HP Diagnostics

for the Windows® and UNIX® operating systems

Software Version: 8.00

User's Guide

Manufacturing Part Number: T6227-90004
Document Release Date: December 2008
Software Release Date: December 2008

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

The information contained herein is subject to change without notice.

Restricted Rights Legend

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Third-Party Web Sites

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

Copyright Notices

Trademark Notices

Java™ and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Microsoft® and Windows® are U.S. registered trademarks of Microsoft Corporation.

Oracle® is a registered US trademark of Oracle Corporation, Redwood City, California.

UNIX® is a registered trademark of The Open Group.

Documentation Updates

This guide's title page contains the following identifying information:

- Software Version number, which indicates the software version.
- Document Release Date, which changes each time the document is updated.
- Software Release Date, which indicates the release date of this version of the software.

To check for recent updates, or to verify that you are using the most recent edition of a document, go to:

http://h20230.www2.hp.com/selfsolve/manuals

This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Or click the **New users - please register** link on the HP Passport login page.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

Support

You can visit the HP Software Support web site at:

http://www.hp.com/go/hpsoftwaresupport

This web site provides contact information and details about the products, services, and support that HP Software offers.

HP Software Support Online provides customer self-solve capabilities. It provides a fast and efficient way to access interactive technical support tools needed to manage your business. As a valued support customer, you can benefit by using the HP Software Support web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage support contracts
- Look up HP support contacts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

Most of the support areas require that you register as an HP Passport user and sign in. Many also require a support contract.

To find more information about access levels, go to:

http://h20230.www2.hp.com/new_access_levels.jsp

To register for an HP Passport ID, go to:

http://h20229.www2.hp.com/passport-registration.html

Additional troubleshooting resources are available on the HP Software Support web site. Choose Help > Troubleshooting & Knowledge Base to access the Troubleshooting page where you can search the Self-solve knowledge base. The URL for this Web site is http://h20230.www2.hp.com/troubleshooting.jsp.

Table of Contents

PART III: UNDERSTANDING DIAGNOSTICS STANDARD VIEWS

PART IV: UNDERSTANDING DIAGNOSTICS SPECIALIZED VIEWS

Welcome to This Guide

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

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 HP Diagnostics.

Part II Common Features and Controls in the Diagnostics UI

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

Part III Understanding Diagnostics Standard Views

Describes the HP Diagnostics standard views.

Part IV Understanding Diagnostics Specialized Views

Describes the HP Diagnostics specialized view groups.

Part V Using the Diagnostics Profiler for Java

Describes how to use the Diagnostics Profiler for Java.

Part VI Using the Diagnostics Profiler for .NET

Describes how to use the Diagnostics Profiler for .NET.

Part VII About Integration, Configuration and Diagnostics Terminology

Describes how to use HP Diagnostics when integrated with other HP Software products. Gives an overview of the Diagnostics Configuration UI. Describes HP Diagnostics terms and metric definitions.

HP Diagnostics Documentation

Your HP Diagnostics application comes with the following documentation:

- **Diagnostics User's Guide and Online Help.** Explains how to use HP Diagnostics to analyze the performance of your enterprise applications. You access the online help for Using HP Diagnostics from the **Help** button in Diagnostics or from the help menu in the integrated HP Software product. You access the PDF version of this guide from the Windows Start menu (**Start** > **Programs** > **HP Diagnostics Server** > **User Guide)**, from the **Documentation** directory on the HP Diagnostics installation disk, or from the Diagnostics Server installation directory.
- **Diagnostics Installation and Configuration Guide.** Explains how to install and configure the Diagnostics components and how to configure Diagnostics for integration with other HP Software products. You access this guide from the Help button in Diagnostics or from the help menu in the integrated HP Software product. You can also access this guide from the Windows Start menu (**Start** > **Programs** > **HP Diagnostics Server** > **Install Guide)**, from the **Documentation** directory on the Diagnostics installation disk, or from the Diagnostics Server installation directory.
- **Readme.** Provides last-minute technical and troubleshooting information about HP Diagnostics. The file is located in the HP Diagnostics installation disk root directory.
- **Diagnostics Probe for Java Installation Quick Start.** Provides the basic instructions for installing the Diagnostics Probe for Java and is available in the **Documentation** directory on the HP Diagnostics installation disk or from the Windows Start menu (**Start** > **Programs** > **HP Java Agent** > **QuickStart**).
- **Diagnostics Profiler for Java Installation and User's Guide.** Describes how to install, configure and use the Diagnostics Profiler for Java. You access this guide online by clicking **Help** in the Diagnostics Profiler for Java.

- **Diagnostics Profiler for .NET Installation and User's Guide.** Describes how to install, configure and use the Diagnostics Profiler for .NET. You access this guide online by clicking **Help** in the Diagnostics Profiler for .NET.

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

Part I

Introduction

1

Diagnostics Product Overview

HP Diagnostics is a composite application monitoring solution you can use to improve the performance of your enterprise applications.

This chapter includes:

Introducing HP Diagnostics

HP Diagnostics is a composite application monitoring, triage and diagnostics solution that is designed to help you improve the performance of your Java, .NET, and other enterprise applications throughout the application lifecycle. HP 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.

HP Diagnostics provides solutions for:

- **Composite Applications** which are a combination of legacy, packaged and new application logic developed on Java/.NET and middleware/ legacy platforms. Diagnostics collects performance metrics from a wide range of composite application components including SAP, WebSphere MQ, Oracle, SQL Server Database, and portal implementations. Diagnostics also collects data from the transport layer supporting HTTP, RMI, WS, and JMS protocols. Diagnostics correlates the performance metrics to provide graphical topology views and cross component transaction views.
- **Java Applications** that run on most of the Java EE 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.
- **Database Applications** including Oracle Database and SQL Server Database.
- **SAP.** HP 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.
- **SOA Services.** Diagnostics detects SOA services, operations and consumers and monitors performance and availability of these services. Diagnostics auto discovers and provides a services topology.

HP Diagnostics has been integrated with HP's Business Availability Center and Performance Center 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 HP Diagnostics to work with one of HP Software's Application Delivery or Application Monitoring products, or you can use it as a stand alone diagnostics tool.

