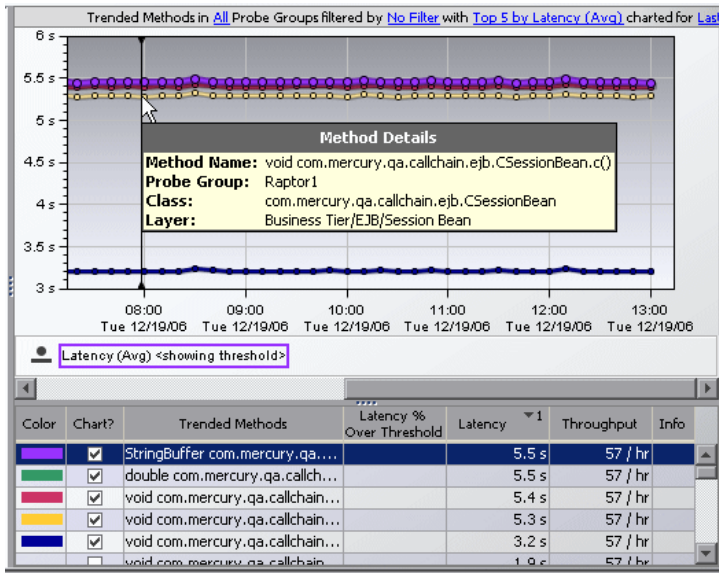
The Trended Methods view is displayed with the default settings so that the five methods that have the highest latency during the selected time range are displayed on the graph.

To access the Trended Methods View from other views:

- Instructions for drilling down to the Trended Methods view for a particular probe in the Probes view can be found in “Drilling Down from a Probe in the Graph Entity Table” on page 180.
- Instructions for drilling down to the Trended Methods view for a particular probe in the Status view can be found in “Drilling Down on a Probe” on page 209
- Instructions for drilling down to the Trended Methods view for a particular probe group in the Status view can be found in “Drilling Down on a Probe Group” on page 208

Description of the Trended Methods View

The following image is an example of the Trended Methods view:



Data

Depending on how you accessed the Trended Methods view, the trended methods displayed in the graph and graph entity table represent different data.

- ▶ When you access the Trended Methods view directly from the view bar, each trended method represents the aggregation of all the times this particular method was executed on different servers within a particular probe group. The actual probe where the method was executed is not identified.
- ▶ When you access the Trended Methods view by drilling down from a probe (see “Accessing the Trended Methods View” on page 235), each trended method represents the aggregation of all the times this method was executed on servers within a particular probe.

Graph

By default, the Trended Methods view graph displays the five methods that have the highest average latency during the selected time range. Diagnostics displays the average latency for each method using a trend line.

When Diagnostics is integrated with Performance Center or LoadRunner, the x-axis of the graph shows the elapsed time since the beginning of the current scenario. In all other cases, the x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the average latency in seconds and milliseconds.

Graph Entity Table

The graph entity table in the Trended Methods view lists all of the methods that pertain to the context shown in the bread crumbs according to the relevant filters defined in the view filters.

Details Pane

The Details pane in the Trended Methods view lists the metrics for the selected row of the graph entity table. For more information, see Chapter 4, “Working with the Details Pane.”

Tooltip

When you hold your mouse pointer over the method in the **Trended Methods** column of the graph entity table, you can view a tooltip that includes information about the selected method. You can view this same information by holding your mouse pointer over one of the points on the trend line for a charted metric.

The information displayed in the tooltip differs, depending on how you accessed the Trended Methods view.

When you access the Trended Methods view directly from the view bar, Diagnostics identifies the probe group, but not the probe, in which the method was executed.

When you access the Trended Methods view by drilling down from a probe (see “Accessing the Trended Methods View” on page 235), Diagnostics identifies both the probe and probe group in which the method was executed.

The tooltip also includes the name of the method and the associated class and layer.

21

Layers View

This chapter explains how to work with the Layers view.

This chapter describes:	On page:
Using the Layers View	240
Description of the Layers View	241
Accessing the Layers View	242
Customizing the Layers View	243
Interpreting the Layers View	243
Drilling Down from a Layer in the Graph Entity Table	245

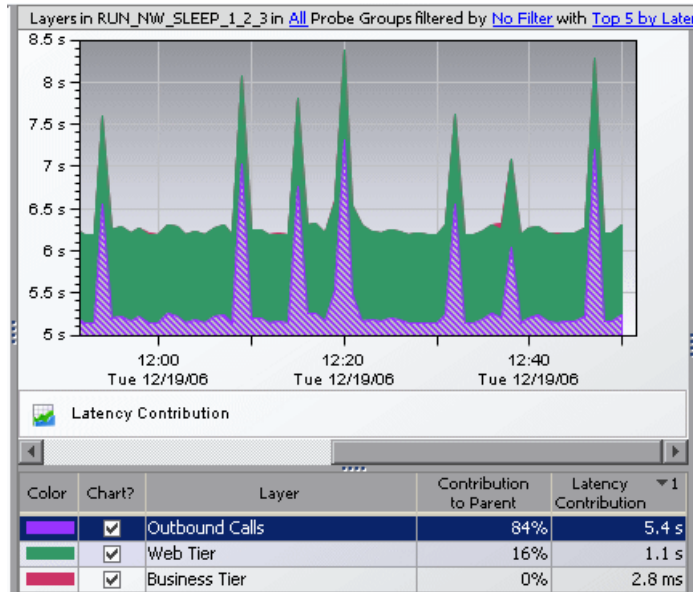
Using the Layers View

The Layers view displays the performance metrics for the 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 layout view. For information about the layout and controls in the detail layout, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 currently selected time frame. Diagnostics displays the latency contribution for each layer using a stacked area graph, as shown in the following example:



Layers View Graph

When Diagnostics is integrated with Load Runner or Performance Center, the x-axis of the graph shows the elapsed time since the beginning of the current scenario. In all other cases, the x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the total latency contribution in seconds.

Graph Entity Table

The graph entity table in the Layers view lists all of the layers that pertain to the context shown in the breadcrumbs 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.

Details Pane

The Details pane in the Layers view lists the metrics for the selected row of the graph entity 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 Graph Entity Table” on page 232.
- ▶ 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 Graph Entity Table” on page 194.

Customizing the Layers View

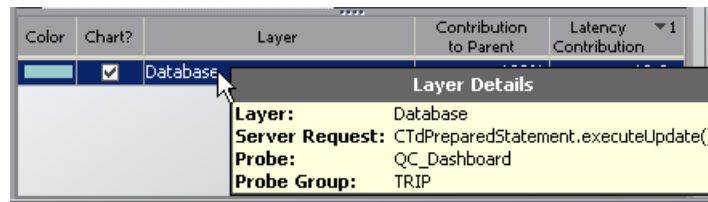
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Layers view. For more information, see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 Graph Entity Table

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

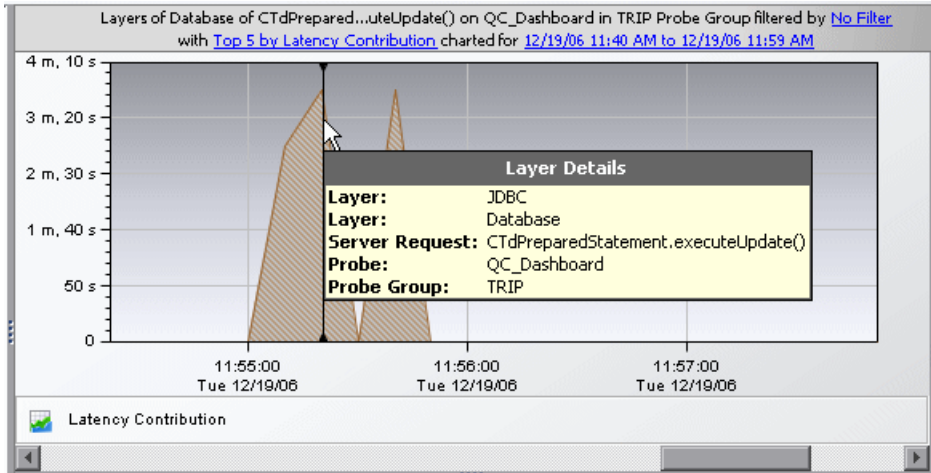


The information displayed in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Layers view. In the example above, the Layers view was accessed by drilling down in the Server Request view. The **Layer Details** tooltip displays the following information:

- ▶ **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 Graph Entity Table

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

When you double-click a row in the graph entity 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 layout view, with a note in the table indicating that no data is available.

When you right-click a row in the graph entity 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.
- ▶ **View Profiler for <probe name>.** This option is available for server request layers. For information on the Mercury Diagnostics Profilers, see Part V, "Using the Mercury Diagnostics Profiler for J2EE" or Part VI, "Using the Mercury Diagnostics Profiler for .NET."
- ▶ **Create New Incident and Add.** For information, see Chapter 8, "Performing Incident Analysis."

22

Call Profile View

This chapter explains how to work with the Call Profile view.

This chapter describes:	On page:
Using the Call Profile View	248
Description of the Call Profile View	249
Accessing the Call Profile View	254
Interpreting the Call Profile View	254
Analyzing J2EE to SAP R/3 Remote Function Calls	256
Analyzing Remote Method Invocation (RMI)	257
Analyzing Life Cycle Methods for Portlets	257
SOAP Fault Call Profiles	257

Using the Call Profile View

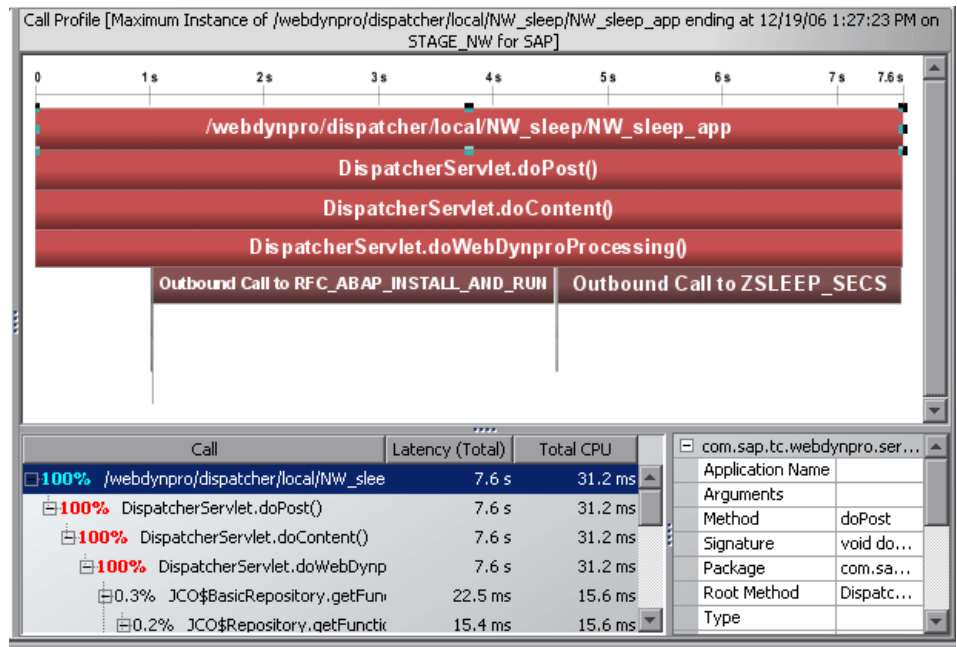
The Call Profile view displays the method calls and their latency for a particular instance of a monitored server request in your application. The Call Profile view displays the metrics in a graphical call profile, and in a table as a call tree. For an explanation of instance trees, see Chapter 9, “Instance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

The Call Profile view shares many features with the other detail layout 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:

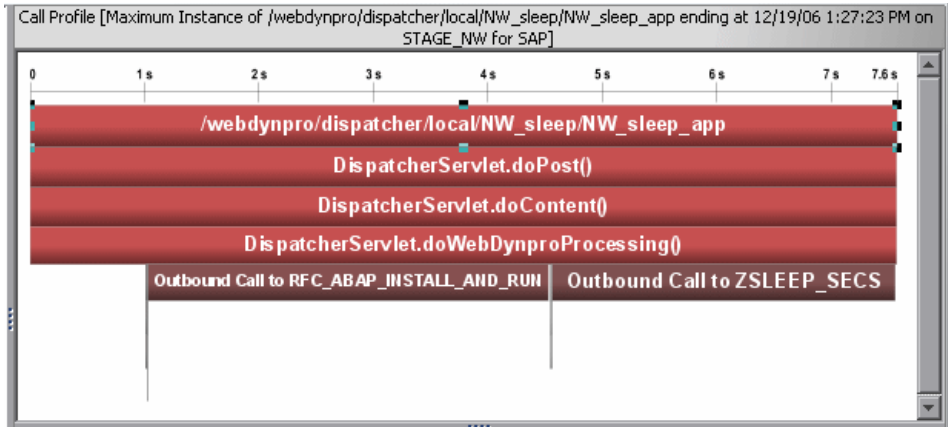


The Call Profile view is made up of three areas in addition to the View bar that is part of all Diagnostics views:

- Call Profile Graph
- Call Tree Table
- Details pane

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.

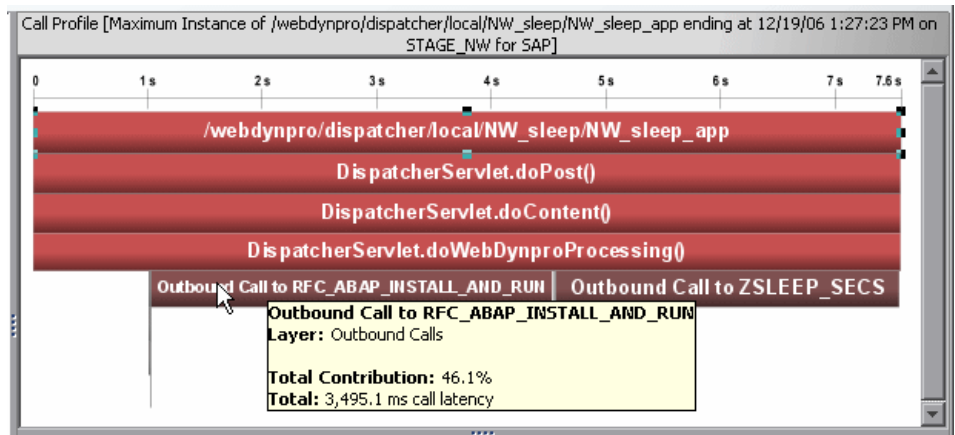
Each box in the instance profile represents a call where the left edge of the box is the start of the method call and the right edge is the return from the call. The length of the box indicates the duration of the call execution. The position of the call box along the horizontal axis indicates the actual 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 Details pane to show the details for the call you clicked.

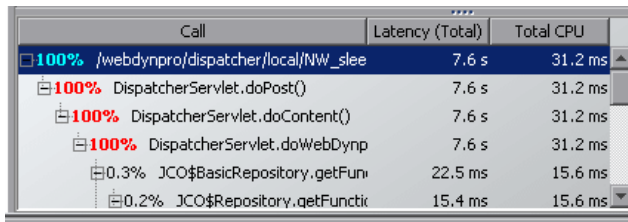


The tooltip displays the following call details:

- 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	Latency (Total)	Total CPU
100% /webdynpro/dispatcher/local/NW_slee	7.6 s	31.2 ms
100% DispatcherServlet.doPost()	7.6 s	31.2 ms
100% DispatcherServlet.doContent()	7.6 s	31.2 ms
100% DispatcherServlet.doWebDynp	7.6 s	31.2 ms
0.3% JCO\$BasicRepository.getFun	22.5 ms	15.6 ms
0.2% JCO\$Repository.getFuncio	15.4 ms	15.6 ms

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.

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 Details pane.

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.
- ▶ **Latency (Total).** The total latency for the call. The time is reported in appropriate units.

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.

- ▶ **Total CPU.** The time spent by the CPU executing this call.

Description of the Details Pane

The Details pane lists the metrics for the call that is selected in the Call Tree table and the Call Profile. For more information about the Details pane, see Chapter 4, “Working with the Details Pane.”

Note: The Details pane in the Call Profile view does not enable setting thresholds, adding to an incident, 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 Requests View, see Chapter 16, “Server Requests View.”

Note: It is possible that the instance tree that is displayed for an instance tree marker on the Server Requests 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 Details pane.

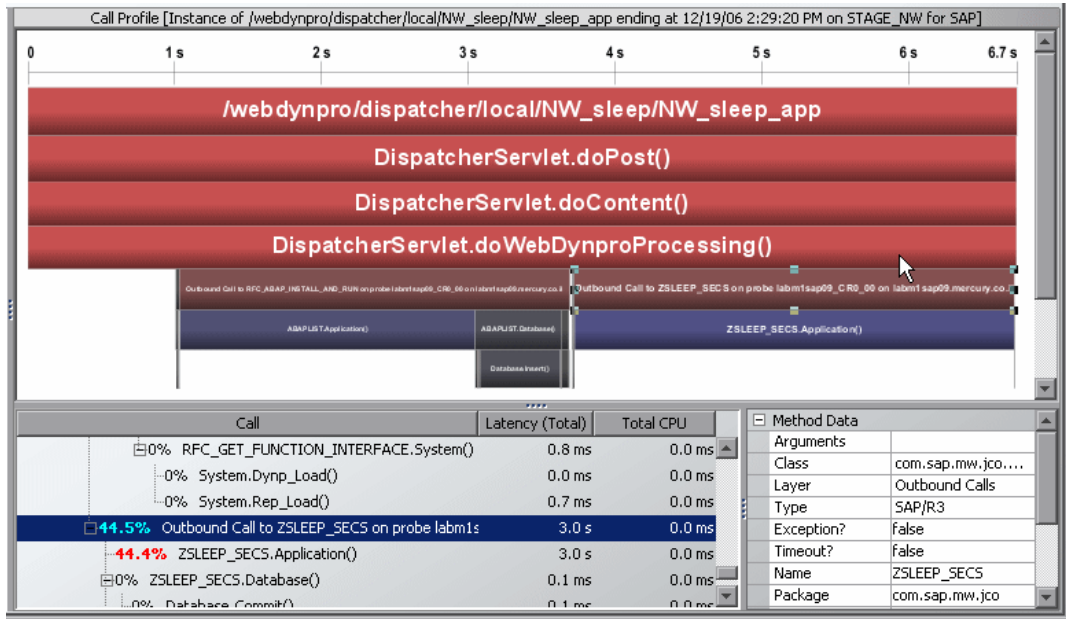
Comparing Call Profile Graphs

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 R/3 Remote Function Calls

The Remote Function Call (RFC) protocol in SAP allows communication to take place between SAP J2EE and SAP R/3 environments. The remote calls that are made between SAP J2EE and SAP R/3 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 Details pane.

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.

SOAP Fault Call Profiles

SOAP is an XML-based messaging protocol used in the exchange of Web services over a network. A SOAP fault is used to carry error information within a SOAP message. Diagnostics captures SOAP faults and classifies them into different types, based on their SOAP fault codes.

Captured SOAP faults are represented in the Inbound Web Services views by a unique icon on the graph. Each SOAP fault icon represents a different type of SOAP fault. When you click on one of these icons, Diagnostics opens a call profile window that includes detailed information about the SOAP fault.

For more information about SOAP fault call profiles, see “SOAP Fault Call Profile” on page 283.

Part IV

Understanding Diagnostics Specialized Views

23

Portal Views

This chapter explains how to work with the Portal views in the Portals view group.

This chapter describes:	On page:
Using the Portal Views	261

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)” on page 262
- “Scenario Summary View (LoadRunner or Performance Center)” on page 263
- “Server Summary View (Standalone mode)” on page 263
- “Portal Components View” on page 264

Business Process Summary View (Mercury Business Availability Center)

If you are using Diagnostics with Mercury Business Availability Center, 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 17, “Status View.”
- **Transactions.** A monitoring version of the standard Diagnostics Transactions view. For more information about the Transactions view, see “Transactions View” on page 227.
- **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 264.

Scenario Summary View (LoadRunner or Performance Center)

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

The Portals Scenario 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 “Transactions View” on page 227.
- ▶ **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. For more information about the Server Requests view, see “Server Requests View” on page 187.
- ▶ **Load.** A monitoring version of the standard Diagnostics Load view. For more information about the Load view, see “Load View” on page 161.
- ▶ **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 264.