When you install the Java or .NET probes that gather your application's performance metrics, the Diagnostics Profiler is automatically installed as well. The Profiler is an independent diagnostics application that has its own UI. It is accessed either directly on the local machine or through the main HP Diagnostics UI.

Integration with Business Availability Center

HP Diagnostics is integrated with 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 Business Availability Center, you can track the performance status of your applications that are being monitored by HP Diagnostics.

The Diagnostics integration with Business Availability Center allows you to drill down to Diagnostics data from specific Business Availability Center configuration items and reports. You can also generate high level reports in 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 Business Availability Center, see the Chapter 44, "Diagnostics Data in Other HP Software Products."

Integration with LoadRunner and Performance Center

HP 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 HP Diagnostics data for the whole scenario or for a particular transaction. After you have run your scenario, you can use LoadRunner Analysis to analyze offline Diagnostics data generated during the scenario.

For more information about the Diagnostics integration with LoadRunner and Performance Center, see the Chapter 44, "Diagnostics Data in Other HP Software Products."

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.

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 HP Software 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 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 HP Software Representative to enquire about integrating Diagnostics with one of the HP Software Application Delivery or Application Monitoring products listed above.

Diagnostics Profiler

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:

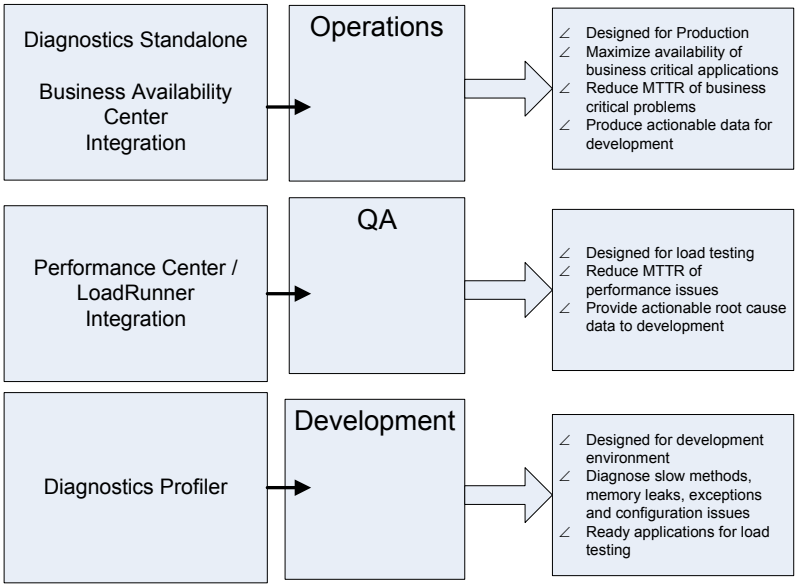
- where time is spent in an application; either processing data or waiting for a response from another part of the application.
- the slowest layers.
- the slowest server requests which are the application entry points.
- outliers to help diagnose intermittent problems.
- threads that may be contributing 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 Diagnostics Profiler for Java, see Chapter 38, "Using the Java Diagnostics Profiler."For more information about using the Diagnostics Profiler for .NET, see Chapter 41, "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

HP Diagnostics consists of the following components:

➤ **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.

Note: The functionality of the Java probe is provided by the HP Diagnostics/ TransactionVision Java Agent (Java agent). The functionality of the .NET probe is provided by the HP Diagnostics/TransactionVision .NET Agent (.NET agent). These agents combine the capabilities of the Diagnostics probe and the TransactionVision sensors into a single component.

The agents are configured to serve as probes in a Diagnostics environment or as sensors in a TransactionVision environment and for combined environments, the agent simultaneously serves as both the probe and the sensor.

- **Diagnostics Collectors.** To gather data from external ERP/CRM environments including Oracle 10g Database, SQL Server, IBM WebSphere MQ or SAP NetWeaver - ABAP system, you install the Diagnostics Collector and define specific instances of these systems to be monitored. Each instance of a collector is represented as a Probe in the Diagnostics UI.
- **Diagnostics Servers.** Responsible for working with the Probes and with other HP Software 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 products with which Diagnostics is working.

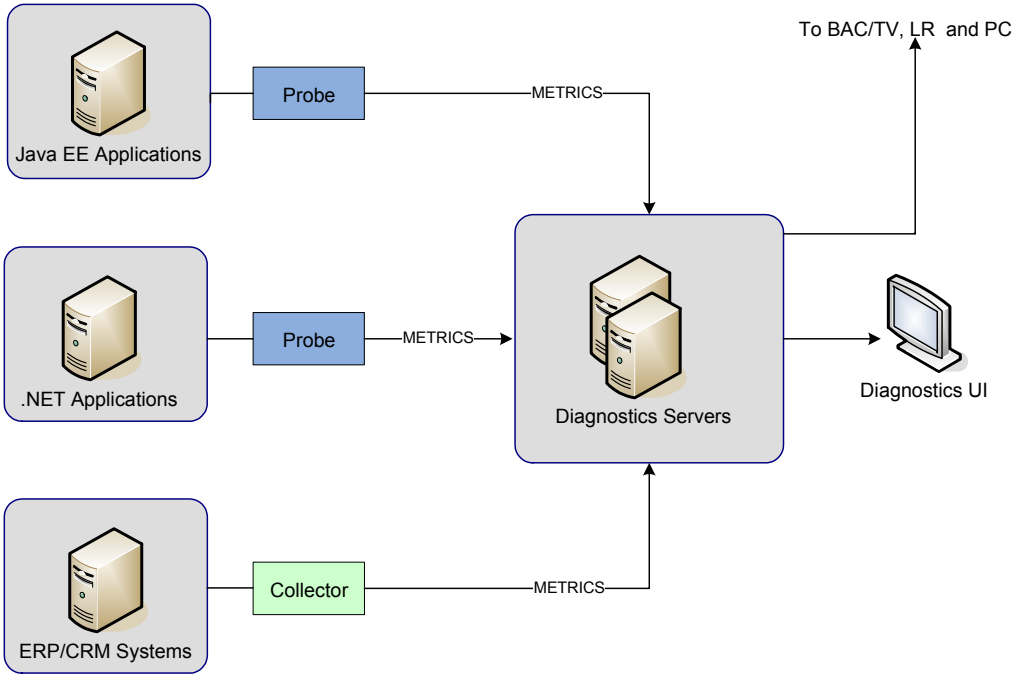
  The Diagnostics Server in Commander mode keeps track of the location and status of the other Diagnostics components, and is the communication hub between the other components.