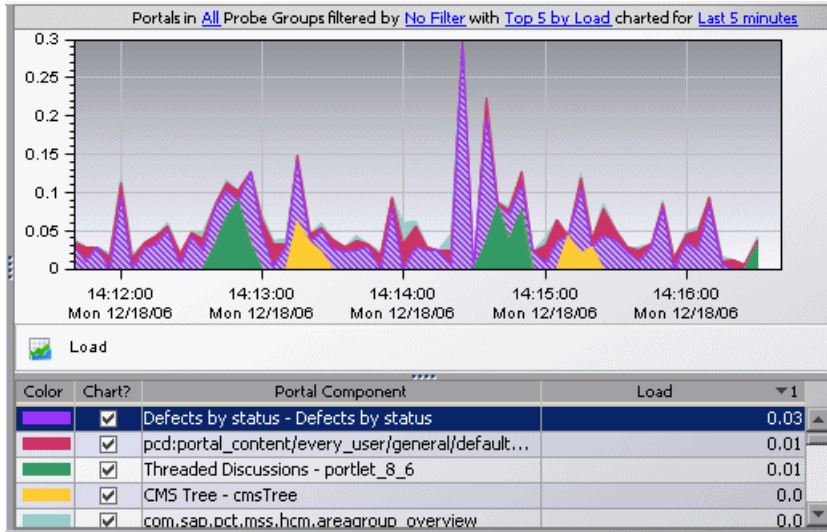
Server Summary View (Standalone mode)

If you are using Diagnostics in standalone mode, the Portals view group contains the Server Summary view. 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 17, “Status View.”
- ▶ **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. For more information about the Server Requests view, see “Server Requests View” on page 187.
- ▶ **Portals.** A monitoring version of the Portal Components view. For more information about the Portal Components view, see “Portal Components View” on page 264.

Portal Components View

The Portal Components view displays load performance metrics for portlets (iViews in SAP) that are monitored by Diagnostics. The following image is an example of the Portal Components view:



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 detailed information about the Portal Components view, see “Portal Components View” on page 265.

24

Portal Components View

This chapter explains how to work with the Portal Components view.

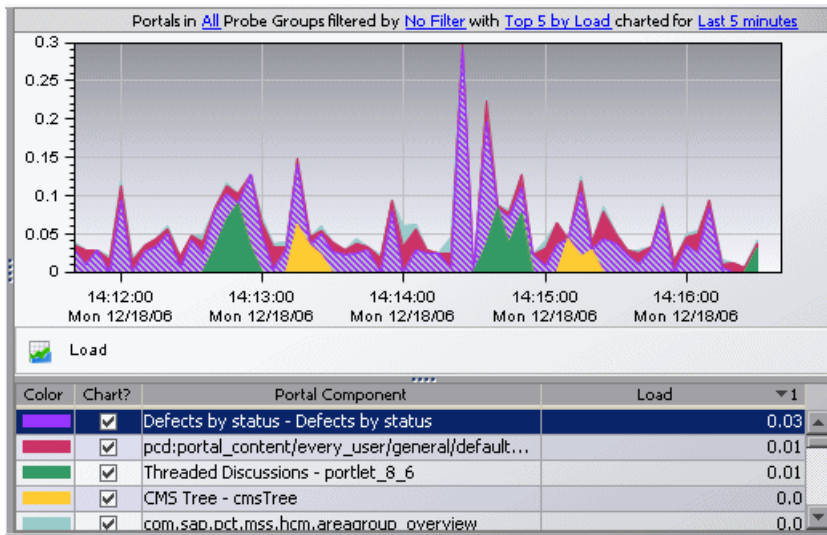
This chapter describes:	On page:
Using the Portal Components View	266
Description of the Portal Components View	266
Accessing the Portal Components View	267
Customizing the Portal Components View	268
Interpreting the Portal Components View	268
Drilling Down from a Portal Component in the Graph Entity Table	269

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 layout view. For information about the layout and controls in the detail layout see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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.

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

Graph Entity Table

The graph entity 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 and the probe group specified in the view filters.

Details Pane

The **Details pane** in the Portal Components view lists the metrics for the selected row of the graph entity 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 **Portals** view group on the View bar.
- 2 Click **Portal Components** in the **Portals** view 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, for example, the **Portals > Server Summary** 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 layout 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 Graph Entity Table” on page 232.

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 Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

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 Graph Entity Table

You can view details for a portal component listed in the graph entity 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 Component	Load
	<input checked="" type="checkbox"/>	Defects by status - Defects by status	0.03
	<input checked="" type="checkbox"/>	Threaded Discussions - portlet_8_6	0.02
	<input checked="" type="checkbox"/>	pcd:portalContent/entry_user/general/default	0.01
	<input checked="" type="checkbox"/>	CMS Tree	0.01
	<input checked="" type="checkbox"/>	com.sap.p	0.0
	<input type="checkbox"/>	Mv_Task_I	0.0

Layer Details	
Layer:	Threaded Discussions - portlet_8_6
Probe Group:	Raptor1

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 Graph Entity Table

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

When you double-click a row in the graph entity 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 graph entity table and selecting **View Life Cycle Methods**.

25

Web Services Views

This chapter explains how to work with the Web Services views in the Web Services view group.

This chapter describes:	On page:
About Web Services Diagnostics Views	272
Using the Web Services Views	273
Viewing Web Services Correlations	279
Viewing SOAP Faults	281

About Web Services Diagnostics 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 service calls made from within your monitored environment, and *inbound* Web service calls received and processed in your monitored environment.

For example, a consumer requests an exchange rate from a currency conversion Web service. From the consumer'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 consumer is defined as an inbound call.

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

Supported Platforms

Diagnostics supports Web services tracking for the following application servers:

- ▶ BEA WebLogic 8.1
- ▶ BEA WebLogic 9.0
- ▶ BEA WebLogic 9.2 (No support for SOAP fault capturing)
- ▶ IBM WAS 5.1
- ▶ IBM WAS 6.0
- ▶ SAP NetWeaver 6.4 (No support for SOAP fault capturing)
- ▶ MS .NET 1.1
- ▶ MS .NET 2.0
- ▶ Apache Axis 1.4 (No support for SOAP fault capturing)
- ▶ Apache SOAP 2.3.1 (No support for SOAP fault capturing)

Using the Web Services Views

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

The Web Services view group includes the following views:

- ▶ **Inbound/Provider.** Displays inbound Web service calls. For more information, see “Viewing Inbound Web Service Calls” on page 274.
- ▶ **Inbound/Provider by Consumer IP.** Displays inbound Web service calls according to the Consumer IP from which they originated. For more information, see “Viewing Inbound Web Service Calls” on page 274.
- ▶ **Outbound/Consumer.** Displays outbound Web service calls. For more information, see “Viewing Outbound Web Service Calls” on page 278.

In the Web service views, Web service calls are displayed in the **Web Service & Operation Name** column of the graph entity table. They appear in the following format: **<Web-service-name>::<operation-name>**.

For example, CurrencyConversionService::ConvertToNum.

Note: If there is an instrumentation failure in the capturing of Web services by the probe, the Web service name and/or operation may appear as **Unknown**.

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.

Viewing Inbound Web Service Calls

Mercury Diagnostics displays information about the inbound Web service calls received and processed in your monitored environment in the **Inbound/Provider** and **Inbound/Provider by Consumer** views.

This section describes:

- “Inbound/Provider” on page 274
- “Inbound/Provider by Consumer IP” on page 275
- “Working with the Inbound Web Services Views” on page 277

Inbound/Provider

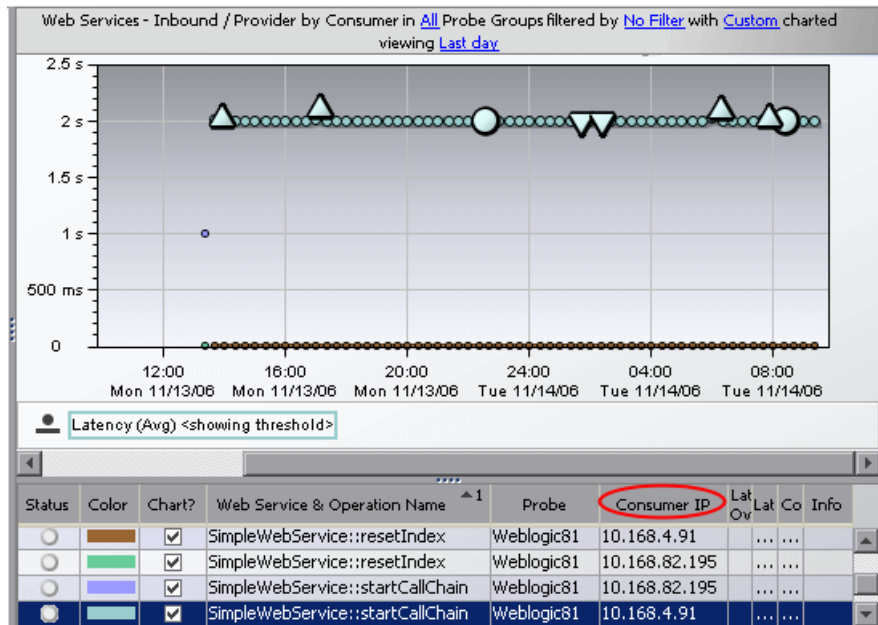
This view displays all inbound Web service operations. Each unique Web service operation is displayed in the graph entity table.

If the same Web service operation was called by multiple consumer IPs, the data from all these calls is aggregated and presented as a single Web service operation in the graph entity table.

Inbound/Provider by Consumer IP

This view displays inbound Web service operation calls according to the consumer IP from which they originated.

If the same Web service operation was called by multiple consumer IPs, the Web service operation is displayed in separate rows of the graph entity table for each unique consumer IP. The actual consumer IP is displayed in the **Consumer IP** column.



In the above example of a Diagnostics screen, the Web service operation, SimpleWebService::resetIndex, is called from two different Consumer IPs.

Diagnostics monitors and displays a limited number of unique consumer IPs per probe. Assume for example that the probe was configured to display five unique consumer IPs. In this case, if there are more than five consumer IPs, the first five are displayed on separate rows in the graph entity table. The remaining consumer IPs are aggregated into a single row in the table with the consumer IP label **other**.

You can define the number of consumer IPs that are monitored and displayed per probe. This is done differently for the .NET Probe and the J2EE Probe.

To change the number of consumer IPs monitored by the .NET Probe:

- 1 Go to the `<diag_server_install_dir>\etc\server.properties` file on the mediating server for the .NET probe.
- 2 Enter the relevant number in the following line:

```
max.tracked.addresses.per.probe=<number>
```

- 3 Restart the Diagnostics Server for the change to take effect.

To change the number of consumer IPs monitored by the J2EE Probe:

- 1 Go to the `<probe_install_dir>\etc\dispatcher.properties` file.
- 2 Enter the relevant number in the following line:

```
max.tracked.addresses.per.probe=<number>
```

- 3 Restart the J2EE Probe for the change to take effect.

Note: Diagnostics creates samples for each consumer IP that calls a Web service. These samples are sent to Mercury Business Availability Center when the two products are integrated.

Increasing the number of consumer IPs captured by Diagnostics will therefore increase the amount of data sent to, and processed by, Mercury Business Availability Center. This will also increase the amount of CPU/Memory/Diskspace that by the Diagnostics Servers and probes.

You can stop Diagnostics creating these samples by going to the `<diag_server_install_dir>\etc\server.properties` file on the mediating server and changing the value of the following line to **false**:

```
bac.webservice.create.samples = true.
```

You can view the list of consumer IPs that are tracked by the probe in the following files:

- ▶ For the J2EE Probe, you view the list in the `<probe_install_dir>/log/<probe_name>_addr.properties` file
- ▶ For the .NET Probe, you view the list in the `<mediating_server_install_dir>/etc/consumers/<probe_name>_addr.properties` files.

These files contains comma separated lists of consumer IPs tracked by the probe or server. If you know the address of a particular consumer IP that you want Diagnostics to monitor, you can add the name of that consumer IP to the relevant file.

For the J2EE Probe, you need to restart the probe for the tracking of the manually added consumer IP to take effect. For the .NET Probe, you need to restart the Diagnostics Server for the tracking to take effect.

Working with the Inbound Web Services Views

In the inbound Web services views, inbound Web service calls are displayed as a type of server request.

You can view information about SOAP faults and drill down to a call profile window for a particular SOAP fault. For more information, see “Viewing SOAP Faults” on page 281.

You can drill down to instance trees that display the method calls and their latencies for a particular instance of a server request. For more information, see “Drilling Down to an Instance Tree for a Server Request” on page 195.

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

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

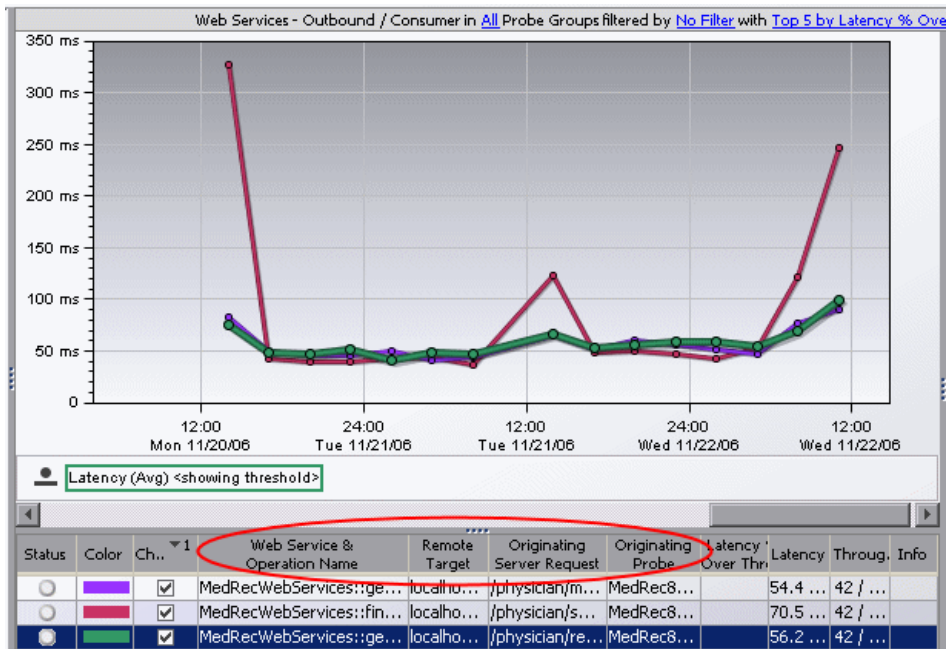
Changing the Display Name of a Web Service

Diagnostics and Mercury Business Availability Center use different ways of identifying the names of Web Services. As a result, there may be certain cases where Diagnostics and Mercury Business Availability Center identify different names for the same Web Service.

In such cases, you can change the Web Service name as it is displayed in Diagnostics to match the name used in Mercury Business Availability Center. You use the **Web Service Alias** field in the Details pane to change the Web service name. For detailed instructions about changing the Web service name, see “Setting Custom Attributes Values” on page 60.

Viewing Outbound Web Service Calls

Information about the outbound Web service calls made from within your monitored environment is displayed in the Outbound/Consumer view.



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

The Outbound Calls graph entity table includes the following columns:

- ▶ **Web Service & Operation Name.** 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`.
- ▶ **Remote Target.** The hostname and port number (displayed as `<host:port>`) of 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 originating server request displayed in the Server Requests view by right-clicking the outbound call in the graph entity table and selecting **View Server Requests**.

Viewing Web Services Correlations

If both the outbound and inbound Web service 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 X on the Server Requests view containing the outbound Web service 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 server request, displayed at the top of the diagram and table, contains an outbound Web service call. The inbound call for the same Web service is displayed in blue. .

Web Services - Outbound / Server Requests
 Consumer > (/WS_Client/WeblogicWebServiceClient.jsp) > Call Profile
 (SimpleWebService::startCallChain) sp)

Call Profile [Instance of /WS_Client/WeblogicWebServiceClient.jsp ending at 11/8/06 1:59:03 AM on Weblogic81 for Default]

1 2.1s

/WS_Client/WeblogicWebServiceClient.jsp
 JspBase.service()
 Outbound Call to SimpleWebService::startCallChain on probe Weblogic81 on blazer.mercury.co.i
 WebServiceServlet.doPost()
 ServletBase.doPost()
 Dispatcher.process()

Call	Latency (Total)	Total CPU
100% /WS_Client/WeblogicWebServiceClient.jsp	2.1 s	31.2 ms
100% JspBase.service()	2.1 s	31.2 ms
0.4% Outbound Call to SimpleWebService::startCallChain on probe Weblogic81 on blazer.mercury.co.i	7.6 ms	0.0 ms
0.1% WebServiceServlet.doPost()	1.2 ms	0.0 ms
0.1% ServletBase.doPost()	1.2 ms	
0% Dispatcher.process()	0.3 ms	
97.4% Outbound Call to SimpleWebService::startCallChain on probe Weblogic81 on blazer.mercury.co.i	2.0 s	0.0 ms
97.3% WebServiceServlet.doPost()	2.0 s	
97.3% ServletBase.doPost()	2.0 s	
97.2% Dispatcher.process()	2.0 s	

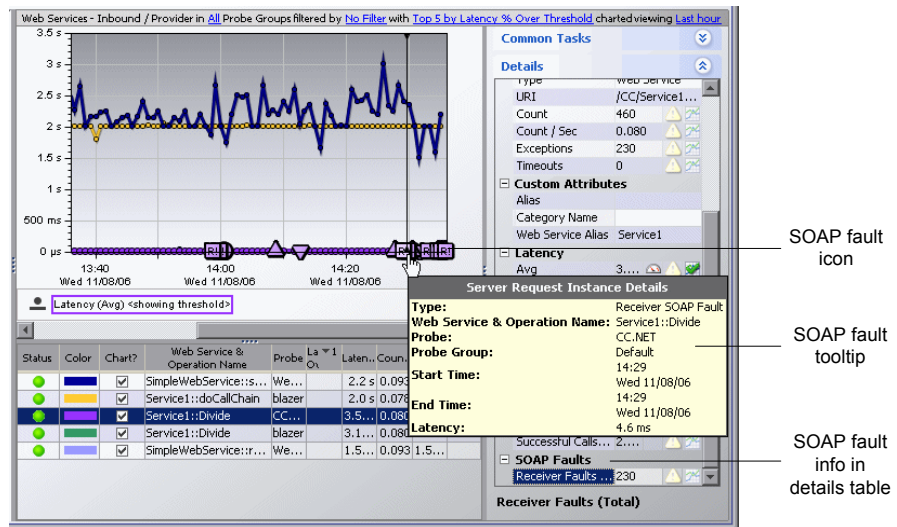
Method Data

Arguments	
Class	weblogi... ..
Layer	Outbound ...
Type	Web Service
Timeout?	false
Exception?	false
Name	SimpleWebS...
Package	weblogic.we...
Return T...	Object
Latency	
Latency ...	2,003.6 ms
CPU (To...	0.0 μs

Viewing SOAP Faults

SOAP is an XML-based messaging protocol used in the exchange of Web services over a network. A SOAP fault is used to carry error information within a SOAP message.

Diagnostics captures SOAP faults and classifies them into different types, based on their SOAP fault codes. Captured SOAP faults are represented in the Inbound Web Services views by a unique icon on the graph.








Each SOAP fault icon represents a different type of SOAP fault. When you click on one of these icons, Diagnostics opens a call profile window that includes detailed information about the SOAP fault.

This section includes:

- ▶ “Types of SOAP Faults” on page 282
- ▶ “SOAP Fault Tooltip” on page 282
- ▶ “SOAP Fault Information in Details Pane” on page 282
- ▶ “SOAP Fault Call Profile” on page 283

Types of SOAP Faults

The following table describes each type of SOAP fault and indicates the icon used to represent it in the inbound Web Services views:

Icon	SOAP Fault	Description
	VersionMismatch	The processing party found an invalid namespace for the SOAP Envelope element.
	MustUnderstand	An immediate child element of the SOAP Header was not understood.
	DataEncodingUnknown	SOAP header block or SOAP body child element uses unsupported data encoding.
	Sender	The message was incorrectly formed or did not contain the appropriate information in order to succeed. SOAP 1.1 Client faults are displayed in Diagnostics as Sender faults.
	Receiver	The message could not be processed for reasons attributable to the processing of the message rather than to the contents of the message itself. SOAP 1.1 Server faults are displayed in Diagnostics as Receiver faults.

SOAP Fault Tooltip

When you hold the mouse pointer over one of the SOAP fault icons on the graph, a tooltip identifies the type of SOAP fault, along with general information about the selected server request instance.

SOAP Fault Information in Details Pane

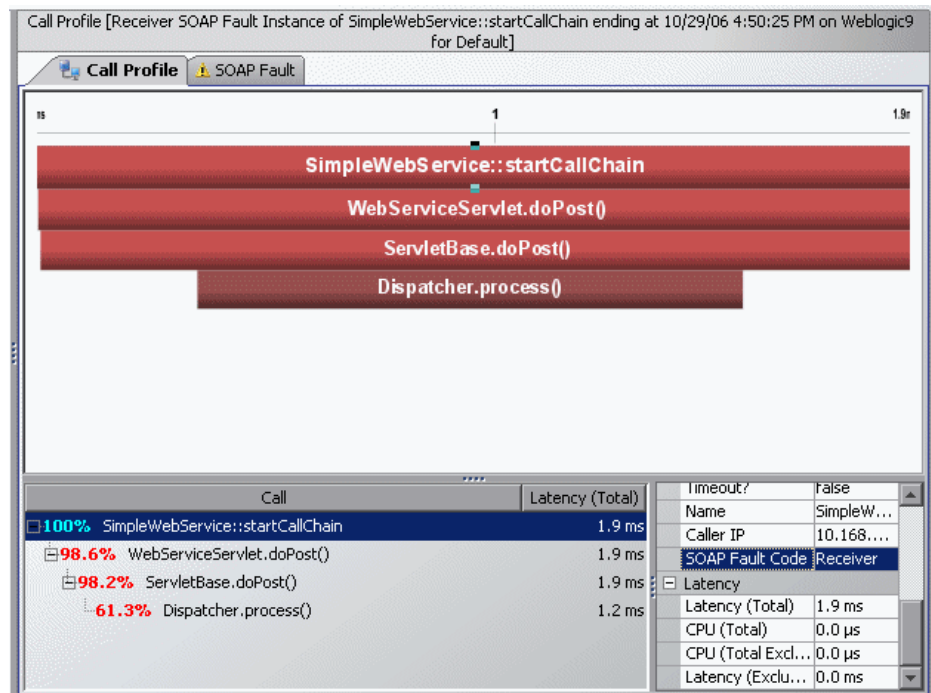
When you select a Web service for which one more SOAP faults have occurred, in the graph entity table, Diagnostics displays the total SOAP faults in the Details pane, under the **SOAP Faults** category.

In the **Latency** category of the Details pane, you can view the average latency of all the calls for the selected Web service (**Avg** metric), or you can view the average latency of only the successful calls—those that did not generate a SOAP fault (**Successful Calls (Avg)** metric).

Note: Web service data sent to Mercury Business Availability Center includes successful Web service calls only.

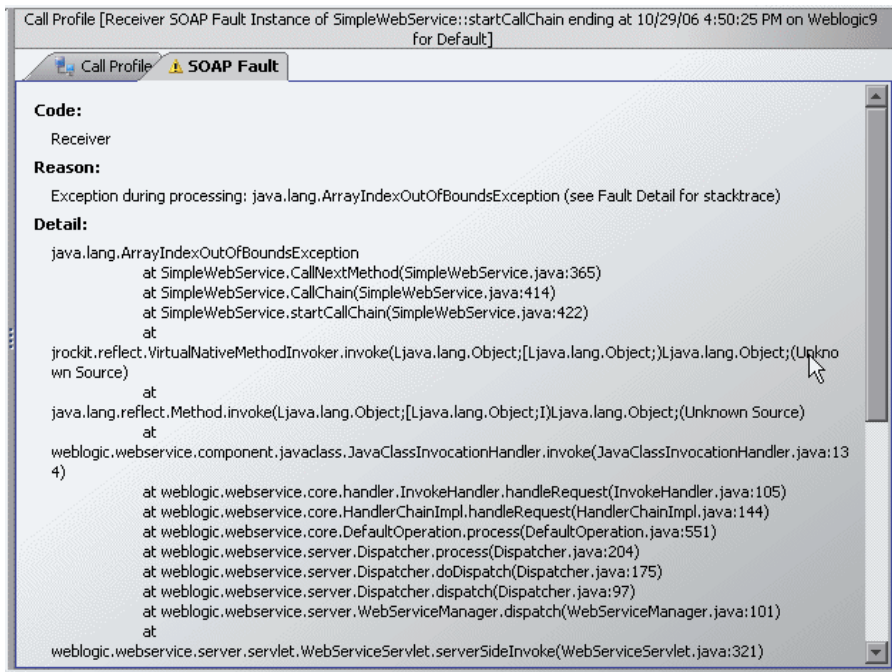
SOAP Fault Call Profile

When you click a SOAP fault icon, Diagnostics displays a call profile of the particular Web service call that generated the SOAP fault and an additional tab containing detailed information about the SOAP fault.



The heading of the Call Profile includes the type of SOAP fault that was generated. When you select the Web service in the first line of the call tree table or the call profile graph, the SOAP fault code is displayed in the Details pane.

When you click the **SOAP Fault** tab in the Call Profile window, you can view detailed information about the SOAP fault.



The SOAP fault tab in the Call Profile window includes the following information:

- ▶ **Code.** The type of SOAP fault.
- ▶ **Reason.** The reason that the SOAP fault was generated.
- ▶ **Detail.** Either the exception stack trace or other detailed information contained in the generated SOAP Fault.

26

SAP Views

This chapter explains how to work with the SAP views in the SAP view group.

This chapter describes:	On page:
Using the SAP Views	285

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” on page 286
- ▶ “R3 Summary” on page 287
- ▶ “NetWeaver Requests” on page 288
- ▶ “NetWeaver Threads” on page 289
- ▶ “R3 Probes” on page 290
- ▶ “R3 Server Requests” on page 291

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 17, “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 “Server Requests View” on page 187.
- ▶ **Portals.** A monitoring version of the standard Diagnostics Portal Components view. For more information about the Portal Components view, see Chapter 24, “Portal Components View.”
- ▶ **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 “Probes View” on page 173.
- ▶ **NetWeaver Threads.** A monitoring version of the NetWeaver Threads view. For more information, see “NetWeaver Threads” on page 289.
- ▶ **NetWeaver Requests.** monitoring version of the NetWeaver Requests view. For more information, see “NetWeaver Requests” on page 288.

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 17, “Status View.”

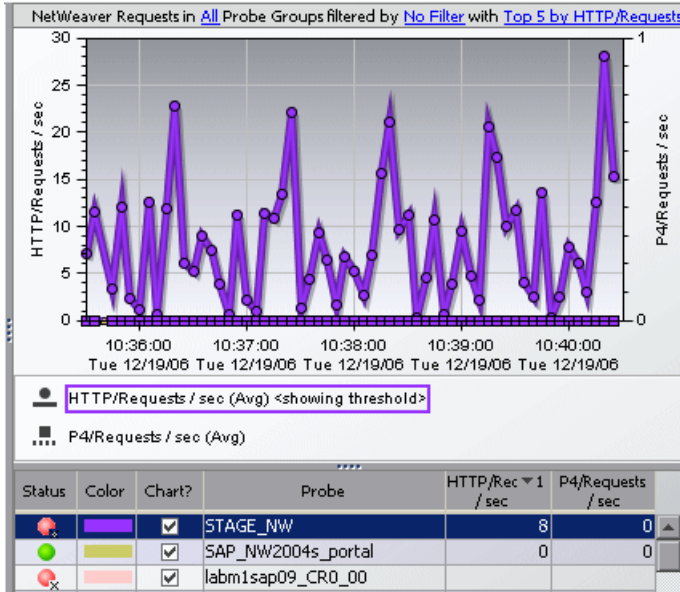
The only metrics in the table that are relevant for the SAP R/3 Probes are the **Latency** and **Count** metrics.

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 290.
- ▶ **SAP R/3 Server Requests.** A monitoring version of the R3 Server Requests view. For more information, see “R3 Server Requests” on page 291.

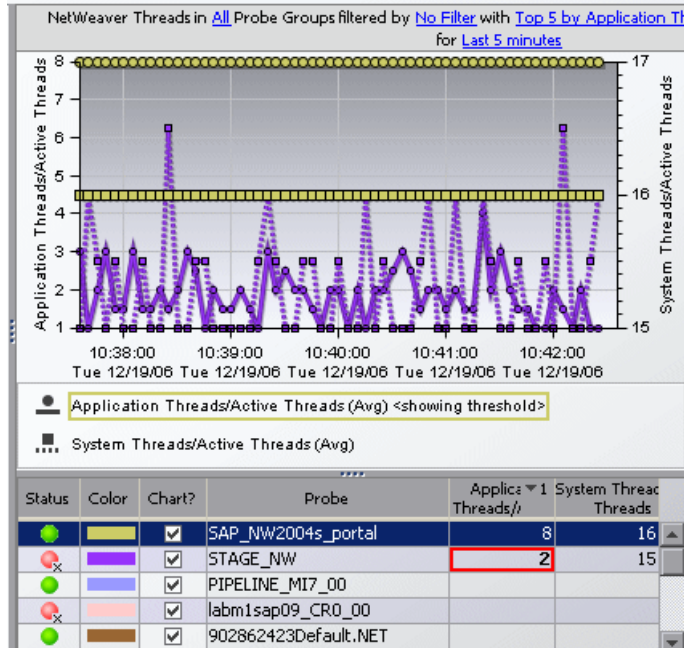
NetWeaver Requests

The NetWeaver Requests view displays the average HTTP and P4 requests per second running on the SAP Web Application Server Java stack.



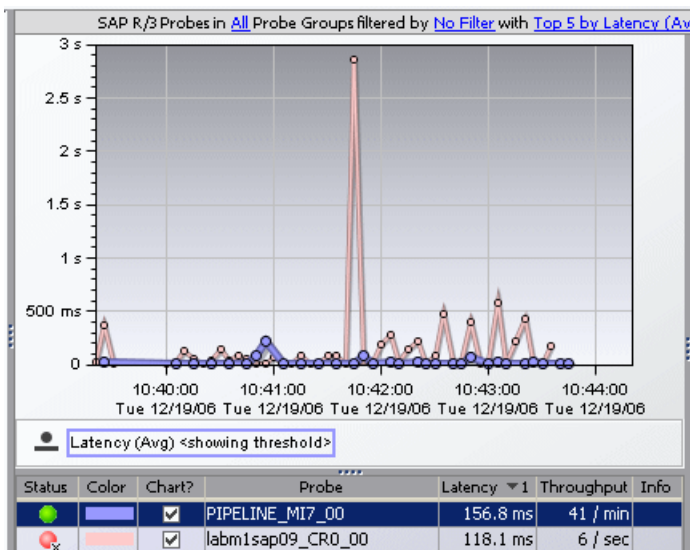
NetWeaver Threads

The NetWeaver Threads view displays information about the active threads running on the SAP Web Application Server Java stack.



R3 Probes

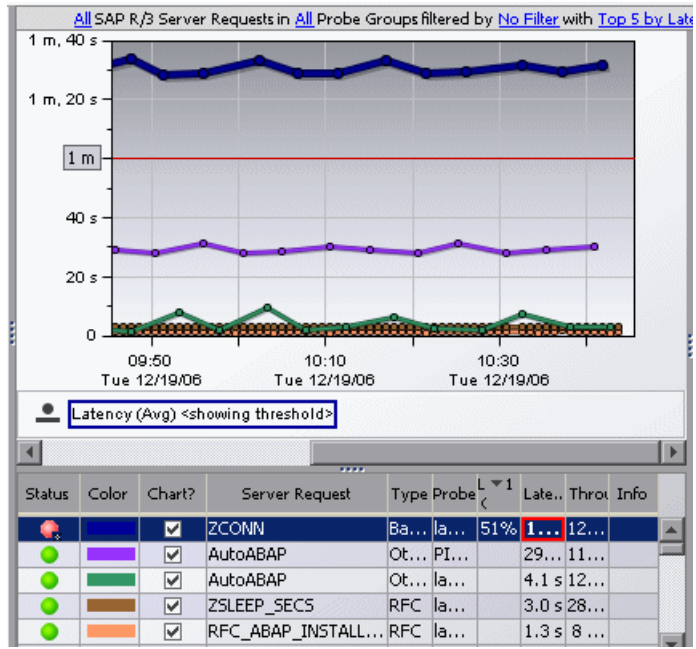
The R3 Probes view displays unique metrics for the SAP R/3 instances.



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.



27

Oracle Database Views

This chapter explains how to work with the Oracle Database views in the Oracle Database view group.

This chapter describes:	On page:
Using the Oracle Views	293

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 Database view group.

To access the Oracle views:

Click **Oracle Database** 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” on page 294
- ▶ “Oracle Probes View” on page 294
- ▶ “Oracle Wait Time View” on page 295

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” on page 294.
- ▶ **Wait Time.** A monitoring version of the Oracle Wait Time view. For more information, see “Oracle Wait Time View” on page 295.

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.

28

BEA WebLogic Views

This chapter explains how to work with the BEA WebLogic views in the BEA WebLogic view group.

This chapter describes:	On page:
Using the BEA WebLogic Views	297

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 17, “Status View.” For more information about the Server Requests view, see “Server Requests View” on page 187.

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 JDBC requests waiting for a connection.
- ▶ **JDBC Wait Seconds High.** Number of seconds the longest request had to wait for a connection.

Server Threads

This view displays the following default metrics:

- ▶ **Execute Queues Requests.** Number of requests, which have been processed by the application server's default execute queue.
- ▶ **Execute Queues Pending Requests.** Number of requests waiting in the application server's default execute queue.

29

IBM WebSphere Views

This chapter explains how to work with the IBM WebSphere views in the IBM WebSphere view group.

This chapter describes:	On page:
Using the IBM WebSphere Views	299

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 17, “Status View.” For more information about the Server Requests view, see “Server Requests View” on page 187.

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.

30

CICS Views

This chapter explains how to work with the CICS views in the CICS view group.

This chapter describes:	On page:
Using the CICS Views	303

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 “Server Requests View” on page 187.

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.

Part V

Using the Mercury Diagnostics Profiler for J2EE

31

Using the J2EE Diagnostics Profiler

This chapter provides an overview of the Mercury Diagnostics Profiler for J2EE, descriptions of the tabs, and instructions for using the controls that are common to the tabs of the user interface.

This chapter describes:	On page:
Accessing the Mercury Diagnostics Profiler for J2EE	308
Mercury Diagnostics Profiler for J2EE Processing	309
Common J2EE Diagnostics Profiler Tab Navigation and Display Controls	311

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 with 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 views of the Mercury Diagnostics user interface.

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 **View Profiler for <probe name>** from the menu.
- ▶ Alternatively, in the standard Probes view in Mercury Diagnostics, right-click the probe entity in the graph entity table and select **View Profiler for <probe name>** 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 all call instances for a type of service request and calculates the aggregated latency

When a server request instance is encountered that is slower than one of the captured instances for the server request, the slower instance 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 method instances that are included in the captured call stacks.

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 32, “Analyzing Method Latency with J2EE Diagnostics Profiler Tabs.”

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

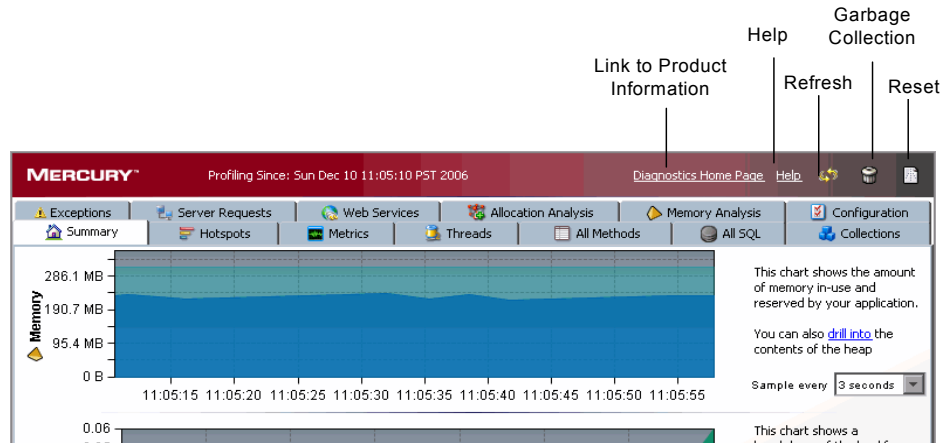
Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you to identify objects that may be leaking. By default, Light Weight Memory Diagnostics and Heap Breakdown are disabled.

For more information about Light Weight Memory Diagnostics, see “Analyzing Memory Using the Collections Tab” on page 349.

For more information on Heap Breakdown, see “Analyzing Memory Using the Heap Breakdown Tab” on page 359.

Common J2EE Diagnostics Profiler Tab Navigation and Display Controls

This section describes the features and controls that are common to the tabs of the J2EE Diagnostics Profiler:



Refreshing Metrics



When you are ready to view more current performance metrics, click **Refresh** on the top right corner of the screen to refresh the information displayed. The Profiler is refreshed 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 **Reset**.

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 of the probed application by clicking **Garbage Collection** 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.

32

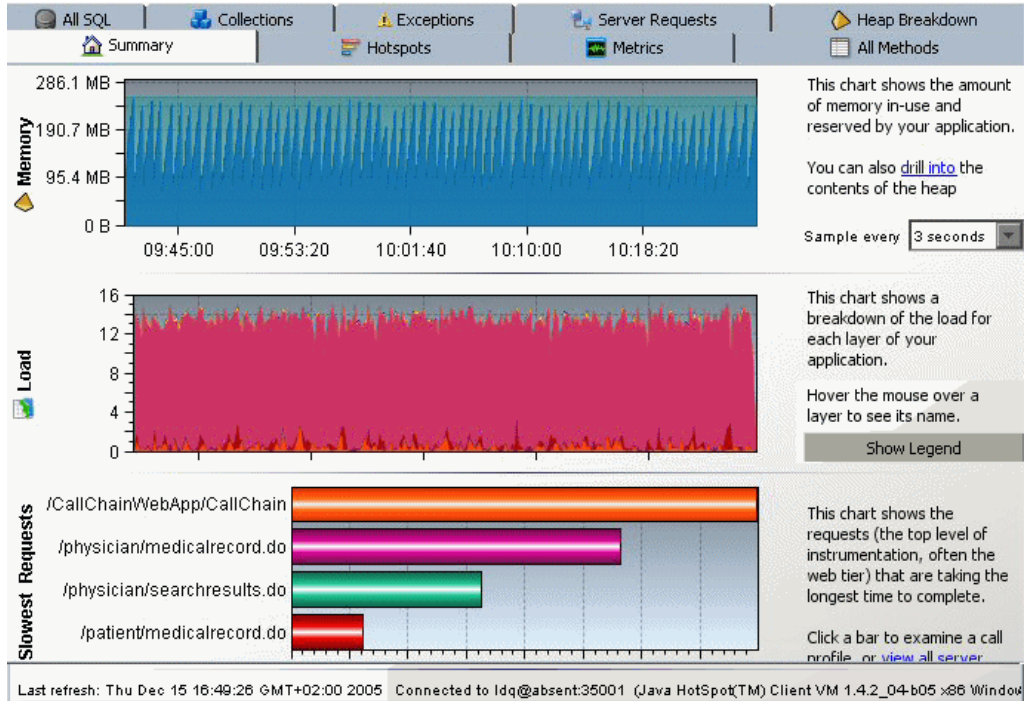
Analyzing Method Latency with J2EE Diagnostics Profiler Tabs

This chapter provides a detailed description of the tabs, 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	314
Analyzing Performance Using the Hotspots Tab	318
Analyzing Performance Using the Metrics Tab	320
Analyzing Performance Using the Threads Tab	322
Analyzing Performance Using the All Methods Tab	328
Analyzing Performance Using the All SQL Tab	331
Analyzing Performance Using the Exceptions Tab	333
Analyzing Performance Using the Server Requests Tab	335
Analyzing Performance Using the Call Profile Window	338
Analyzing Performance Using the Web Services Tab	345
Using the Configuration Tab	347

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.

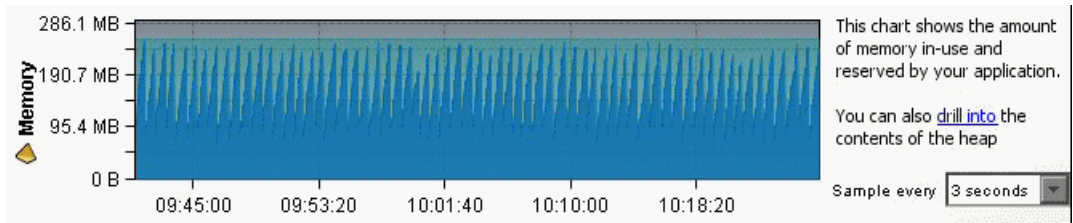


Understanding the Summary tab

The Summary tab is divided into the following sections:

Memory Graph

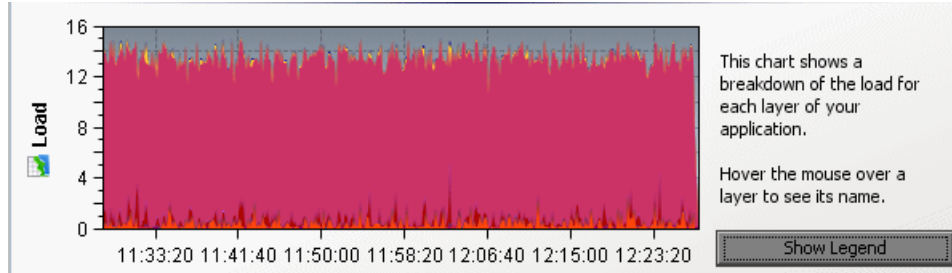
The Memory graph shows the amount of memory allocated in your application and the amount of memory (JVM heap size) reserved by your application.