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 HP Software 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 among Diagnostics components.



In the diagram displayed above, there is one Diagnostics Server in Commander mode connected to a one or more 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 the monitored 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 Business Availability Center or directly from the Diagnostics Server. During a load test, you access the Diagnostics UI from LoadRunner or Performance Center.

You can also access the Java Diagnostics Profiler and the .NET Diagnostics Profiler directly from the probe whose Profiler you want to view or through the Diagnostics UI.

Installation and Configuration Tasks

Your first step in implementing HP Diagnostics is to install and configure the components of Diagnostics. Careful planning and preparation for installing and configuring the components of HP Diagnostics can help you to avoid complications and errors. Some of the key installation and configuration tasks are listed below. See the *HP Diagnostics Installation and Configuration Guide* for complete procedures.

1. **Check the system requirements and installation considerations.**
2. **Install the Diagnostics Server.**
3. **Install and configure the Diagnostics Probe(s) and/or Collectors.**
   - ➤ For a Java environment
   - ➤ For a .NET environment
   - ➤ For Oracle, SAP, SQL Server and MQ environments

4. **For Java Probes, configure the application servers to work with the probes.**
5. **Setup additional instrumentation and do advanced configuration, as needed.**
6. **Optionally, configure HP Diagnostics for integration with Business Availability Center, TransactionVision, LoadRunner or Performance Center.**

Accessing the HP Diagnostics UI

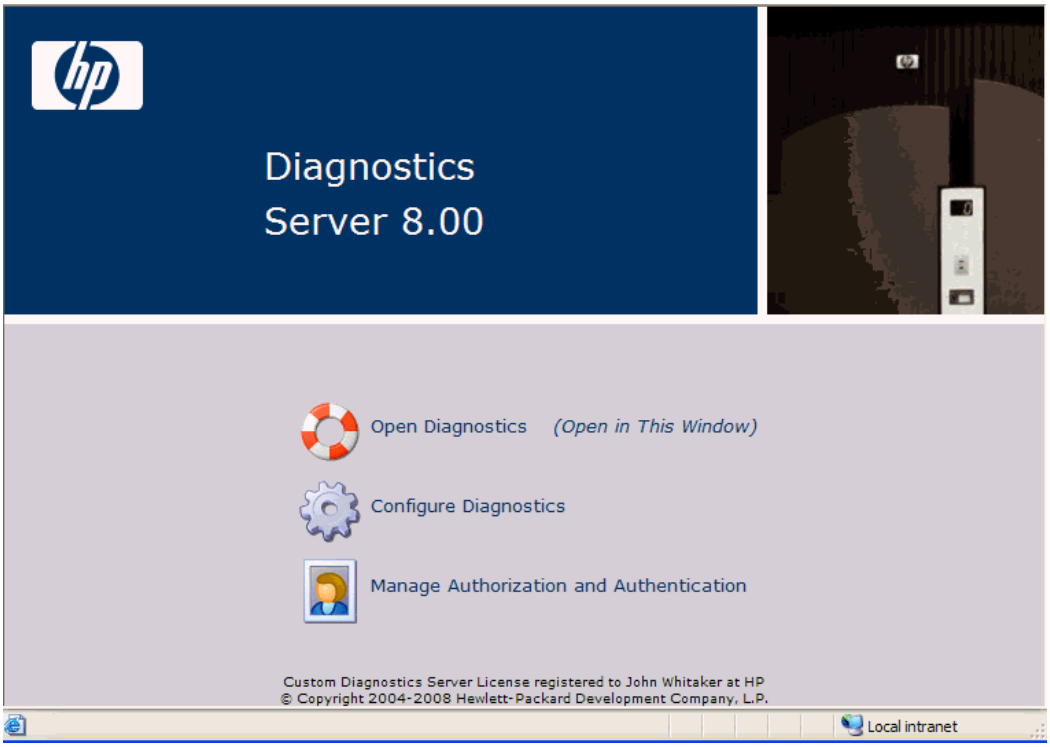
You can access HP Diagnostics views directly from the Diagnostics Server in Commander mode or from the user interface of other HP Software applications that have been integrated with Diagnostics.

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

To access Diagnostics from the Diagnostics Server:

1 In your browser, navigate to http://<diagnostics_server_host>:2006, or select **Start** > **Programs** > **HP 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 *HP Diagnostics Installation and Configuration Guide*.

Click **OK**.

Notes:

- 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 message 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**.

Accessing Diagnostics from Business Availability Center

Note: Before you can access the Diagnostics views from Business Availability Center, you must configure Business Availability Center with the details for the Diagnostics Server. For more information, see the *HP Diagnostics Installation and Configuration Guide*.

You can access HP Diagnostics views from 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 Business Availability Center configuration items and reports.

For more information, see the Chapter 44, "Diagnostics Data in Other HP Software Products."

Accessing Diagnostics from LoadRunner

Note: Before you can use HP Diagnostics with LoadRunner, you must ensure that you have configured LoadRunner with the information for the Diagnostics Server. For more information, see the *HP Diagnostics Installation and Configuration Guide*.

You can access the HP 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 HP Diagnostics data from a particular listed transaction.

For more information, see the Chapter 44, "Diagnostics Data in Other HP Software Products."

Accessing Diagnostics from Performance Center

Note: Before you can use HP Diagnostics with Performance Center, you need to ensure that you have configured Performance Center with the information for the Diagnostics Server. For more information, see the *HP Diagnostics Installation and Configuration Guide*.

You can access the HP 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 HP Diagnostics data from a particular transaction.

For more information, see the Chapter 44, "Diagnostics Data in Other HP Software Products."

Interpreting Diagnostics Data

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

About Layers and Sub-Layers

HP 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 *HP Diagnostics Installation and Configuration Guide*.

For more information on viewing the performance metrics by layer, see "Layers View" on page 363.

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 HP Software 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 have multiple threads executing simultaneously. In Diagnostics the time spent on each thread is counted as opposed to the total overall time, 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 Java/.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

Common Features and Controls in the Diagnostics UI

2

Working with Applications

The Diagnostics UI displays performance data for monitored applications. In the initial Diagnostics Applications window you define and then select the application context for viewing data in the Diagnostics UI.

This chapter includes:

About Applications

An **Application** in Diagnostics is a composite application made up of a logical grouping of other Diagnostics entities, such as probes, hosts and server requests.

You can define applications in Diagnostics which allows you to view Diagnostics data within the context of that application.

An **application group** allows you to organize applications together into groups of related applications. An application group can contain applications, and it can contain other application groups.

By default, there is a predefined application (**Entire Enterprise**) that includes the entire managed environment context. In this application, all Diagnostics views are displayed showing all the relevant entities.

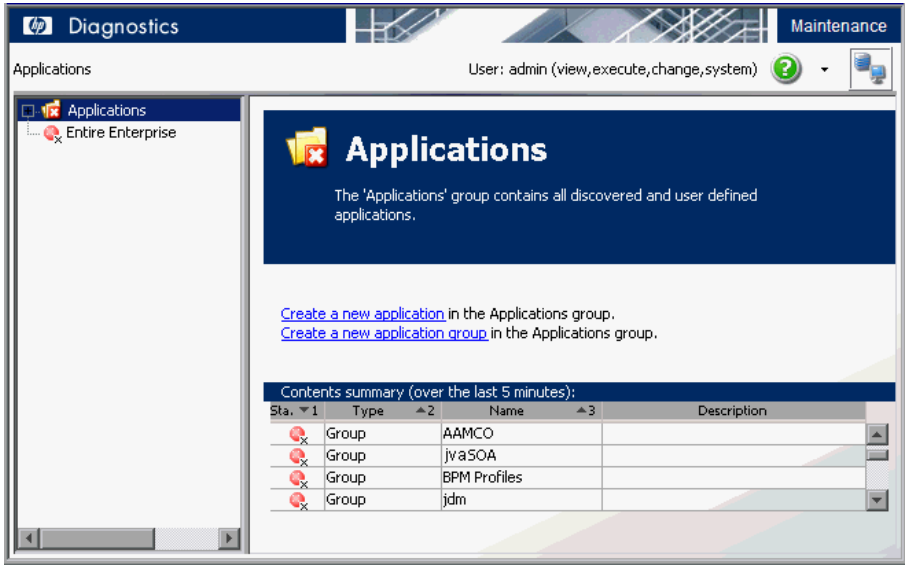
Diagnostics also generates an application corresponding to each **BPM profile** when you have BPM monitors running in your environment.

Note: The Applications window is only available when running Diagnostics by itself or when integrated with Business Availability Center, and not when viewing Diagnostics from LoadRunner or Performance Center.

Description of the Applications Window

The Applications window is the first screen you see when you select **Open Diagnostics** in the main Diagnostics Server UI window.

When you select the top-level group (named **Applications**) in the navigation pane, the details for this application group are displayed in the right pane as shown in the example screen shot that follows:



Navigation Pane

The Applications window has a navigation pane that lists the application groups and applications.

The status of the application group or application is displayed next to the entity in the tree.

The default application group, named **Applications**, contains user-created application groups and applications that are detected automatically. BPM profiles are detected automatically, and listed as applications in the **BPM Profiles** application group.

The pre-defined application, named **Entire Enterprise**, contains all components known to Diagnostics. You select the Entire Enterprise application to monitor the entire environment. As a result, data in the Diagnostics views is not filtered by application.

To view Diagnostics data in the context of an application, select the application in the navigation pane. As a result, the Diagnostics views are filtered to show only the entities that are part of the selected application. You can select an application in several ways.

To select the pre-defined application Entire Enterprise:

1. Click Entire Enterprise in the navigation pane.
2. Double-click Entire Enterprise to open the application summary view or select an entity from the **Application Contents** list to go directly to the related Diagnostics view. Diagnostics views shows data for all components known to Diagnostics, allowing you to monitor the entire enterprise.

To select an application from an application group:

1. In the navigation pane, open the application group named **Applications** and navigate to select the application you want to monitor.
2. Double-click the application to open the Application Explorer view or select an entity from the Applications Contents list to go directly to the related Diagnostics view. Diagnostics views are filtered to show only the entities that are part of the selected application.

Application Group View

Selecting an application group in the navigation pane opens the Application Group view in the details pane. You can also double-click on an application group in the details pane to open the application group view.

You will only see the application groups that contain applications that the user you are logged in as, has permissions to view.

In the example shown below, the application group jvaSOA is selected and the details for this application group are displayed in the right pane.

If the user you are logged in as has the appropriate permissions, you can **Create a new application group** and **Create a new application** from the Application Group view. See “Application Permissions” on page 47 and “Creating a New Application” on page 50 for details.

The **Contents Summary** lists the applications in the application group selected in the navigation pane.