You can see more details about the exact amount of allocated memory or reserved memory in your application, by holding your pointer over various points on the graph to view the tooltip.

Load Graph

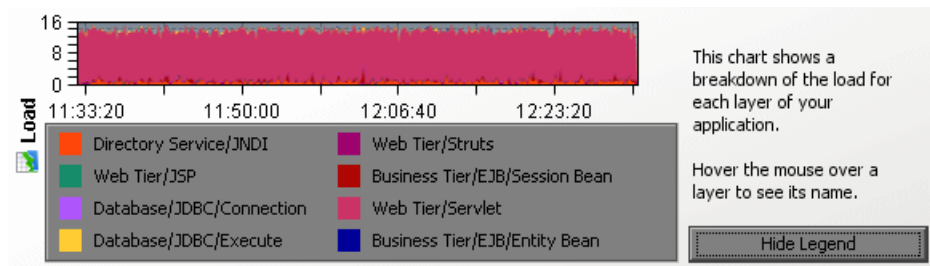
The Load graph shows the breakdown of the load for each layer of your application.



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.

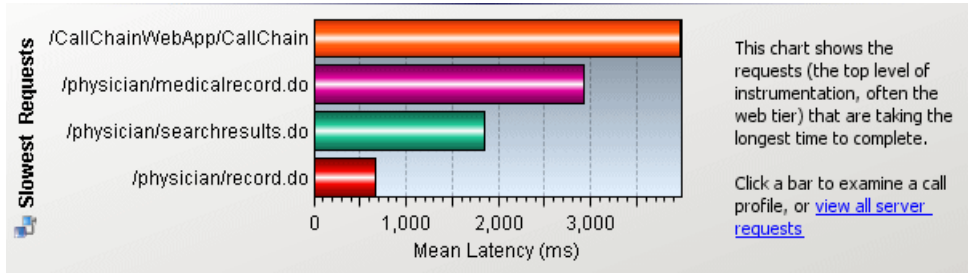
You can see the name of each layer by holding your pointer over various points on the graph to view the tooltip.

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 bar for the server request. For more information about the call profile window, see “Analyzing Performance Using the Hotspots Tab” on page 318.

Information Pane

The information pane at the bottom of the window displays the following information:

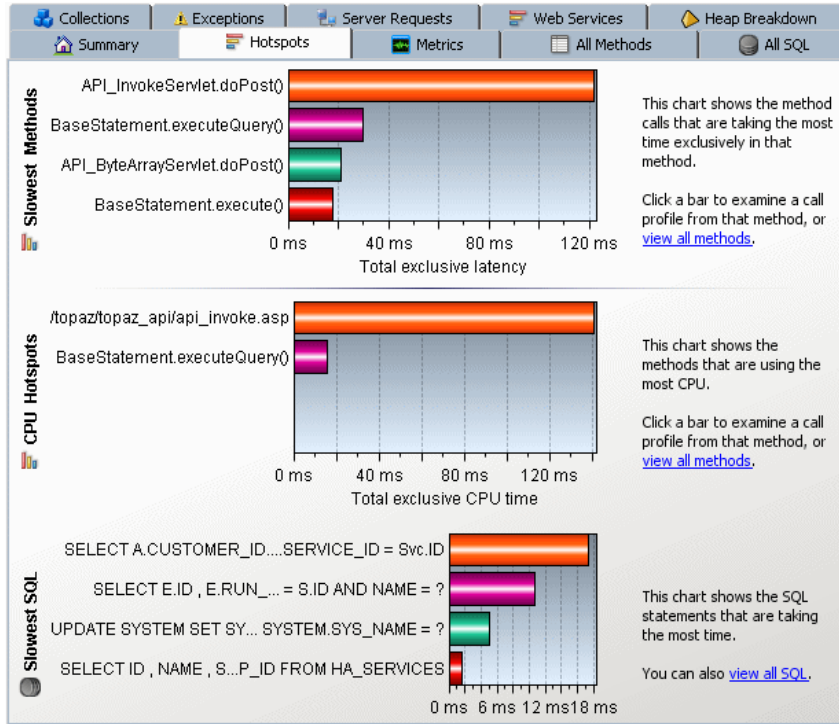
- The date and time of the last time you refreshed the Profiler data.
- The JVM details.
- The probe ID.

Last refresh: Fri Dec 01 10:35:50 PST 2006

Probe ID: WL91MedRe

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 for a graphed metric by holding your pointer over the bar for the metric and viewing the tooltip.

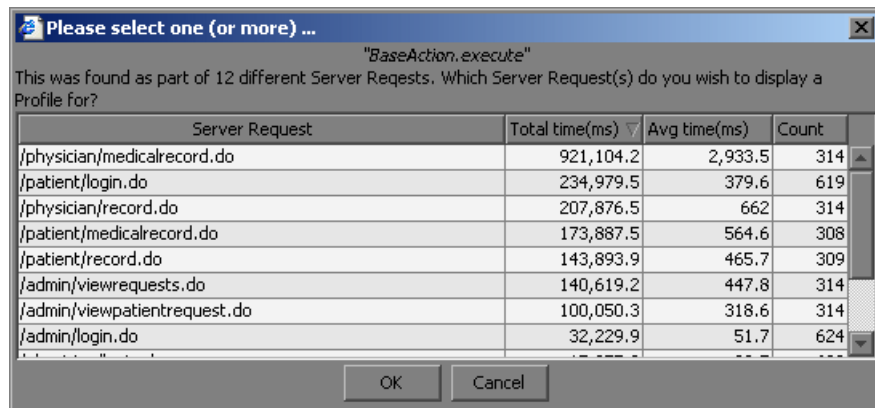
Understanding the Hotspots Tab

The Hotspots tab contains the following 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 selected method call in the Slowest Methods graph, click the bar for the method. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 338.

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 particular server request for which you want to see the call profile.



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 bar for the method. For more information about the call profile window, see “Analyzing Performance Using the Call Profile Window” on page 338.

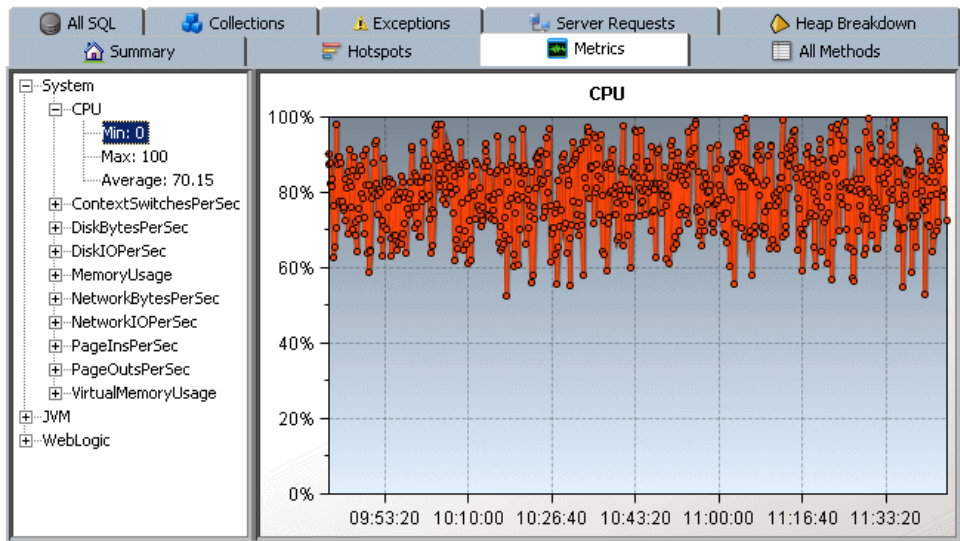
Slowest SQL Graph

This chart shows the SQL statements that are taking the most time.

To view the SQL statement details for a particular statement in the Slowest SQL graph, click the bar for the SQL statement to select it. For more information about SQL statement details, see “Analyzing Performance Using the All SQL Tab” on page 331.

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 metrics selected 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. Depending on the environment, 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 the probe for which you opened the profiler.

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** or **Shift** keys.

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 working with another Mercury product, in which case they are displayed for three hours.

Analyzing Performance Using the Threads Tab

The **Threads** tab displays thread performance metrics for the Java threads that are captured by the probe and provides a way for you to capture stack traces for the captured threads.

This tool can be useful for helping to diagnose the following situations:

- ▶ Incorrect thread pooling or attempting to do too much in a single thread.
- ▶ Performance problems caused by deadlocks or concurrency-related issues.
- ▶ Problems that go deep into the interactions with the OS kernel where you need to see the CPU time broken into user and kernel times.

Note: This **Threads** tab is only displayed when the application is running on JDK 1.5 and above.

This section includes:

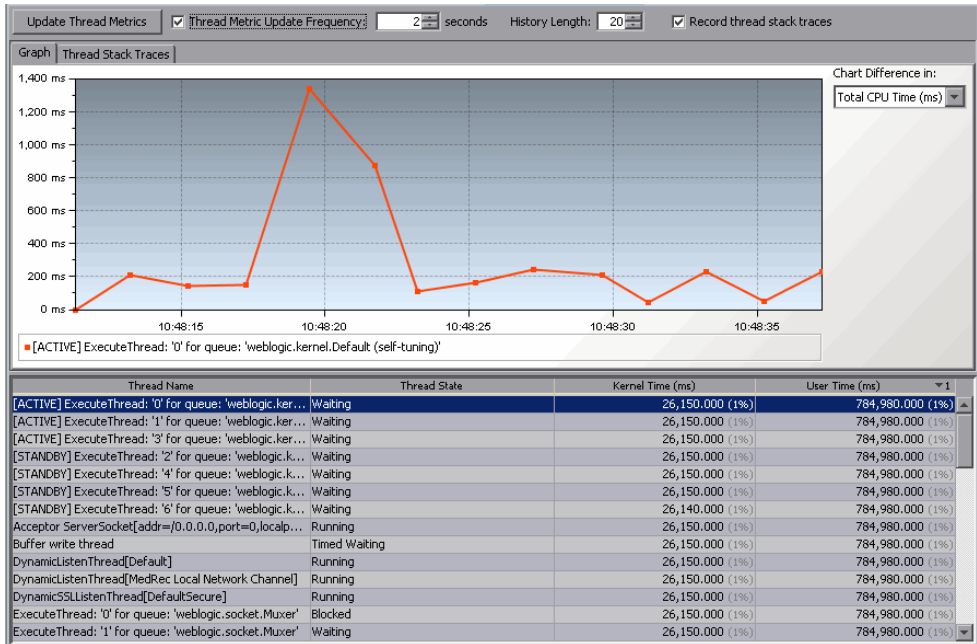
- ▶ “Understanding the Threads Tab” on page 322
- ▶ “Using the Thread Tab Controls” on page 323
- ▶ “Understanding the Threads Table” on page 325
- ▶ “Charting Thread Metrics” on page 325

Understanding the Threads Tab

The **Threads** tab is divided into four parts:

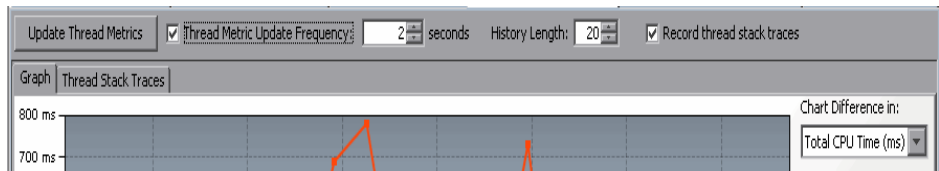
- ▶ **Controls.** Where you indicate how much, how often, and what kind of data is captured and displayed for the thread processing in your application.
- ▶ **Thread Table.** Where the metrics for each thread are listed.
- ▶ **Graph tab.** Where the metrics for the threads selected from the threads table are charted.
- ▶ **Thread Stack Traces tab.** Where the stack traces for the threads selected from the threads table are displayed when you have indicated that you want them to be captured.

The following image shows the **Threads** tab with the metrics for the selected thread charted in the graph:



Using the Thread Tab Controls

The controls at the top of the **Threads** tab allow you to control how often the thread metrics on the tab are updated and whether stack traces are captured for each thread. The following image shows the controls:



Controlling Thread Metric Updates

When the Threads tab is updated, the information displayed on the tab is refreshed with the latest thread metrics. If you are capturing stack traces, a stack trace is saved each time that the Profiler updates the metrics on the tab. You control how often the Profiler updates the thread metrics on the Threads tab.

To manually trigger an update of the Threads tab:

Click **Update Thread Metrics**

The Profiler refreshes the information in the graph and the thread table and captures stack traces.

To trigger periodic automatic updates of the Thread tab:

- 1 Select the **Thread Metric Update Frequency** check box to turn automatic updates on.

The Profiler immediately begins refreshing the thread metrics in the tab based upon the update interval specified in the **Thread Metric Update Frequency** spinner.

- 2 Select the update interval from the **Thread Metric Update Frequency** spinner.

The Profiler continues to automatically refresh the **Threads** tab using the new update interval.

Controlling Thread Stack Trace Capture

By default, whenever the Profiler updates the **Threads** tab, stack traces are captured for each of the threads listed in the thread table. You can control how many stack traces for each thread are displayed in the stack trace history and can turn off the capture of stack traces.

To set the number of stack traces that are displayed for each thread:

- ▶ Select the number of stack traces from the **History Length:** spinner.

The Profiler lists the indicated number of stack traces for the selected threads on the Thread Stack Traces tab.

To turn thread stack trace on and off:

- Select the **Record thread stack traces** checkbox to turn thread stack traces on or off.

Understanding the Threads Table

The threads table lists the metrics for the captured thread in the following columns:

- **Thread Name.** The name of the captured thread.
- **Thread State.** The state of the thread at the last thread metric update interval.
- **Kernel Time.** The portion of the CPU time during which the thread was executing in kernel mode.
- **User Time.** The portion of the CPU time during which the thread was executing in user mode.

The table can also include columns for **Thread Waited** and **Blocked Time** metrics if you enable them. To enable these metrics, set the **threads.contention.monitoring.enabled** property to true in the `<probe_install_dir>/etc/probe.properties` file. This setting may cause slightly higher probe overhead.

Charting Thread Metrics

When you start capturing thread metrics, the Profiler charts the thread with the highest User Time in the thread table by default. You can control which metrics are charted in the graph. You may chart the metrics for one or more of the threads listed in the threads table and you can select the metric that is to be charted for each thread.

To select the metric that is to be charted for each thread:

- Select the metric from the **Chart Difference in:** drop-down.

Diagnostics updates the graph to chart the indicated metric for each of the threads selected in the thread table.

To select the threads that are to have their metrics charted:

- 1 Select the row in the thread table that contains the thread whose metrics you would like to chart.

Diagnostics removes the metrics for any previously charted threads from the graph and charts the metrics for the selected thread. The graph legend is updated to indicate the color with which the selected threads metrics were charted.

- 2 To chart additional metrics in the graph along with the any that you have already charted:

- ▶ To select each additional thread one at a time, select each row in the thread table using **Ctrl-Click**.

Diagnostics charts the metrics for the selected thread in the graph and updates the graph legend to indicate the color with which the selected threads metrics were charted.

- ▶ To select a range of threads, select the row in the thread table using **Shift-Click**.

Diagnostics charts the metrics for the selected thread along with the metrics for all of the threads in the thread table that are between the selected threads and the newly selected thread. The graph legend is updated to indicate the colors with which the selected threads metrics were charted.

To remove the metrics from the chart for the selected threads:

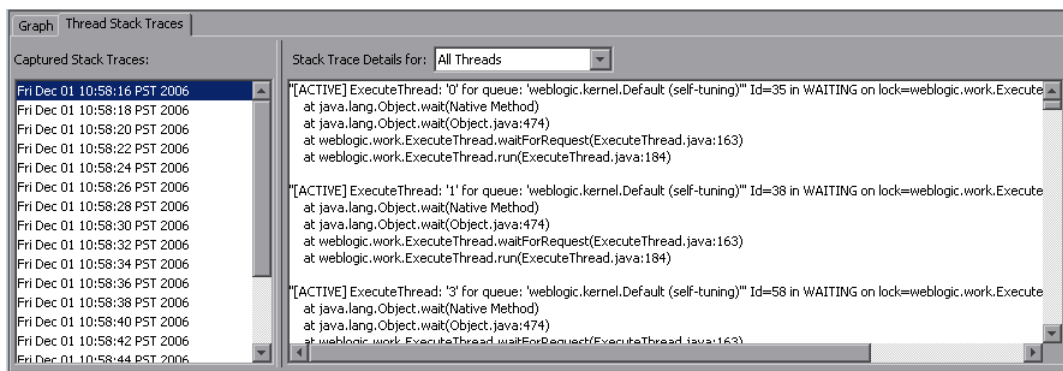
- ▶ Select the row in the thread table that contains the thread whose metrics you would like to remove from the chart using **Ctrl-Click**.

Diagnostics removes the metrics for the selected thread from the graph and updates the graph legend.

Viewing Stack Traces

When you start capturing thread metrics, the Profiler captures thread stack traces for each captured thread. You can view the stack traces for the threads selected in the thread table from the **Thread Stack Traces** tab.

The **Thread Stack Traces** tab is divided into three areas as shown in the following screen image:



- ▶ The **Stack Trace Details for** drop down allows you to control which thread's stack traces the Profiler is to display in the Stack Trace details area.

When you select **All Threads**, the stack traces for all threads are displayed in the stack trace details area. The selections made in the threads table do not impact the stack traces that are displayed in the stack trace details area when **All Threads** is selected.

When you select **Selected Threads**, the stack traces displayed in the stack trace details area are limited to those for the threads that you select in the threads table for the stack trace capture selected in the stack trace capture list.

- ▶ The Captured Stack Traces list contains a list of the times when stack trace captures occurred.
- ▶ The Stack Trace Details area is where the Profiler displays the stack traces that you indicated based on your selections from the stack trace capture list, the scope selection drop down, and the thread table.

Note: For instructions on selecting threads from the threads table, see “Charting Thread Metrics” on page 325.

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		0.3	Business Tier/EJ...
com.sun.j2ee.blueprints.pet...	110.3	110.3	1.0	0.0			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 names 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.

In order to see CPU time measurements you must turn on CPU Timestamp collection. For more information, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

- ▶ **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.

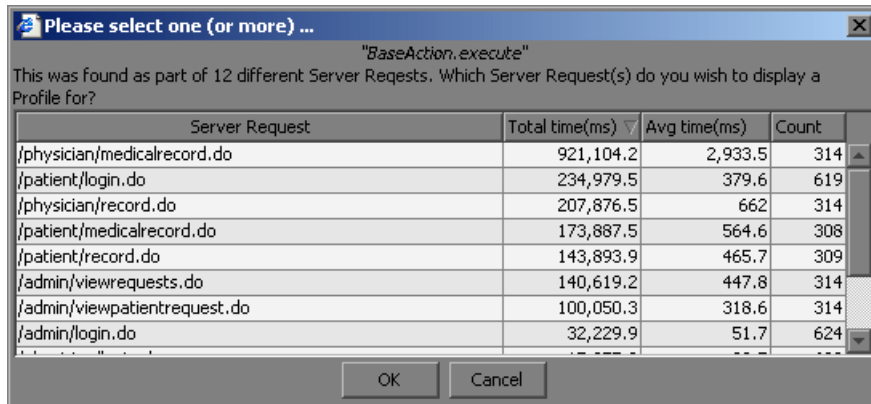
- ▶ **Layer.** The Layer associated with this method according to the instrumentation in the `auto_detect` points file. The layers are displayed on one level and there is no distinction made between layers and sub-layers.

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 338.

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

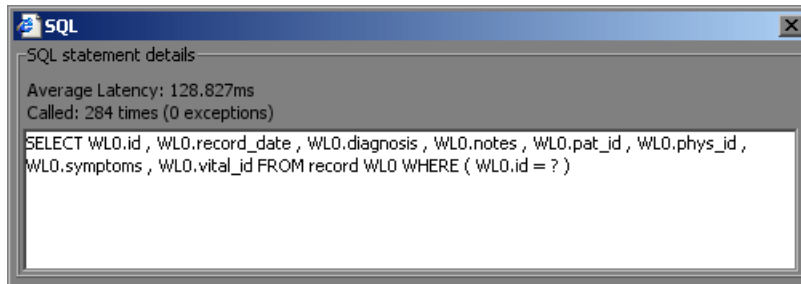
Understanding the All SQL Tab

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:319) ...</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.

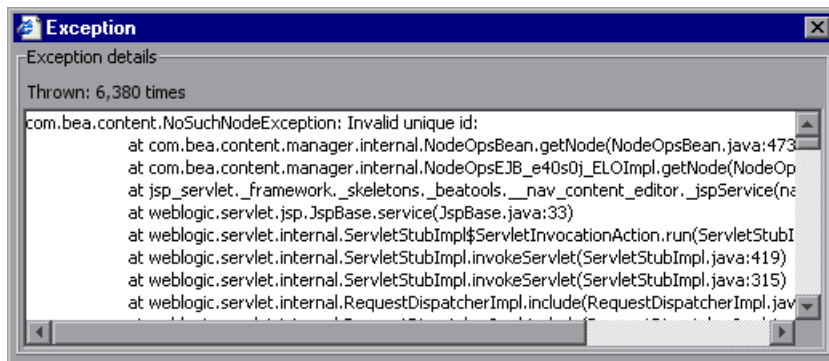
Note: If a non-instrumented method throws an exception which was caught and handled, that exception will not be reported

Understanding the Exceptions Tab

The Exceptions tab displays the Exceptions table which contains the following columns:

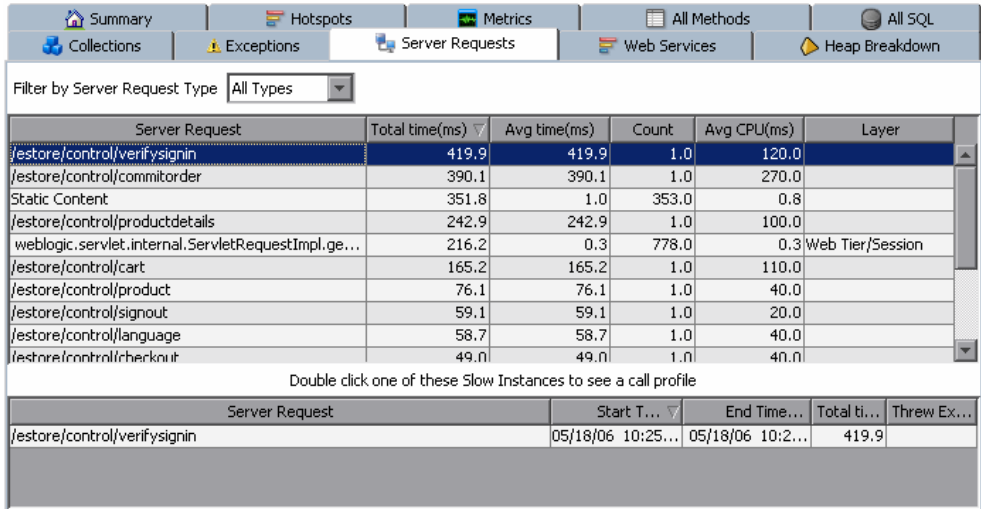
- ▶ **Stack.** Shows the first three lines of the exception stack trace.
- ▶ **Count.** The number of times the exception was generated.

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/rcheckout	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 338.

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 338.

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.

This section includes:

- ▶ “Types of Call Profile Windows” on page 338
- ▶ “Description of Call Profile Window” on page 338
- ▶ “Call Profile Graph” on page 340
- ▶ “Call Tree Table” on page 342
- ▶ “Details Pane” on page 344

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:

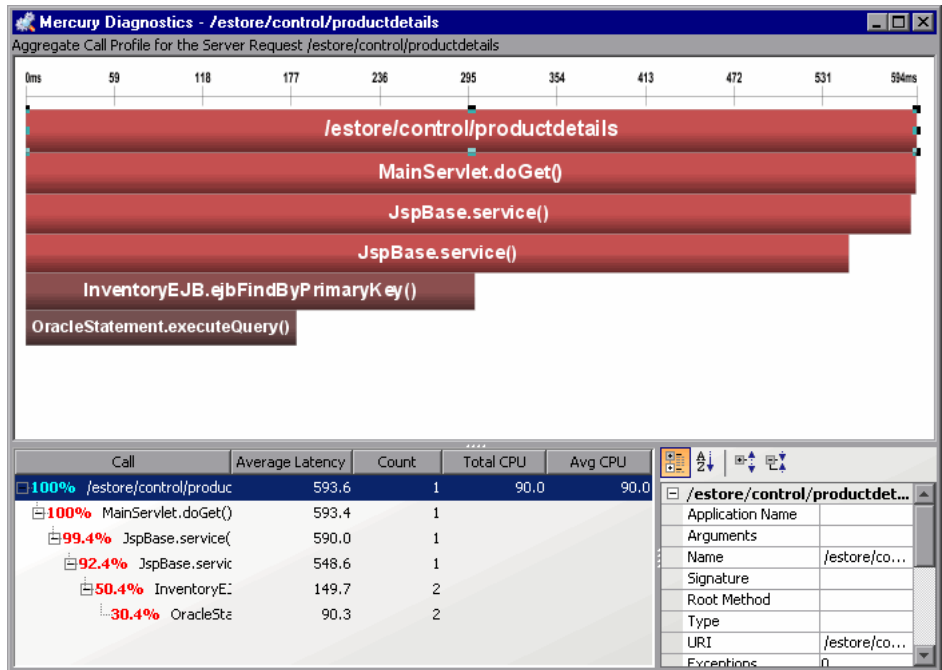
- ▶ **The Instance Call Profile window** displays the method calls that were made during the processing of the server request on which you drilled down.
- ▶ **The Aggregate Profile window** displays an aggregation of all of the method calls that were made during the processing of all of the server requests 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
- ▶ Details Pane

The following image is an example of an Aggregate Call Profile Window showing all three of these areas:

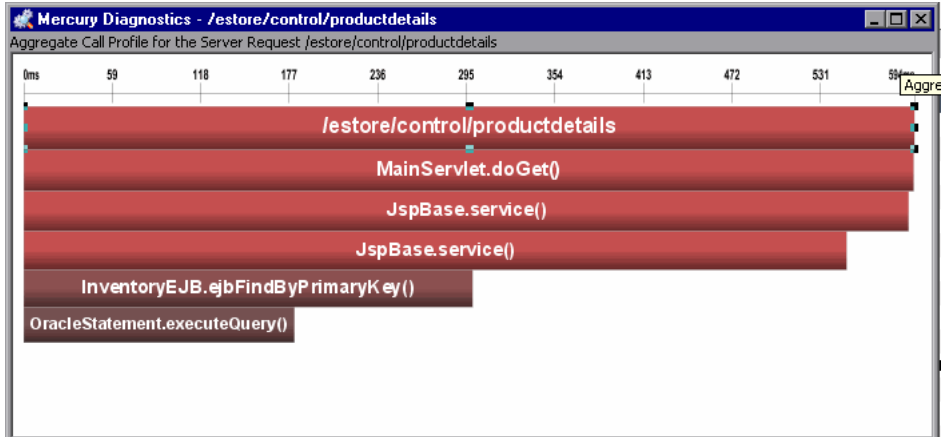


When you click a call box in the Call Profile graph, the corresponding row is selected in the Call Tree table and the metrics for the selected call are displayed in the Details pane. When you click on a row in the Call Tree table the corresponding call box in the Call Profile graph is selected and the metrics for the selected call are displayed in the Details pane.

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 Tooltips

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 Details pane.

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% ...	0.0	2		

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 Details pane.

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

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.

Details Pane

The **Details pane** 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 Details pane for the aggregated call profile.

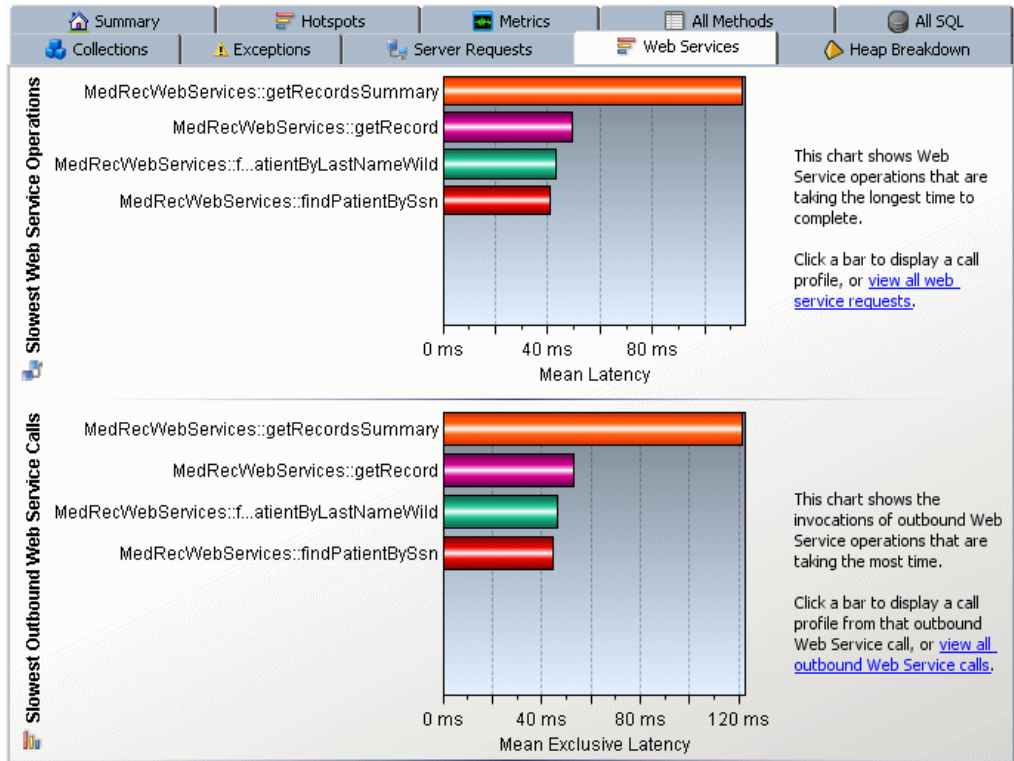
[-] /estore/control/cart	
Application Name	
Arguments	
Name	/estore/control/cart
Signature	
Root Method	
Type	
URI	/estore/control/cart
Exceptions	0
Count	1
Timeouts	0
[-] Latency	
Total CPU	80.0 ms
Exclusive Total Latency	0.1 ms
Max. Latency	258.4 ms
Min. Latency	258.4 ms
Standard Deviation	0.0 μ s
Total Latency	258.4 ms
Average Latency	258.4 ms

To view the details of a particular call in the Details pane, 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 338.

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 335.

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 338.

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 328.

Using the Configuration Tab

The Configuration tab in the J2EE Diagnostics Profiler provides a way for you to maintain the instrumentation points and the Probe configuration without having to manually edit the capture points file or property files. For information on how to use the Configuration tab see Chapter 32, “Using the Configuration Tab.”

33

Analyzing Memory with J2EE Diagnostics Profiler Tabs

This chapter provides a detailed description of the tabs, 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	349
Analyzing Memory Using the Allocation Analysis Tab	353
Analyzing Memory Using the Memory Analysis Tab	358
Analyzing Memory Using the Heap Breakdown Tab	359

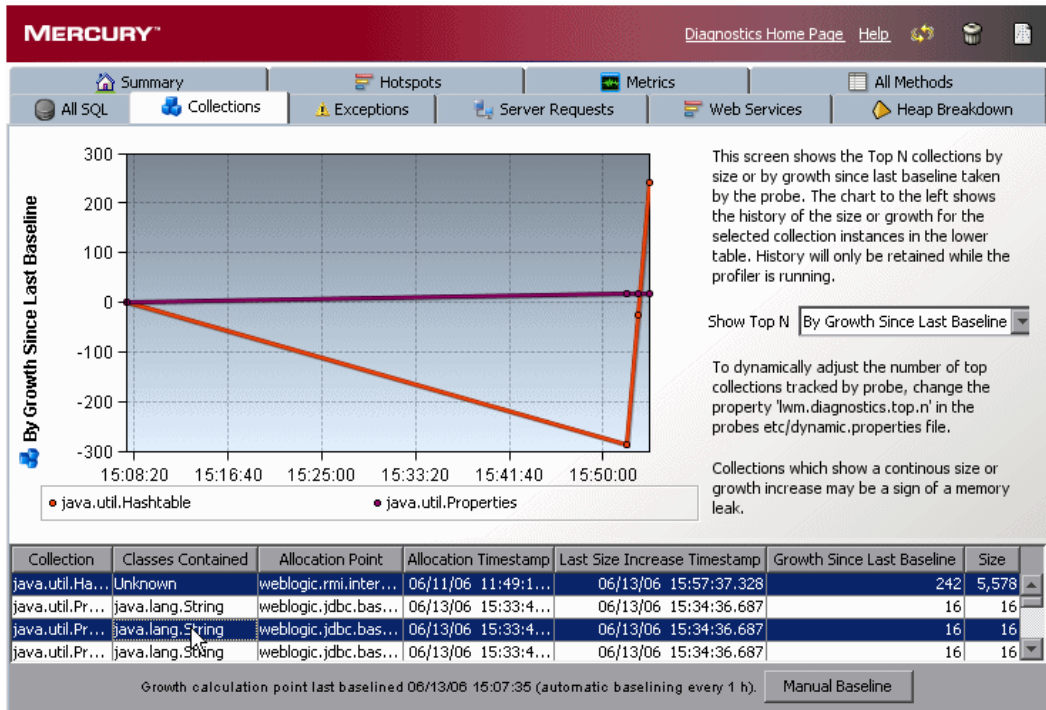
Analyzing Memory Using the Collections Tab

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.

The Collections tab is intended to be used to investigate a memory leak that you observe using the Heap Breakdown tab.

Understanding the Collections Tab

The Collections tab shows the metrics for the collections in your application in a graph and a corresponding table. The table lists the collections along with information about the allocation of the collection and the metrics for its growth rate and size. The graph contains the metrics charted for the collections that you selected. The growth rate of the collections are calculated from a baseline that the Profiler will periodically update or that you can update manually.



The Collections tab is divided into two parts: the collections table and the collections graph.

Collections Table

The collection table lists the collections sorted either by the amount of growth since the last baseline or the size of the collection depending on the selection that you make from the **Show Top N** box to the right of the graph.

The Collections 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.

Collections Graph

When you click the row for a collection in the collections table, the collections graph is updated to chart either the size or the growth of the collection since the last baseline, depending on the selection that you made from the **Show Top N** box to the right of the graph. You may chart the metrics for more than one of the collections by selecting subsequent rows with a CTRL-click.

Enabling the Collections Tab

By default, LWMD is disabled so that the J2EE Probe will not impose the additional overhead on its host when you are not going to use memory diagnostics metrics. When you detect a memory leak using the Heap Breakdown tab, you can enable LWMD. When you have completed your investigation, you can disable LWMD once more.

To enable LWMD and the Collections tab:

- Make the LWMD point in the `auto_detect.points` file active by setting `active` equal to `true`:

```
[Light-Weight Memory Diagnostics]
keyword = lwmd
active=true
```

Controlling The Charted Metrics and Table Sort Order

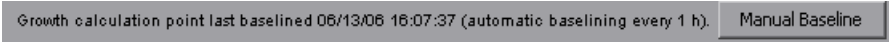
Your selection from the **Show Top N** box controls both the metrics that are charted in the collections graph and the sort order of the rows in the collections tables.

When you choose **By Size**, the collection table is sorted in descending order by collection size and the size metrics for the selected collections are charted in the collections graph.

When you chose **By Growth in Last Baseline**, the collection table is sorted in descending order by the amount of growth in the collection since the last baseline and the growth metrics for the selected collections are charted in the collections graph.

Establishing a Baseline

The baseline determines the time from which the growth in the size of the collections is measured. You can view the time that the last baseline was set at the bottom of the Collections tab as shown in the following image:



Growth calculation point last baselined 06/13/06 16:07:37 (automatic baselining every 1 h). [Manual Baseline](#)

The Profiler automatically sets a new baseline at preset periodic intervals. You can also manually set a new baseline.

To manually set a new baseline click **Manual Baseline**. The Profiler resets the Growth Since Last Baseline metric for each collection and refreshes the charted metrics in the graph.

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_install_dir>\etc\dynamic.properties` file, locate the following line:

```
lwm.diagnostics.auto.baselining.interval=60m
```

- 2** In the referenced line, change the time interval according to your needs, as explained in the comments of the file.

Note: If you want to stop automatic baselining, enter **0** for the time interval.

Analyzing Memory Using the Allocation Analysis Tab

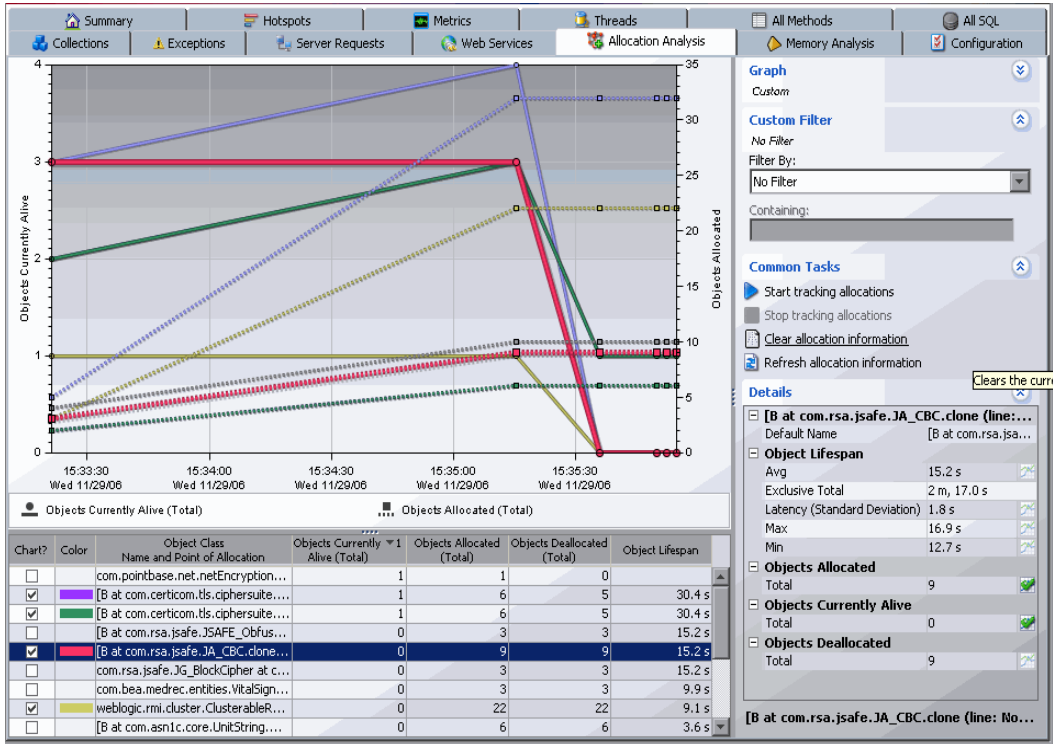
The Allocation Analysis tab provides a way for you to investigate a memory leak that you have observed in the Heap Breakdown tab by examining the allocation and deallocation of objects while the leak is happening.

Understanding the Allocation Analysis Tab

The Allocation Analysis tab shows the metrics for the objects that have been allocated by your application in a graph and a corresponding table. The table lists the allocated objects along with the number of allocated instances and their lifespan. The graph contains the charted metrics for the selected allocated objects.

To analyze allocations, you must use the controls in the Common Tasks menu to track allocations and refresh the displayed metrics as you exercise the application functionality that you believe may be experiencing leaks.

The following is an example of the Allocation Analysis tab.



The Allocation Analysis tab is similar to the views with a detail layout in the Diagnostics views. For more information on using these controls see Chapter 3, “Viewing Diagnostics Data in Detail Layout”. Instead of the view filters appearing in the view title, they appear in view filter menus along the side of the graph. The Common Tasks menu controls the tracking of the allocations and the refreshing of the information that is displayed for the entire view.

Allocation Analysis Table

The allocation analysis graph-entity table lists the objects that have been allocated since you started tracking allocations. This table can be customized to adjust the sort order and the columns that appear in the table just like the graph-entity tables in other views with the detail layout. For more information on using these controls see Chapter 3, “Viewing Diagnostics Data in Detail Layout”. By default, the table is sorted in order by Objects Currently Alive and displays the following columns:

- **Chart?**. Allows you to indicate if the metrics for the allocated object are to be charted in the graph. You can select objects to be charted by manually clicking on the box in this column or you can let the Profiler to dynamically select the objects to chart using the criteria that you specify in the Graph filter.
- **Color**. Indicates the color that the Profiler used to chart the metrics for the allocated object. No color is shown for metrics have not been charted.
- **Objects Currently Alive**. A count of the total number of allocated objects that have not yet been garbage collected.
- **Objects Allocated**. A count of the total number of objects that have been allocated whether they have been garbage collected or not.
- **Objects Deallocated**. A count of the total number of objects that have been garbage collected.
- **Object Lifespan**. The average duration of the life of all deallocated objects. When no objects have been deallocated this column will be blank.