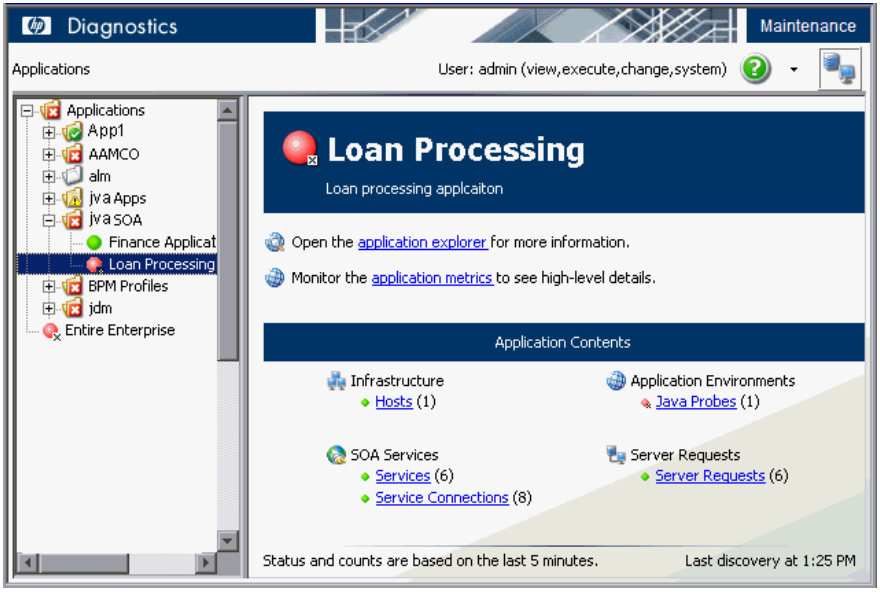
The following information is displayed in the Contents Summary:

- **Status.** Indicates how the application is performing. The status of an application is based on the status of the entities that comprise the application.
- **Type.** The type can be either application or group (a group contains other groups or applications).
- **Name.** User-specified name of the application.
- **Description.** User-specified description of the application.

Double-clicking on an application in the Contents Summary table replaces the details pane with the Application view.

Application View

When you select an application (or the **Entire Enterprise** application) in the navigation pane, the Application view is displayed in the details pane.



The Application view has a link to the **application explorer**, which (by default) takes you to the Application Explorer view in Diagnostics. There is also a link to **application metrics**, which takes you to the Application Metrics view in Diagnostics for high level details about an application.

Double-clicking on an application in the navigation pane opens the Application Explorer.

The **Application Contents** lists the entities that are part of the selected application. It shows which entities were active in the last five-minute interval. Server requests or transactions may not display if they were not active during this time period.

To open a specific Diagnostic view, select the link for an entity displayed in the details pane (for example, Hosts, Probes, Server Requests). For example, selecting the Java Probes link opens the Probes view. If the application is configured to not show the view group which contains the view for this entity type, the link will not be selectable.

If you've selected the Entire Enterprise application, all data is shown in the Diagnostics views. If you've selected a single application, the Diagnostics views show data in the context of the selected application.

A **discovery process** runs every five minutes to see which entities are present in the application and to determine the status and count, which is displayed in the **Application Contents** as a red, yellow or green status indicator. If new entities appear, they are added to the list. If entities are not found, they are removed from the list.

BPM Profile applications have transactions listed in the Application Contents. User-defined applications do not.

Using the Applications Window

From the Applications window, you can select an existing application and go to a Diagnostics view in the context of the selected application. You can also select the Entire Enterprise application and view all data displayed in the Diagnostics views.

From the Applications windows, you can create, update, and delete applications. See "Creating Application Entities in Diagnostics" on page 47 for details.

A typical process for working with applications in Diagnostics is as follows:

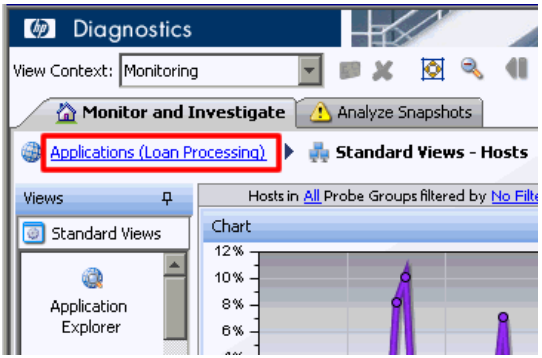
1. Define application groups and applications (you need Execute permissions).
2. Select **Entire Enterprise** in the Applications window, and enter the Diagnostics views.
3. Add entities to the applications you created. Many entities are automatically discovered and added to the application based upon what you add. For example, if you add a particular WebSphere JVM to an application, the Host it is running on as well as the database it communicates with are also added.
4. Click on the **Applications** link in the breadcrumb to go back to the Applications window.
5. Double-click the application you want to monitor, and enter the Diagnostics views.

6 You may want to check that the correct view groups are shown/hidden for each application and make any changes necessary. You can right-click in the View bar and select **Open a View Group** to see hidden view groups.

7 At this point, when users select an application they will be viewing application-specific data in the UI.

Returning to the Applications Window

Once you are in the Diagnostics views, the breadcrumb trail shown at the top shows you the context for the information that is presented in the view. You can return to the **Applications** window from the Diagnostics views by clicking the **Applications** level in the breadcrumb trail.



The first level displayed in the breadcrumb shows the application context of the Diagnostics views. If you select to view the **Entire Enterprise**, you see All Applications in the Applications window. If you select a specific application, you see the name of the application. Selecting the first-level link in the breadcrumb takes you back to the Applications window.

The last level displayed in the breadcrumb is not a link because it represents the current Diagnostics view.

Creating Application Entities in Diagnostics

When you create an application in Diagnostics it functions as a container for other non-application entities. These containers enable you to group entities, and to associate them with an application. For example, you may want to separate the entities used to implement and support the Human Resources application from the entities used for the CRM application.

Application Permissions

By default, any user with execute permissions can create an application (or application group) in Diagnostics.

After an application has been created, permissions for the application can be assigned to users or roles.

See the table below for details about the application permissions (view, edit, or edit screens) required to perform various actions, such as creating or renaming an application.

Application permissions are set in the Applications window. See *HP Diagnostics Installation and Configuration Guide* (Appendix B) for details on setting user permissions at the enterprise or probe level.

Area and Action	Enterprise Permissions			Application Permissions		
	view	execute	change	view	edit	edit screens
User defined applications						
Create application		X				
Delete application		X			X	
Rename application		X			X	
Change application path		X			X	
Modify application permissions		X			X	

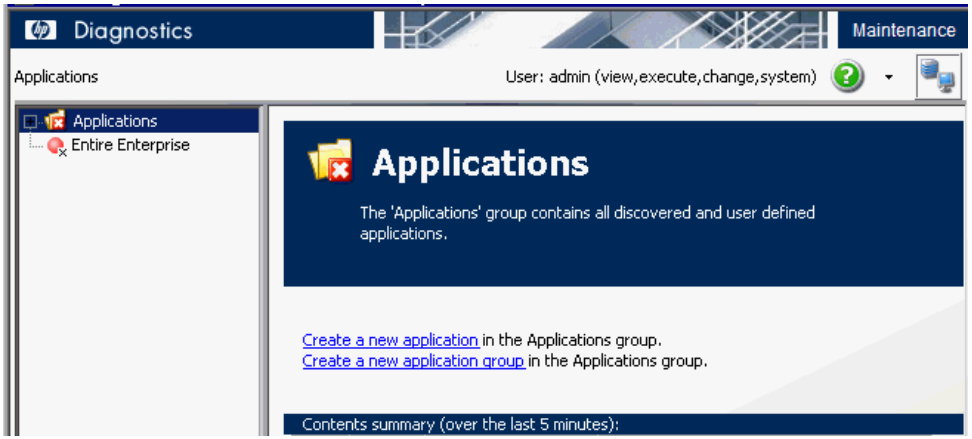
Area and Action	Enterprise Permissions			Application Permissions		
	view	**execute**	**change**	**view**	**edit**	**edit screens**
Edit custom applications screen	X					X
Add entity to application	X				X	
Remove entity from application	X				X	
Edit entity properties (thresholds, comments, etc.)		X		X		
View application	X			X		

Creating a New Application Group

If the user you are logged in as, has the appropriate permissions, you can create a new application group in the top-level application group named **Applications**. You can also create a new application group within another application group.

To create an application group, select the default application group named Applications in the navigation pane, or select an existing application group from the tree in the navigation pane.

The Application Group view is displayed in the details pane as shown below:



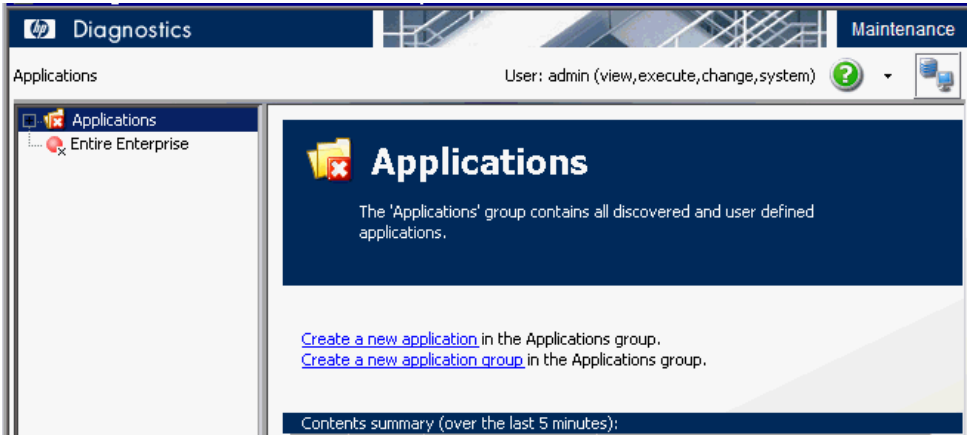
Then, in the Application Group view, select **Create a new application group**. Alternately you can select the top-level Applications group or another application group in the navigation pane and right-click to select **Create New Group**.

The Add Group dialog box opens.

Enter a name for the application group.

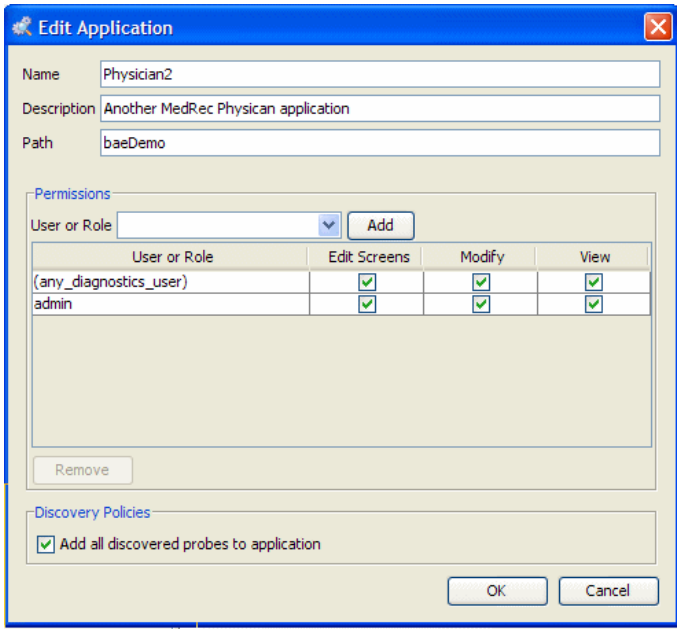
Creating a New Application

If the user you are logged in as, has the appropriate permissions, you can create a new application. First select an application group, either the default application group named Applications in the navigation pane, or select an existing application group from the tree in the navigation pane. The Application Group view opens in the details pane, as shown below:



Then, in the Application Group view, select **Create a new application**. Alternately you can select the top-level Applications group or another application group in the navigation pane and right-click to select **Create New Application**.

The Create New Application dialog box opens, as shown below.



Enter the following to define a new application:

- **Name.** Enter a name for the application you want to create.
- **Description.** Enter a description for the application.
- **Path.** The path associates the application with an application group. The path contains the application group name you selected when creating the application. If you want to move (not copy) an application from one group to another, you can change the path to a different application group name.
- **Permissions.**
  - **User or Role.** Select users and roles to associate with this application, and associate permissions to each user and role.

    The users and roles you select are allowed access to this application's data in the Diagnostics views.

    For each user or role, select the permissions allowed:

View. View applications, and edit application properties (also requires Enterprise permission set to change).

Modify. Delete applications, rename applications, modify applications, and add or remove an entity from an application.

Edit Screens. Make edits using the Application Overview screen.