Allocation Analysis Graph

The allocation analysis graph charts the metrics that you selected from the details table for each of the objects selected in the allocation analysis table.

Using the controls in the views with a detail layout, you control which metrics are charted and which entities have their metrics charted. For more information on using these controls see Chapter 3, “Viewing Diagnostics Data in Detail Layout.”

Enabling the Allocation Capture

The Allocation Analysis tab cannot display any allocation objects or their metrics until allocation capture has been enabled for the Probe. By default, allocation capture is disabled so that the J2EE Probe will not impose the additional overhead on its host when you are not going to use memory diagnostics metrics. When you suspect that you may have a memory issue with the way that your application manages its object allocations, you can enable allocation capture. When you have completed your investigation, you can disable the allocation capture again.

To enable allocation capture and the Collections tab:

- 1 In the `auto_detect.points` file located in `<probe_install_directory>\etc`, modify the default settings to match the following:

```
[Allocation]
keyword = allocation
detail = leak
scope = !com\.mycompany\.mycomponent\.*
active = true
```

If you want to have reflective allocation tracked, you can add the reflection attribute to the detail argument in the Allocation point.

```
[Allocation]
keyword = allocation
detail = leak,reflection
scope = !com\.mycompany\.mycomponent\.*
active = true
```

This will instrument `Class.newInstance`, `Constructor.newInstance` and `Object.clone` methods. The reflection instrumentation tracks all classes that are created.

- 2 Restart the monitored application so that the Probe restarts and can apply the updated instrumentation.

Analyzing Object Allocations Using the Allocation Analysis Tab

Once you have identified a memory problem using the Heap Breakdown tab, you can analyze the object allocations that your application is performing by examining the allocations while the suspected application functionality is being executed. The following instructions show you how to run an experiment and study the resulting application performance.

To analyze object allocations:

- 1** If you have not already enabled allocation capture for the Probe, do so as instructed in “Enabling the Allocation Capture” on page 356.
- 2** Begin tracking allocations by selecting **Start Tracking Allocations** from the Common Tasks menu.

The probe starts collecting the metrics for the objects that are being allocated and deallocated. No collection metrics are displayed in the tab until you select the **Refresh Allocation Information** or **Stop Tracking Allocations** menu options.

- 3** Execute the application functions that you suspect may be causing a leak so that any objects that are allocated while performing the function can be tracked.
- 4** Select the **Stop Tracking Allocations** menu option to limit the tracked objects to those that were captured while the suspect application functions were being performed.

No additional instances are tracked once you stop tracking. The instances of the objects that were already allocated continue to be tracked as they are deallocated so that the metrics on the tab can be refreshed with accurate counts of the objects that are alive or deallocated and accurate object lifespans.

- 5 Select the **Clear Allocation Information** menu option to update the tab with the current metrics for the allocated objects.

Each time that you select this menu option, the Profiler updates the metrics for the tracked objects in the allocations analysis table with the current counts and lifespans. The trend lines for the metrics in the graph are updated to chart the data points for the metrics at the refresh time.

You should repeat this step as your application continues to run so that you can see what happens to the allocated objects over time.

- 6 If you want to run your experiment again, select the **Refresh Allocation Information** menu option to clear the table and graph of all of the objects and metrics currently displayed and begin this process again from the second step.

Analyzing Memory Using the Memory Analysis Tab

The Profiler displays the Memory Analysis tab instead of the Heap Breakdown tab when your application is running JVM 1.5 or higher.

Note: The Memory Analysis tab is included in the Mercury Diagnostics Profiler for J2EE as a technology preview. Please contact Mercury Support for any questions or concerns.

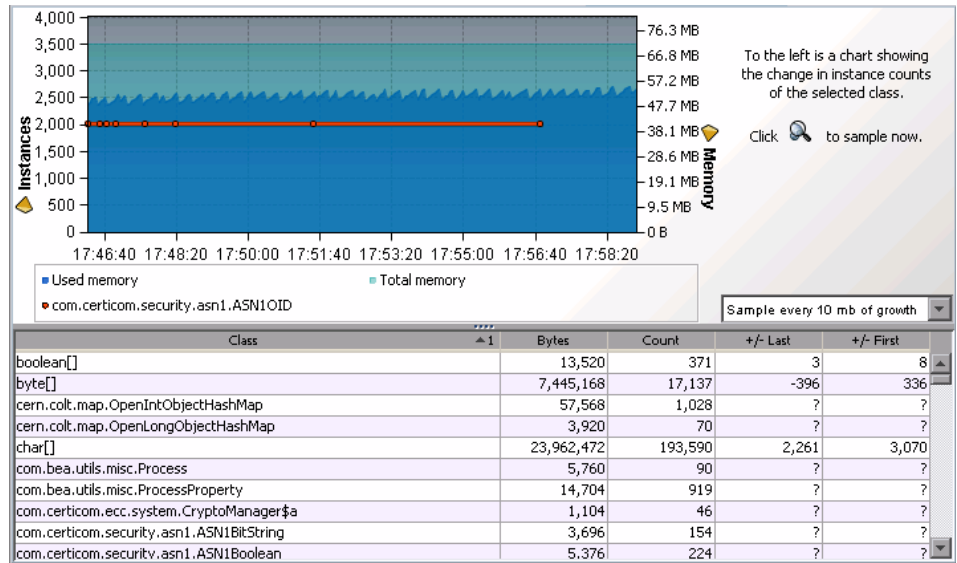
The Memory Analysis tab is a wizard that is made up of the Heap Breakdown tab and the Reference Graph tab. The instructions in the wizard guide you through the steps that help you to diagnose a memory leak that you observed in the Heap Breakdown tab

Analyzing Memory Using the Heap Breakdown Tab

The Profiler displays the Heap Breakdown tab instead of the Memory Analysis tab when your application is running a JVM version earlier than JVM 1.5.

Note: Heap Breakdown has been known to be unstable and to crash when used with IBM JVM 1.3.x.

The **Heap Breakdown** tab allows you to 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.



Enabling 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_install_dir>\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

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 by objects of this class.
- **Count.** The number of object instances of this class that are allocated 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. You can select more than one class to display on the graph using the **Control** key.

Adjusting Sampling Rate

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 VI

Using the Mercury Diagnostics Profiler for .NET

34

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	366
Mercury Diagnostics Profiler for .NET Processing	367
Common .NET Profiler Tab Navigation and Display Controls	369

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.

Depending on your authentication settings, you may be prompted to enter a username and password.

The default username is `admin`. The default password is `admin`.

For more information about authentication and authorization of users of the Profiler, refer to the *Mercury Diagnostics Installation and Configuration Guide*.

For more details about usernames and passwords, refer to 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 **View Profiler for <probe name>** from the menu.
- ▶ Alternatively, in the probe standard view screen in Mercury Diagnostics, right-click the probe entity in the graph entity table and select **View Profiler for <probe name>** 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 35, “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

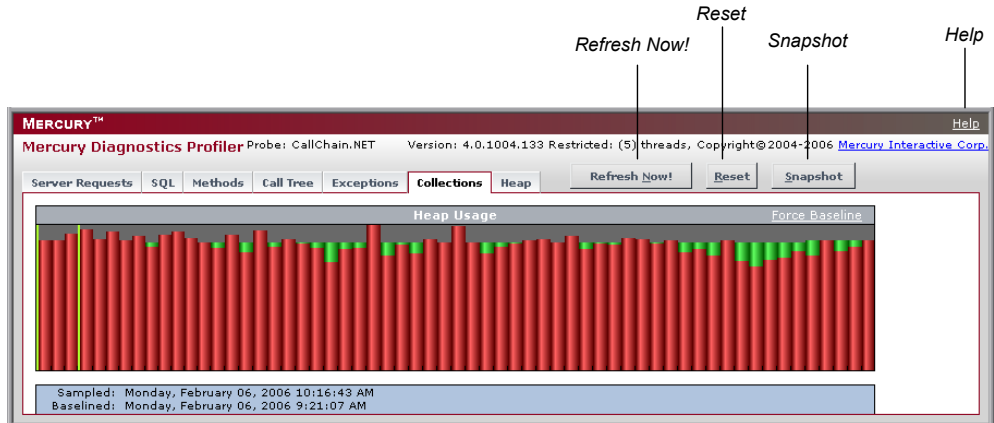
Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you to identify objects that may be leaking.

For more information about Light Weight Memory Diagnostics, see “Analyzing Memory Using the Collections Tab” on page 393.

For more information about Heap Breakdown, see “Analyzing Memory Using the Heap Tab” on page 399.

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.

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**.

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 the **Snapshot** button.

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.

35

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	372
Analyzing Performance Using the SQL Tab	377
Analyzing Performance Using the Methods Tab	380
Analyzing Performance Using the Exceptions Tab	383
Analyzing Performance Using the Call Tree Tab	385

Analyzing Performance Using the Server Requests Tab

This section includes:

- ▶ “Introducing the Server Requests Tab” on page 372
- ▶ “The Server Requests Tab at a Glance” on page 373
- ▶ “Viewing Instance Information for Server Requests” on page 376
- ▶ “Viewing the Layer Breakdown for a Server Request Instance” on page 376
- ▶ “Viewing the Call Tree for a Server Request Instance” on page 376

Introducing the Server Requests Tab

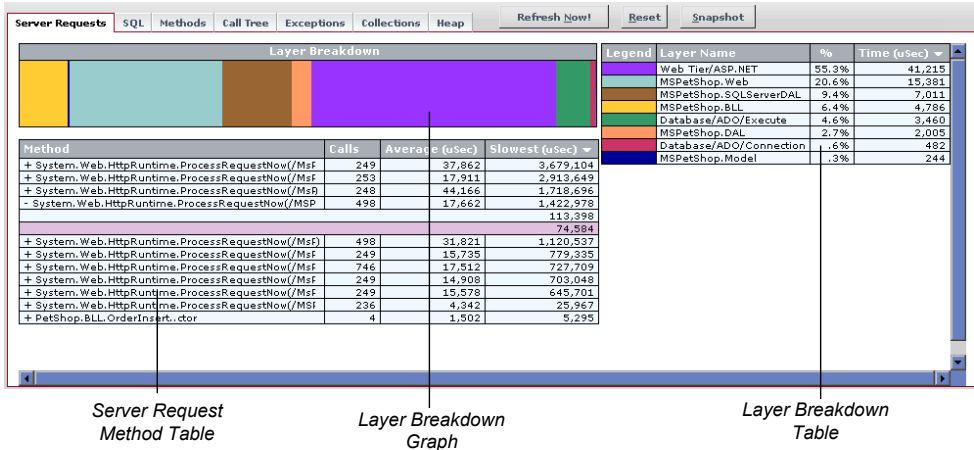
The .NET Diagnostics Profiler keeps track of all of the method calls made by your application. The **Server Requests** tab displays information about the server request methods. The server request methods are listed in a table that shows the number of times that each method was executed, along with the 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

The Method table lists the server requests that have been called. You can sort the table by clicking the column headers.

Method	Calls	Average (µs)	Slowest (µs) ▼
+ 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.

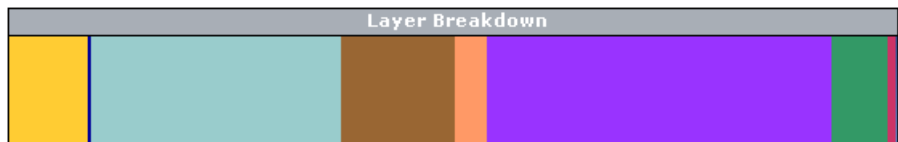
Legend	Layer Name	%	Time (µs) ▼
	Web Tier/ASP.NET	55.3%	41,215
	MSPetShop.Web	20.6%	15,381
	MSPetShop.SQLServerDAL	9.4%	7,011
	MSPetShop.BLL	6.4%	4,786
	Database/ADO/Execute	4.6%	3,460
	MSPetShop.DAL	2.7%	2,005
	Database/ADO/Connection	.6%	482
	MSPetShop.Model	.3%	244

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 385.

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.

This section includes:

- ▶ “Undersanding the SQL Method Table” on page 378
- ▶ “Viewing Instance Information for SQL Method Calls” on page 379
- ▶ “Viewing the Call Tree for an SQL Method Instance” on page 380

Undersanding the SQL Method Table

The SQL tab contains the SQL Method table.

Method	Calls	Average (µs)	Slowest (µs) ▼	SQL
+ PetShop.SQLServerDAL.SQHe	998	4,403	1,401,536	
+ System.Data.SqlClient.SqlCon	249	7,805	1,400,624	SELECT Account.Emat.L...
- System.Data.SqlClient.SqlCorr	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, Nategory...
+ PetShop.SQLServerDAL.SQHe	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 385.

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.

This section includes:

- ▶ “Understanding the Method Table” on page 381
- ▶ “Viewing Instance Information for Method Calls” on page 382
- ▶ “Viewing the Call Tree for a Method Instance” on page 382

Understanding the Method Table

The Methods tab contains the Method table.

Method	Calls	Average (µs)	Slowest (µs)
+ 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 385.

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.

Note: Exceptions are only captured by the probe if the exception causes the termination of a method. If the instrumented method handles the exception, no exception information is gathered by the probe.

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.

This section includes:

- ▶ “Understanding the Exception Table” on page 384
- ▶ “Viewing Instance Information for Exceptions” on page 384
- ▶ “Viewing the Call Tree for an Exception” on page 385

Understanding the 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 385.

Analyzing Performance Using the Call Tree Tab

This section includes:

- “Introducing the Call Tree Tab” on page 386
- “Accessing the Call Tree Tab” on page 386
- “The Call Tree Tab at a Glance” on page 387
- “Call Tree Methods” on page 389

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.

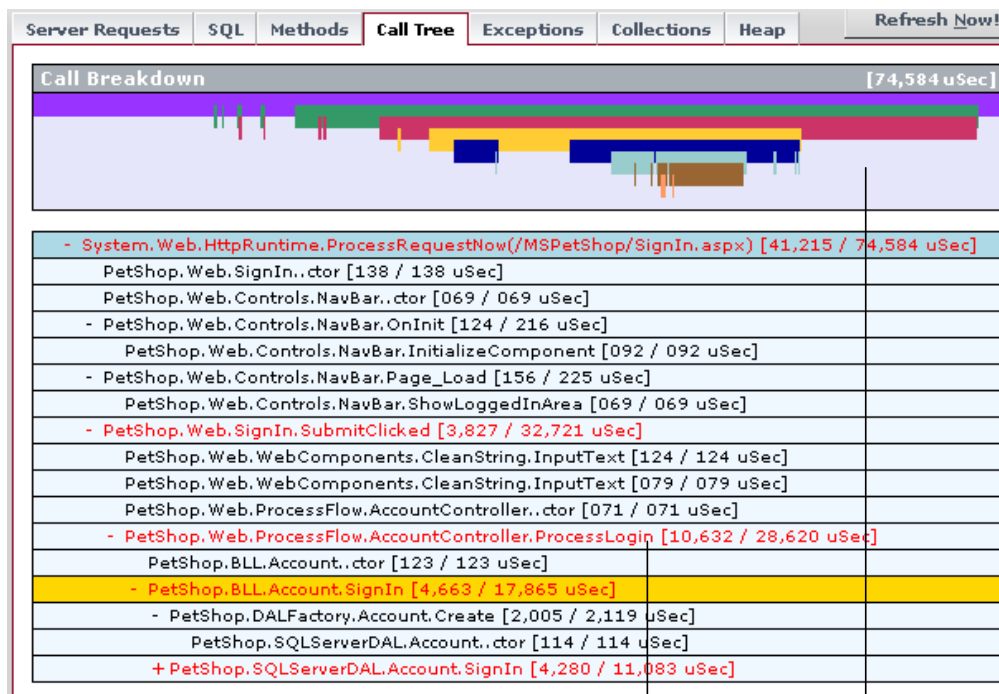
Accessing the Call Tree Tab

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

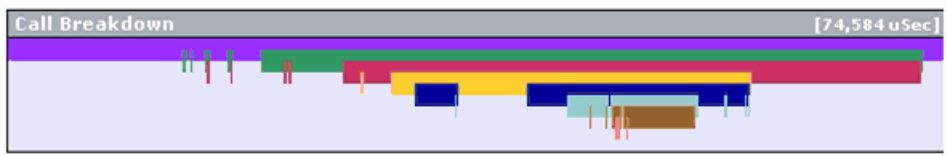


Call Tree
Table

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

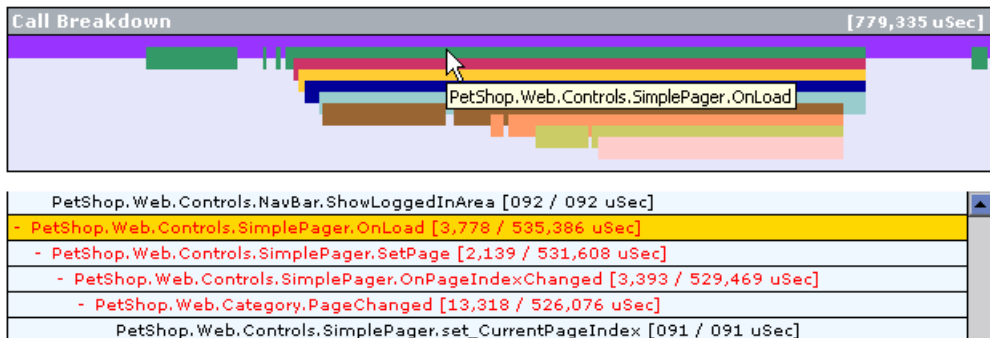
To see a captured call tree on the Call Tree tab you must select a method instance from one of the other .NET Diagnostics Profiler tabs. When you select an method instance from a tab the Call Tree tab opens with the call tree that contains the selected instance scrolled so that the selected method is 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.DAL.Factory.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]

36

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	393
Analyzing Memory Using the Heap Tab	399

Analyzing Memory Using the Collections Tab

This section includes:

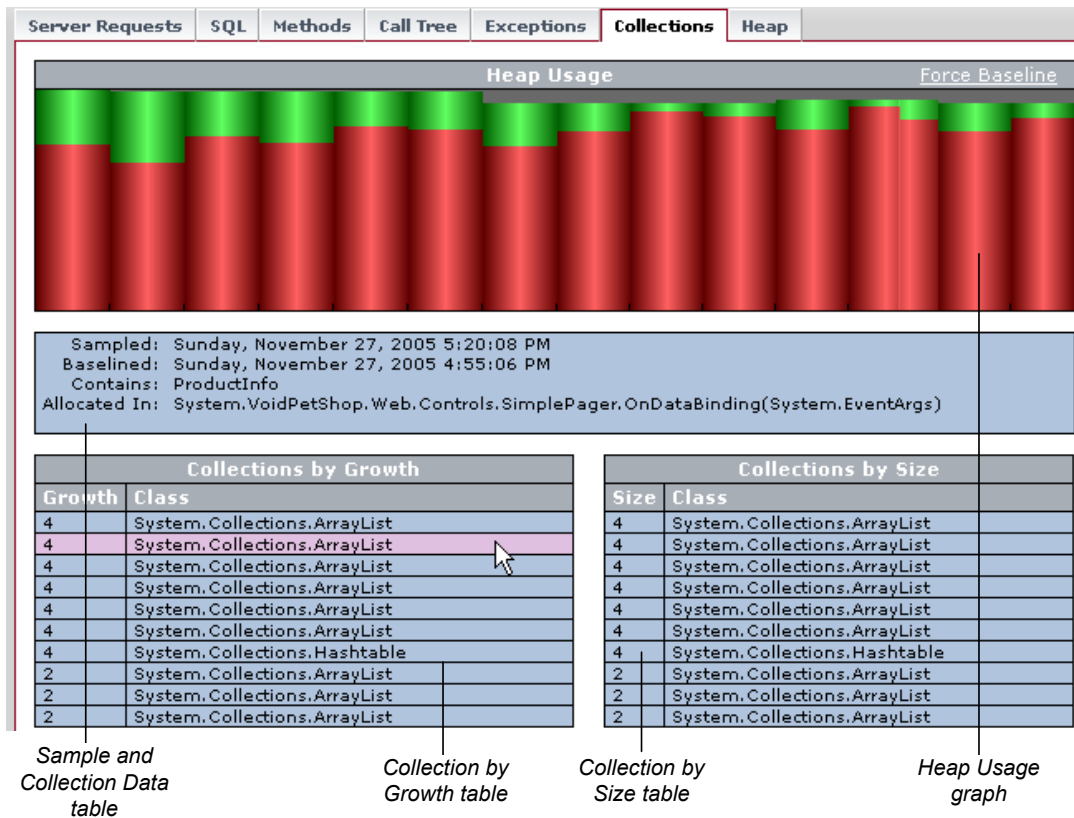
- “Introducing the Collections Tab” on page 394
- “The Collections Tab at a Glance” on page 394
- “Heap Usage Graph” on page 395
- “Sample and Collection Detail Table” on page 396
- “Collections by Growth Table” on page 397
- “Collections by Size Table” on page 398
- “Viewing Details for a Selected Collection” on page 398

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.

The Collections Tab at a Glance

The following image is an example of the Collections tab:

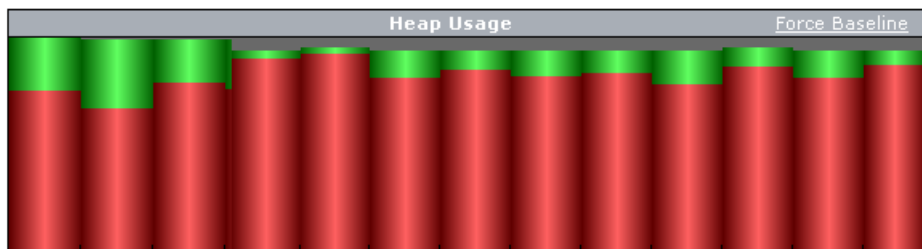


The Collections tab is divided into the following sections:

- Heap Usage Graph
- Sample and Collection Detail Table
- Collections by Growth Table
- Collections by Size Table

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. This information is displayed when you mouse over the Collections by Growth or Collections by Size tables.
- ▶ **Allocated In.** The method that allocated the selected collection. This information is displayed when you mouse over the Collections by Growth or Collections by Size tables

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.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.

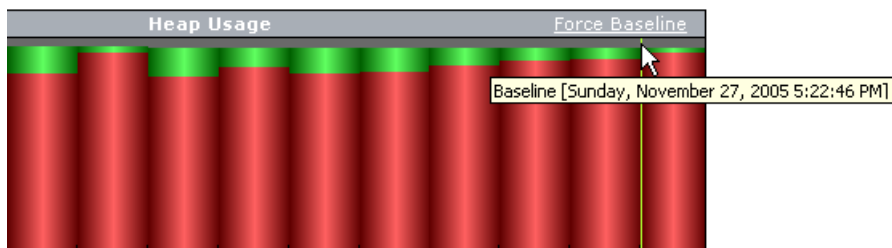
Viewing Details for a Selected Collection

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 pane.

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.

This section includes:

- “Accessing the Heap Tab” on page 400
- “The Heap Tab at a Glance” on page 401

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

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

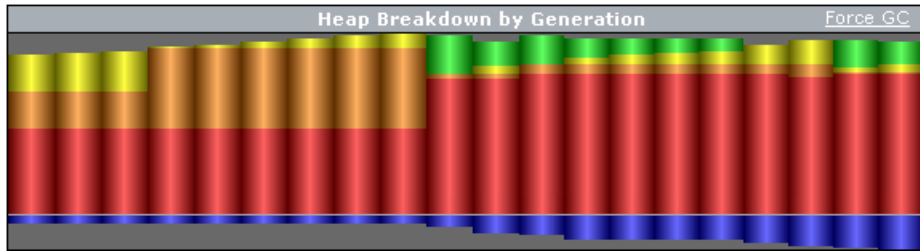
Heap Metrics
table

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 VII

Diagnostics Integration with Other Mercury Products

37

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	408
Accessing the Diagnostics Screens	409
Monitoring Diagnostics Performance Data from Dashboard	409
Drilling Down to Diagnostics from Dashboard	417
Diagnostics Performance Reports in Mercury Business Availability Center	421
Drilling down to Diagnostics Data from Mercury Business Availability Center Reports	425

About Viewing Diagnostics Data in Mercury Business Availability Center

Mercury Diagnostics 6.5 can be integrated with Mercury Business Availability Center 6.1 or later to provide information to help you understand and improve the performance of your applications.

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 417
- ▶ “Drilling down to Diagnostics Data from Mercury Business Availability Center Reports” on page 425

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.

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.

This section includes:

- ▶ “Monitoring Performance Status Using KPIs” on page 410
- ▶ “Monitoring Diagnostics Probe Data from Dashboard” on page 413
- ▶ “Monitoring Transactions from Dashboard” on page 414

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 navigation bar at the top containing tabs: Top View, Console, Filters, Geographical Map, Custom Map, Topology Map, and Reports. Below the navigation 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 eight 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 performance metrics for various BPM transactions. A tooltip is displayed over the 'Application' KPI for the 'admin_login' transaction, providing detailed status information.

Name	Performance	Availability	Application	Ack
admin_login	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>
admin_login	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>
edit p...	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>
patien...	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>
physic...	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>
Simple...	● ◆	● ◆	● ◆	<input checked="" type="checkbox"/>

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.

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 63.

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

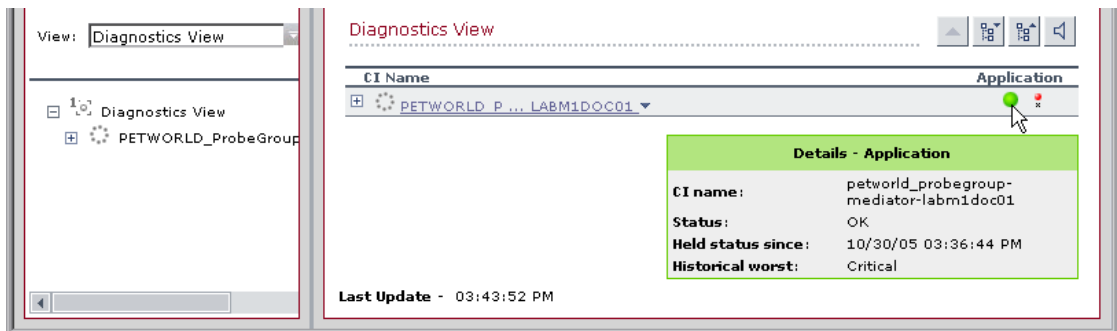
In the Diagnostics View in Dashboard, you can view the status of your applications that are being monitored by Mercury Diagnostics Probes. You can view the status of an individual Diagnostics probe or of a group of Diagnostics probes known as a probe group.

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 410.

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 417.

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 410.

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**.

- 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.

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” on page 417.

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.

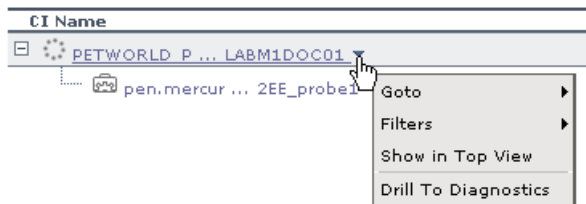


Note: In the BPM related views, you drill down to Diagnostics from the **BPM Transaction CI**. Click the **Expand All** button to expand the **Business Process Step** CIs and to view the individual **BPM transaction CIs**.

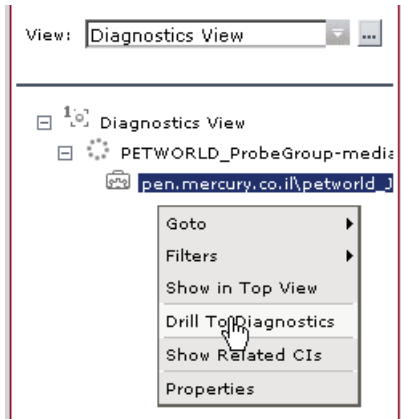
In Diagnostics View, you either drill down to Diagnostics from a **Diagnostics Probe Group CI** or from an individual **Diagnostics Probe CI**.

- 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 or later

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 “Transactions View” on page 227.

- **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 “Server Requests View” on page 187.

- **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.

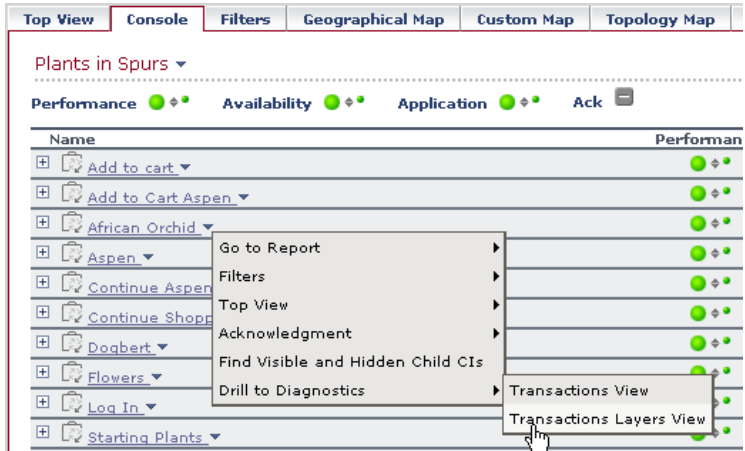
The load view displays the performance metrics for the Diagnostics layers where processing has taken place in your application. For more information about the Diagnostics Load view, see “Load View” on page 161.

The following table displays the Diagnostics drilldown options for CIs in each of the relevant Dashboard views.

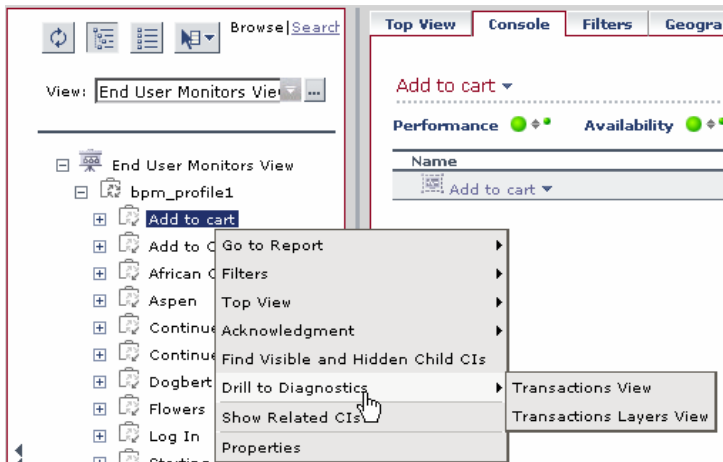
Dashboard View	CI Type	Diagnostics drilldown options
BPM related view	Business Process Group	► Transactions View
	Business Process Step	► Transactions View ► Transactions Layers View
	BPM Transaction from Location	► Transactions View ► Transactions Layers View
RUM related view	End User Management Application Related Group	► Server Requests
	Business Process Step	► Server Requests
	RUM Page Monitor	► Server Requests
Diagnostics view	Diagnostics Probe Group	► Probe Group Summary ► Load View
	Diagnostics Probe	► 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

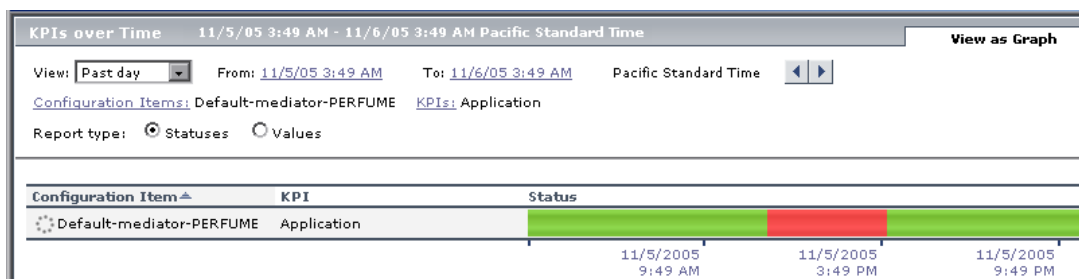
From Dashboard, you can generate the high level reports about the performance of your Diagnostics components and of specific transactions.

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 or later, 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 430.

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 410.

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 424.

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” on page 424.

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.

This section includes:

- ▶ “Drilling down to Diagnostics Data from Transaction Breakdown Reports” on page 425
- ▶ “Drilling down to Diagnostics Data from Real User Monitor Reports” on page 428
- ▶ “Drilling down to Diagnostics Data from KPI Reports” on page 430
- ▶ “Drilling Down to Diagnostics from Mercury Business Availability Center for SOA reports” on page 432

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:

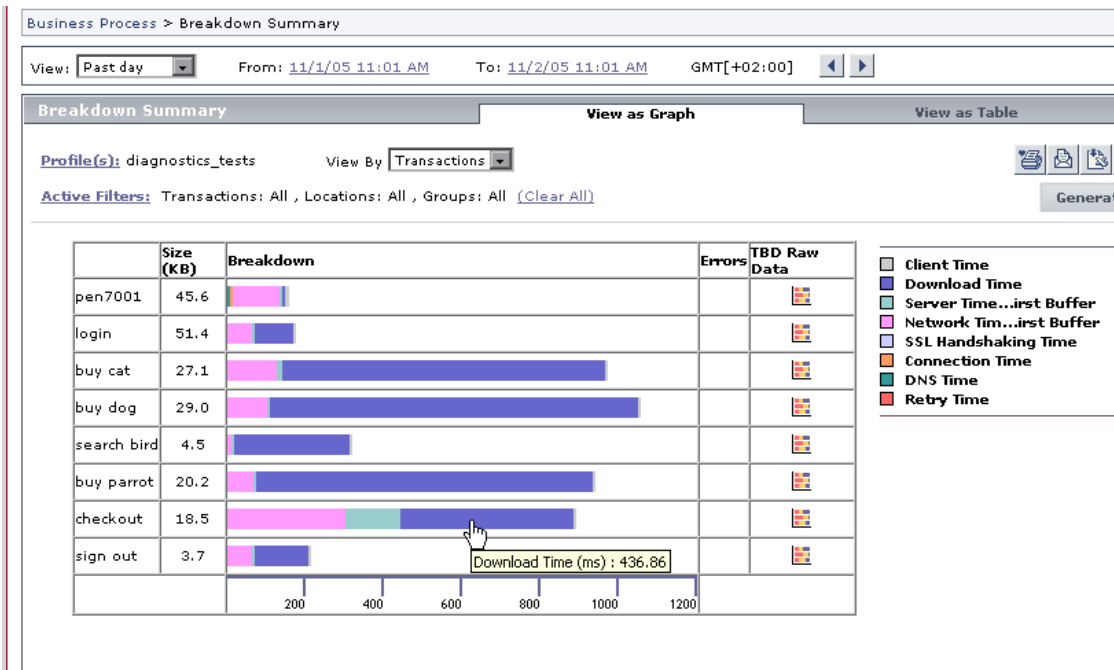
- ▶ **Server Time to First Buffer.** This measurement displays the average amount of time that passes from the receipt of ACK of the initial HTTP request (usually GET) until the first buffer is successfully received back from the Web server.
- ▶ **Download Time.** This measurement displays the time from 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.



- 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 “Transactions View” on page 227.

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	3.277	-	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 “Transactions View” on page 227.

Drilling down to Diagnostics Data from Real User Monitor Reports

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 “Server Requests View” on page 187.

Drilling down to Diagnostics Data from KPI Reports

Note: This drill down option is only available in Mercury Business Availability Center 6.2 or later.

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 410.

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:

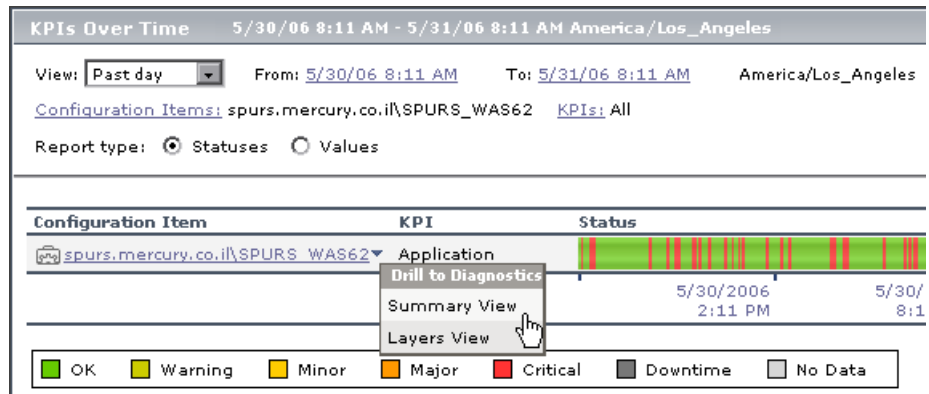
CI Type	Diagnostics drilldown options
Diagnostics Probe Group	<ul style="list-style-type: none"> ▶ Summary View (Probe Group Summary) ▶ Layers View (Load)
Diagnostics Probe	<ul style="list-style-type: none"> ▶ Summary View (Probe Summary) ▶ Layers View (Load)
Business Process Step	<ul style="list-style-type: none"> ▶ Transactions View ▶ Layers View

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 424.

- 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	5/30/2006 2:11 PM - 5/30/2006 8:11 PM

OK
 Warning
 Minor
 Major
 Critical
 Downtime
 No Data

Drilling Down to Diagnostics from Mercury Business Availability Center for SOA reports

Note: This drilldown option is only available in Mercury Business Availability Center 6.5 and later.

The Mercury Business Availability for SOA reports enable you to proactively monitor Web service calls. From certain Mercury Business Availability Center for SOA reports, you can drill down to the Diagnostics Web services views to view detailed breakdowns of the relevant Web service operations.

You can drill down to Diagnostics from the SOA Health report and from the SOA Metrics Over Time Report.

For more information about generating SOA reports in Mercury Business Availability Center, refer to the section on “Working with Mercury Business Availability Center for SOA” in *Using Dashboard* in the *Mercury Business Availability Center Documentation Library*.

Important:

- ▶ In SOA reports, you drill down to Diagnostics from Web Service operations and not from the overall Web services.
 - ▶ The drilldown to Diagnostics is only available from SOA reports whose **Data Type** selection in Mercury Business Availability Center is **Real**.
-

This section includes:

- ▶ “Drilling down from the SOA Health Report” on page 433
- ▶ “Drilling down from the SOA Metrics Over Time Report” on page 434

Drilling down from the SOA Health Report

The SOA Health report displays health metrics for the selected Web services or operations. The Web services or operations can be filtered according to server or the Consumer IP from which they originated or both.

You can drill down to Diagnostics from Web service operations (not from the overall Web service) displayed in the SOA Health report.

To drill down to Diagnostics from the SOA Health report:

- 1 Generate a SOA Health report in Mercury Business Availability Center involving Web service operations monitored by Diagnostics.

Note: When you generate the SOA Health report, under **Data Type**, select **Real**.



- 2 Click the **Drill down to Diagnostics** button next to the Web service operation from which you want to drill down.

Diagnostics displays the selected Web service operations in one of the Inbound Web services views over the same time period that was selected in the SOA report.

- ▶ If you drill down from Web service operations that were filtered by consumer in the SOA report, Diagnostics displays the Web service operations in the **Inbound/Provider by Consumer IP** view.
- ▶ If the Web service operations are not filtered by consumer, Diagnostics displays them in the **Inbound/Provider** view.

For more information about Diagnostics Web services views, see “Web Services Views” on page 271.

Drilling down from the SOA Metrics Over Time Report

The SOA Metrics Over Time report displays performance metrics over time for the selected Web services operations. The Web services or operations can be filtered according to server, the Consumer IP from which they originated, or both.

You can drill down to Diagnostics from Web service operations (not from the overall Web service) displayed in the SOA Metrics Over Time report.

To drill down to Diagnostics from the SOA Health report:

- 1 Generate a SOA Metrics Over Time report in Mercury Business Availability Center involving Web service operations monitored by Diagnostics.

Note:

- When you generate the SOA Health report, under **Data Type**, select **Real**.
- Filter the graph according to Operations and not according to Web services. (In the Active filters dialog box, under **Selected CIs**, select **Operations**)

-
- 2 Click the relevant point in the graph to open the **Available Drilldowns** menu and select **Drill Down to Diagnostics Report**.

Diagnostics displays the selected Web service operations in one of the Inbound Web services views over the same time period that was selected in the SOA report.

- If you drill down from Web service operations that were filtered by consumer in the SOA report, Diagnostics displays the Web service operations in the **Inbound/Provider by Consumer IP** view.
- If the Web service operations are not filtered by consumer, Diagnostics displays them in the **Inbound/Provider** view.

For more information about Diagnostics Web services views, see “Web Services Views” on page 271.

38

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	435
Configuring LoadRunner Scenarios to use Mercury Diagnostics	436
Drilling Down to Diagnostics Data During a Load Test Scenario	440
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	442

About Viewing Mercury Diagnostics Data in LoadRunner 8.1

This section includes:

- “Setting Up LoadRunner to Use Mercury Diagnostics” on page 436
- “Viewing Diagnostics Data in LoadRunner” on page 436

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 436.

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 440.

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 442.