- **Discovery Policies.**
  - **Add all discovered probes to application.** By default (when checked) the discovery process will add additional probes to the application based on connections to the probes that are currently members of the application. The discovery process runs every five minutes to see which entities are present in the application and to determine the status and count. See "Parent Entities Added" on page 54 for details on what is automatically added to the application during discovery.

    You can disable discovery of probes within this application's topology by de-selecting this checkbox. When de-selected, discovery will not add probes to the application based on the connections to currently defined probe members. You can manually add the probes you want to be included in this application.

    Alternately you could allow discovery to add all the probes and then use the Remove/Exclude feature, but this won't prevent new probes from being automatically added to the application.

What actions the user or role is allowed to perform in the Diagnostics UI is determined by a combination of the following:

- User and role permissions set for applications in the Add Application dialog box described above. See "Application Permissions" on page 47.
- User and role permissions set for the enterprise server and probes in the Security dialog box, accessed from the Diagnostics welcome screen by selecting Configure Diagnostics and then selecting Security. See the *HP Diagnostics Installation and Configuration Guide* for details about setting user permissions at the enterprise or probe level.

After you add a new application, you are able to add entities (for example, probes and server requests) to this new application from within the Diagnostics view. For details, see "Adding and Removing an Entity from an Application" on page 53.

Editing Application Properties

If the user you are logged in as, has the appropriate permissions, you can edit an application's properties by selecting the application in the left pane, and right-clicking **Edit Application**.

In the Edit Application dialog box, you can change the application's name, description, path, user, and role. To change permissions for a user or role for this application, check or uncheck the boxes for Edit, Modify, or View. See "Creating a New Application" on page 50 for details on these properties.

Deleting Applications

In the left pane of the Applications window, right-click an application, and select **Delete Application**.

After all the applications in an application group have been deleted, the application group is deleted the next time you exit the Diagnostics UI.

The Application Context within Diagnostics Views

If you select an application in the initial Applications window, the data displayed in the Diagnostics Views is specific to this application (and the entities associated with it).

Adding and Removing an Entity from an Application

In various Diagnostics views, you can associate entities with defined applications by selecting the entity, and then right-clicking **Add to application**. The user you are logged in as, must have the appropriate permissions for you to be able to add or remove entities from an application.

Note: Whenever you add or remove entities from an application it may take several minutes for the changes to take effect and for you to see the change.

You can associate the following entities with an application:

- Hosts
- Probes (Java, .NET, Oracle, SAP, SQL Server, MQ)
- SQL Statements
- Server Requests
- Web Services
- Aggregate Requests
- Outbound Calls
- Portlets
- Trended Methods
- Probe Connections
- MQ Queues
- MQ Channels
- Resources
- Collections

Parent Entities Added

When you add an entity to an application, the 'parent' entities are also added. Some examples of this rule are shown below:

- If you add a probe to an application, the associated host and connected probes (and the connections) will also be added.

  Connected probes are not added when the **Add all discovered probes to application** checkbox is de-selected when creating the application.
- If you add a server request to an application, the probe it is on is also added (which adds the host the probe runs on, and the connected probes, which adds their hosts and their connected probes, and so on).
- If you add a Web service to an application, the associated Web service operations are added and the connected services are also discovered (which includes probes and the host the probe runs on, and so on).

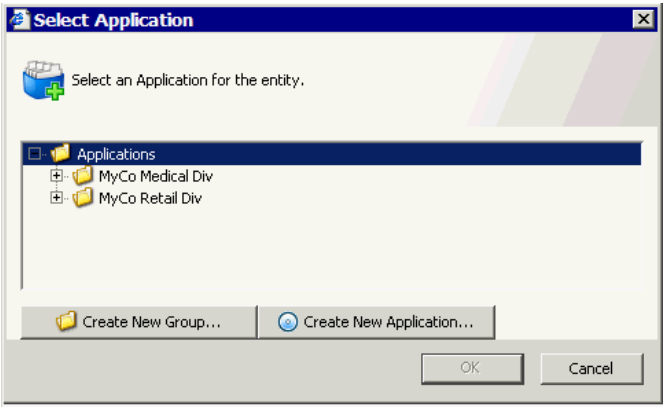
- If you add an aggregate request to an application, any server requests that match it are also added. The server requests add the probes they are on, which adds the hosts they are on, etc.
- If you add a trended method to an application, nothing else is added unless it is a trended method beneath a probe and then the probe (and host and so on) is also added.
- If you add a probe topology link (probe connection) to an application, the endpoint (probe) is automatically added (and the probes that probe points to).
- If you add an MQ queue or MQ channel to an application, the MQ queue manager is also added.
- If you add a resource or collection to an application, the probe it is running on is also added (which adds the hosts, and so on).
- If you add an Outbound call, the Host and Probe are also added.
- If you add a Portlet, the Host and Probe are also added.

Note: When you have BPM monitors running in your environment, Diagnostics automatically generates an application corresponding to each **BPM profile**. Transactions and transaction related entities in that profile are added to the automatically created BPM profile application

---

Add to Application

Selecting **Add to Application** from the entity pop-up menu or from the Common Tasks menu displays the Select Application dialog box. You select an application from this list and the entity will be added to that application.



An entity can be added to any number of applications.

You can select multiple entities in the graph entity table to add or remove them from an application. See “Selecting Multiple Rows in a Table” on page 111 for how to select multiple rows. Typically, when selecting multiple entities you should select groups of entities you want to remove or add to the same application.

If the user you are logged in as, has the appropriate permissions, you can also select **Create New Group** and **Create New Application** from this dialog box. When you create a new group or application, that group or application will automatically be selected in the dialog box.

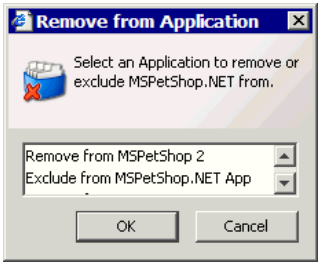
Note: If you create a new application (or multiple applications) from this dialog box, but then cancel out of the dialog box, the created applications will still exist.