Configuring LoadRunner Scenarios to use Mercury Diagnostics

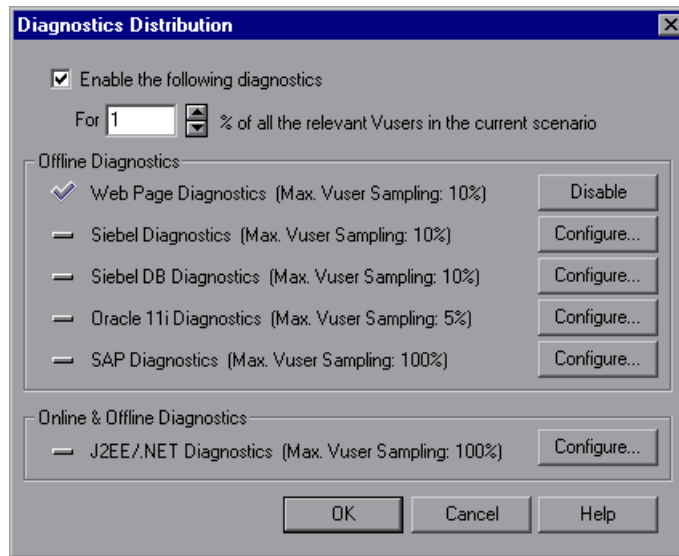
Each time you want to capture Mercury Diagnostics metrics in a load test scenario, you need to configure the Diagnostics parameters for the scenario and select the probes that will be included in the scenario. You configure your scenario for Diagnostics from the LoadRunner Controller.

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, select **Diagnostics > Configuration**.

The Diagnostics Distribution dialog box opens.



- 5** Select **Enable the following diagnostics**.

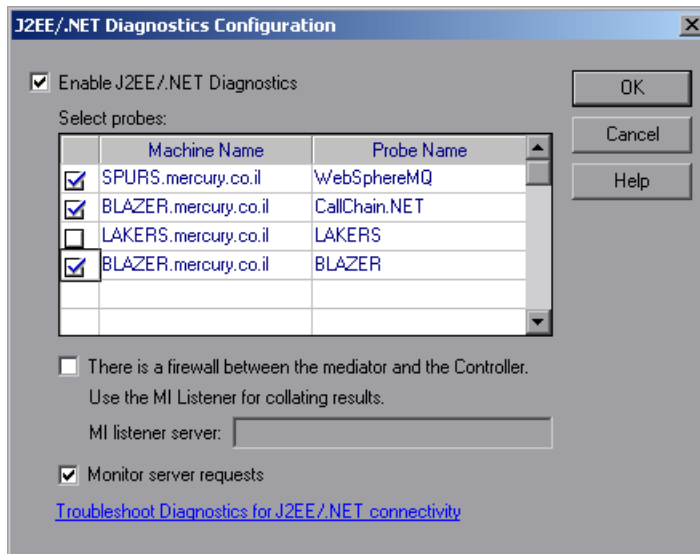
- 6 Set the percentage of Vusers to participate in the Mercury Diagnostics (**J2EE/.NET Diagnostics**) monitoring.

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. You must enable at least one probe in order to save the Diagnostics configuration.
 - To disable a probe for the duration of a scenario, clear the check box.

Note: If you upgraded your Diagnostics installation, probes from existing scenarios may appear with a red status. Clear any probes that appear in red.

- 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:

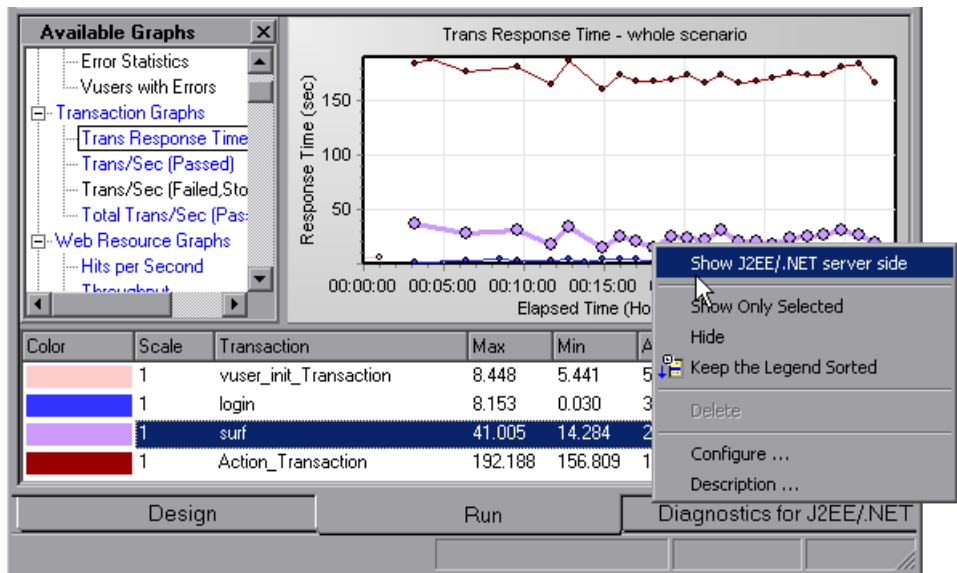
Click the **Diagnostics for J2EE/.NET** tab at the bottom of the LoadRunner window. Mercury Diagnostics opens, displaying the **LR / PC Summary** dashboard view.

The **LR / PC Summary** dashboard view displays monitoring versions of the transactions, server requests, load, and probe views for the current run.

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 “Transactions View” on page 227.

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*.

39

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	444
Configuring Performance Center Load Tests to Use Mercury Diagnostics	445
Drilling Down to Diagnostics Data During a Load Test	450
Analyzing Offline Diagnostics Data Using Mercury LoadRunner Analysis	452

About Viewing Mercury Diagnostics Data in Performance Center 8.1

This section includes:

- “Setting Up Performance Center to Use Mercury Diagnostics” on page 444
- “Viewing Diagnostics Data in Performance Center” on page 444

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 445.

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 450.

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 452.

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.

MERCURY™
Performance Center User: Admin Project: Default

Load Tests

[New Load Test](#)

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**.

Select the Mercury Diagnostics tools that you want to use to identify and pinpoint performance problems in your Web, ERP/CRM, and J2EE/.NET applications.

Enable diagnostics

Perform diagnostics breakdown on % of all relevant Vusers participating in the current Load Test

Offline Diagnostics

- Web Page Breakdown (Max. Vuser Sampling: 10%) Disable
- Siebel Diagnostics (Max. Vuser Sampling: 10%) Configure...
- Siebel DB Diagnostics (Max. Vuser Sampling: 10%) Configure...
- Oracle 11i Diagnostics (Max. Vuser Sampling: 5%) Configure...
- SAP Diagnostics (Max. Vuser Sampling: 100%) Configure...

Offline & Online Diagnostics

- J2EE/.NET Diagnostics (Max. Vuser Sampling: 100%) Configure...

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).

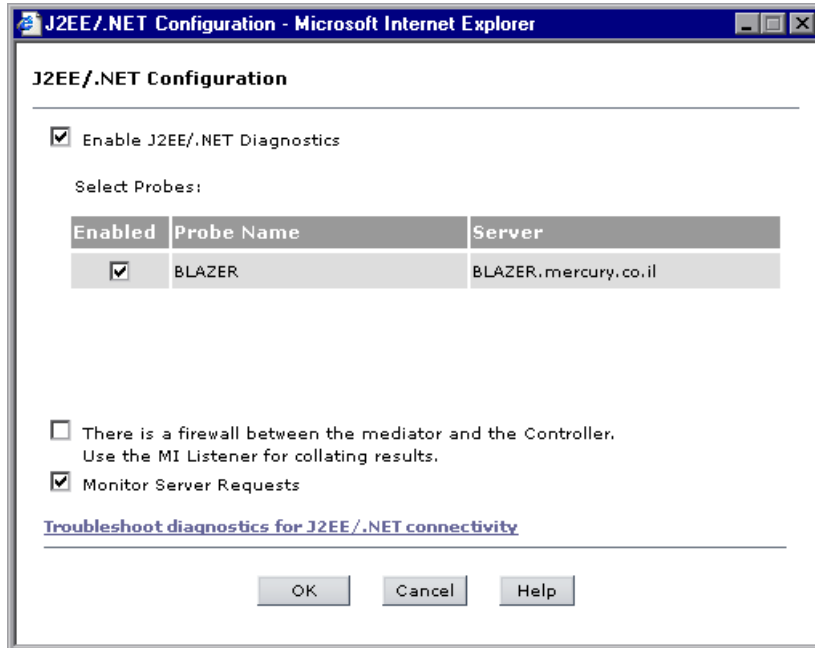
5 Set the percentage of Vusers to participate in the Mercury Diagnostics (**J2EE/.NET Diagnostics**) monitoring.

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*.

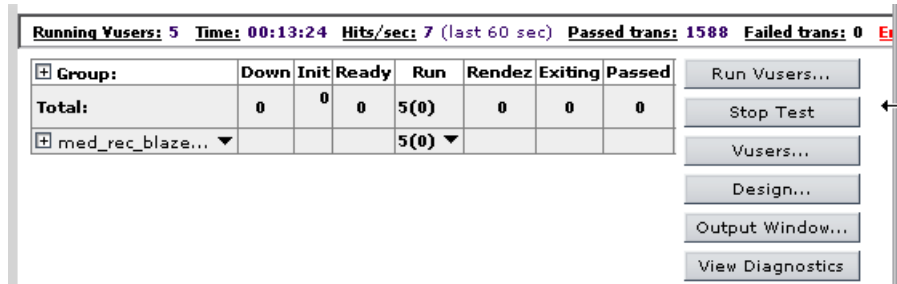
- 11** Click **OK** to confirm your selections and to close the J2EE/.NET Diagnostics Configuration dialog box.
- 12** Click **Save** in the Diagnostics tab to save and validate your settings and complete the configuration.

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.



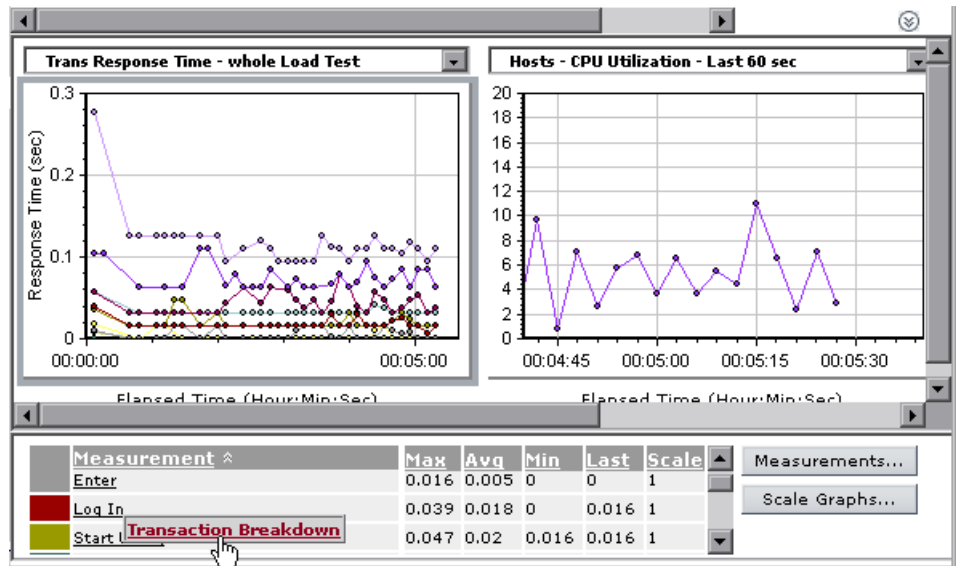
Mercury Diagnostics opens, displaying the **LR / PC Summary** dashboard view.

The **LR / PC Summary** dashboard view displays monitoring versions of the transactions, server requests, load, and probe views for the current run.

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 “Transactions View” on page 227.

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 131
 - resolving problems with 132
- alerts, status view 206
- application memory, monitoring 368

B

- BEA WebLogic views
 - EJB Pooled Resource Contention 298
 - JDBC Connection Status 298
 - JDBC Resource Contention 298
 - Server Threads 298
 - Summary view 297
 - using 297
- Business Availability Center, *See* Mercury Business Availability Center

C

- call profile view 240, 248, 249
 - accessing 254
 - interpreting 254
- call stacks, monitoring 367
- CICS views
 - J2C Connection Pool 304
 - J2C Connections and Faults 304
 - J2C Load 304
 - J2C Usage Time 304
 - J2C Wait Time 304
 - Summary 304
 - using 303
- cross-VM trees 128
- custom views
 - modifying 150
 - sharing 150

D

- data processing, Diagnostics 19
- detail table
 - columns 79, 101, 103
 - customizing 81
 - drilling down into host 160
 - drilling down into layer 167, 245
 - drilling down into portal component 269
 - drilling down into probe 180
 - drilling down into server request 194
 - drilling down into transaction 232
 - graphs 82
 - searching for element in 83
 - selecting row 59
 - viewing element details 84
- detail views 42, 67, 77
 - common features 42
 - common features, view filters 67, 77
- Diagnostics data 11
- Diagnostics Profiler for .NET, *See* NET Diagnostics Profiler
- Diagnostics Profiler for J2EE, *See* J2EE Diagnostics Profiler
- documentation updates xv

E

- elements
 - displaying in graph 71
 - viewing data about 84
 - viewing in detail table 83
- exporting Diagnostics views to HTML reports 34

Index

F

filtering views

- by graphed metrics 47
- by probe group 45, 46
- by time range 48

G

graph

- detail table 82
- displaying elements and metrics in 71
- magnification zooming 49
- metric data 73
- metrics view filter 47
- multiple metrics 71
- resolution zooming 50
- zooming in on metrics 51
- zooming out of metrics 54

H

host

- drilling down into in detail table 160

hosts view 156

- accessing 157
- customizing 158
- interpreting 158

HTML reports, exporting Diagnostics views to 34

I

IBM WebSphere views

- JDBC Connection Status 300
- JDBC Resource Contention 300
- MQ Connection Stats 300
- MQ Message Driven Bean Stats 300
- MQ Queue Stats 300
- Server Threads 301
- Servlet Session Management 301
- Summary view 299
- using 297, 299, 303

instance trees 122

- aggregate trees 131
- cross-VM trees 128
- drilling down into for a server request 195
- insufficient 130
- solving problems with 124, 132

J

J2EE Diagnostics Profiler

- accessing 308
- All Methods tab 328
- All SQL tab 331
- baseline, setting new 352
- call profile 340
- call stacks 309
- Call Tree table 342
- Collections tab 349, 351, 353, 356
- Exceptions tab 333
- garbage collection 312
- Heap Breakdown sampling 361
- Heap Breakdown tab 360
- Hotspots tab 318
- J2EE processing 309
- memory, analyzing using the Heap
 - Breakdown tab 358, 359
- memory, monitoring application 310
- method latency 309
- Metrics Inspector 344
- Metrics tab 320
- metrics, refreshing 311
- metrics, resetting 311
- Server Requests tab 335
- SQL statement details, viewing 332
- Summary tab 314
- Web Services tab 345

L

layers

- drilling down into in detail table 167, 245

- layers view 240, 241, 248
 - accessing 242
 - customizing 243
 - interpreting 243
- life-cycle methods for portlets, analyzing 257
- load view 161, 174
 - accessing 163
 - customizing 164
 - interpreting 164
- LoadRunner
 - analyzing offline diagnostics data 442
 - Diagnostics data, drilling down to
 - during a load test scenario 440
 - Diagnostics data, viewing 435
 - loadtest, configuring to use Mercury
 - Diagnostics 436
 - scenario, configuring to use Mercury
 - Diagnostics 436

M

- magnification zooming 49
- Mercury Business Availability Center
 - Configuration Item Status Alert Report 422
 - Dashboard 413, 414
 - Diagnostics data, viewing 408
 - Diagnostics reports, generating 424
 - Diagnostics screens, accessing 409
 - KPIs Over Time Report 421
 - Transaction Breakdown Reports 425
- Mercury Customer Support Web site xv
- Mercury Diagnostics Profiler for .NET, *See* NET Diagnostics Profiler
- Mercury Diagnostics Profiler for J2EE, *See* J2EE Diagnostics Profiler
- Mercury Diagnostics views
 - accessing 16
 - creating new 144
 - customizing 138
 - deleting 149
 - detail views 42, 67, 77
 - exporting to HTML reports 34
 - modifying custom views 150
 - navigation and display controls 24
 - renaming 148

- saving 142
- sharing custom views 150
- sorting rows 37
- status view 202, 214
- table columns 35
- view group 139
- Mercury Home Page xv
- method latency, monitoring 367
- Metrics Inspector 57
 - reviewing metric details/settings 59
 - thresholds 63
 - viewing metrics 59
- metrics, .Net Profiler 369
- metrics, Diagnostics
 - data in graph 73
 - displaying in graph 71
 - displaying in graphgraph
 - displaying metrics in 62
 - multiple 71
 - reviewing details/settings in Metrics Inspector 59
 - setting thresholds 63
 - threshold alerts 63
 - viewing data in graph 73
 - zooming in on 51
 - zooming out of 54
- metrics, J2EE Profiler 311

N

- navigation/display controls in Mercury
 - Diagnostics views 24
- NET Diagnostics Profiler
 - .NET processing 367
 - accessing 366
 - baseline, setting new 399
 - call stacks 367
 - Call Tree tab 386
 - Exceptions Tab 383
 - Heap tab 400
 - memory, analyzing using the
 - Collections tab 393
 - memory, analyzing using the Heap
 - tab 399
 - method latency 367
 - Methods tab 380

Index

- monitoring application memory 368
- refreshing metrics 369
- resetting metrics 369
- Server Requests tab 372
- snapshots, taking 370
- SQL method calls 379
- SQL tab 377

O

- Oracle views
 - Probes view 294
 - Summary view 294
 - using 293
 - Wait Time view 295
- outbound calls view
 - accessing 170

P

- Performance Center
 - Diagnostics data, analyzing offline 452
 - Diagnostics data, viewing 444
 - Diagnostics, configuring load tests to use 445
 - Diagnostics, drilling down to 450
- portal component view
 - drilling down into in detail table 269
- portal components view 174
 - accessing 267
 - customizing 268
 - interpreting 268
- Portal views
 - about 261, 272
 - Business Process Summary view 262
 - LR PC Summary view 263
 - Portal Components view 264
 - Server Summary view 263
- Probe
 - details, status view 207
 - drilling down into in detail table 180
 - status view 209, 226
- Probe Group
 - status view 208, 226
 - view filter 45, 46

- Probes view 174
 - accessing 176
 - customizing 177
 - interpreting 177
- Profiler for .NET, *See* NET Diagnostics Profiler
- Profiler for J2EE, *See* J2EE Diagnostics Profiler

R

- Remote Function Call (SAP), Diagnostics
 - display 256
- Remote Method Invocation, *See* RMI (Remote Method Invocation)
- resolution zooming 50
- RFC (SAP), *See* Remote Function Call (SAP)
- RMI (Remote Method Invocation), analyzing 257

S

- SAP Remote Function Call, *See* Remote Function Call (SAP)
- SAP views
 - NetWeaver Requests 288
 - NetWeaver Summary 286
 - NetWeaver Threads 289
 - R3 Probes 290
 - R3 Server Requests 291
 - R3 Summary 287
- server request
 - drilling down into in detail table 194
 - drilling down into instance trees 195
- server requests view 187
 - accessing 189
 - customizing 190
 - interpreting 191
- SQL Statements view
 - accessing 184
- standard detail views
 - call profile view 248
 - hosts view 156
 - layers view 240
 - load view 161
 - outbound calls view 170
 - portal components view 266
 - probes view 174

- server requests view 187
- SQL statements view 181
- transactions view 228
- trended methods view 233
- status view 202, 214
 - accessing 204, 221
 - customizing 205, 221
 - customizing tables 205
 - displaying probe details 207
 - drilling down 208, 226
 - host, drilling down 210
 - interpreting 206
 - investigating alert conditions 206
 - Probe Group, drilling down 208, 226
 - probe, drilling down 209, 226

T

- tables in Diagnostics views
 - columns, customizing 35
 - customizing in status view 205
 - header controls 35
 - row, selecting 59
 - sorting rows 37
- threshold alerts 63
- time measurements, Diagnostics 20
- time range, filtering by 48
- transaction
 - drilling down into in detail table 232
- transactions view 228
 - accessing 229
 - customizing 230
 - interpreting 230
- trended methods view 233
 - accessing 235

U

- updates, documentation xv

V

- view filters 44, 67, 77
 - graphed metrics 47
 - probe group 45, 46
 - time range 48
- view group, creating 139

- views, *See* Mercury Diagnostics views

W

- Web Services views
 - about 272

Z

- zooming in on a graph 49
 - magnification zooming 49
 - resolution zooming 50