Remove or Exclude from Application

Selecting **Remove from Application** from the entity right-click menu or from the common Tasks menu displays the Remove from Application dialog box. You can select an application to remove or exclude the entity from. The user you are logged in as, must have the appropriate permissions for you to be able to remove or exclude entities from an application.

The easiest way to remove entities you do not want is from the Topology view. Select the entity in the Topology view and right-click to select **Remove from Application**.

You cannot remove or exclude an entity if it is a parent of an entity that was manually added. For example, if you add the server request **/login.do**, you will not be able to remove or exclude the probe where this server request is running since this is the parent of the server request entity. This limitation exists to ensure that you will always be able to navigate to anything you have added to your application.



In the Remove from Application dialog box you may see Remove from or Exclude from actions. Remove will remove the entity and also any entities that were added to the application because of their relationship (parent/child) to the removed entity. Exclude will just remove the entity. In essence, the action is the same, but the result depends on the way the entity was added to the application initially.

You can remove entities from applications in either the Entire Enterprise or a selected application context.

Viewing Layers

When viewing layers for an entity, no filtering by application is performed.

Viewing Entities Under Probes in an Application Context

Only entities that were added to an application appear when you are viewing that application. When drilling down from a probe, however, you see everything running on that probe. For example, if you drill down from a probe to the server requests for that probe, you see all server requests. If any server requests have been added to the application explicitly, those server requests have a check mark in the **App?** column of the graph entity table.

Application Context in the Views Sidebar

When you enter the Diagnostics views for a selected application, the View sidebar does not show any of the custom view groups (for example, BEA WebLogic or IBM WebSphere). By default, when you create an application, the only views that appear in the View sidebar are the Standard Views and shared views. You can right-click the View sidebar to select **Open a View Group**.

Any view (or view group) that is created in the context of a particular application is shared with all other users viewing that application. This shared viewing is different from viewing the Entire Enterprise, where any views or view groups created are private to the user who created them.

Custom Views

When a custom view is created within an application (excluding the Entire Enterprise default application), the custom view is available to all users with permissions to view the application.

3

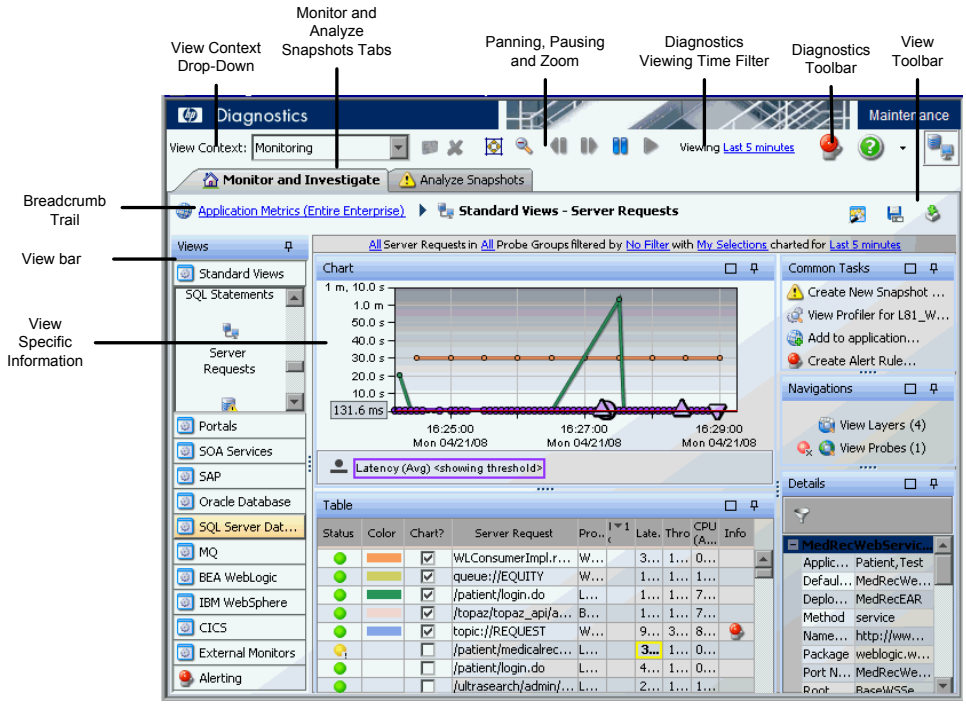
Common Controls in the Diagnostics UI

The Diagnostics UI uses common controls in order to make navigating throughout the UI and drilling down to the data you need easier.

This chapter includes:

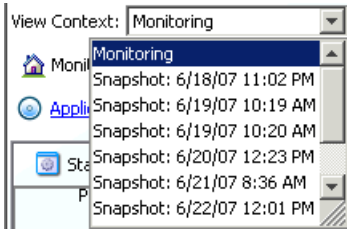
Common Diagnostics View Controls

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



View Context Drop-down

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



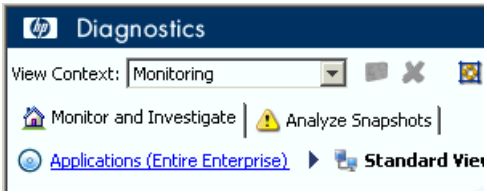
To view performance metrics for your application's current processing, select **Monitoring** from the **View Context** drop-down. Selecting **Monitoring** places Diagnostics in Monitoring mode. When in monitoring mode, the Monitor and Investigate tab presents the Diagnostics views with the application's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

To view the performance metrics for a particular snapshot, select the snapshot from the **View Context** drop-down. Selecting a snapshot from the **View Context** drop-down puts Diagnostics in Snapshot Analysis mode. When in Snapshot Analysis mode, the Monitor and Investigate tab presents the Diagnostics views with the performance metrics for the time of the selected snapshot.

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

Monitor and Analyze Tabs

Diagnostics displays its views on two tabs: the Monitor and Investigate tab and the Analyze Snapshots tab. Diagnostics displays different information in these tabs depending on whether you have selected Monitoring mode or Snapshot Analysis mode. The mode is determined by your selection from the **View Context** 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 Snapshots tab is not used when in Monitoring mode and instead contains instructions for creating a new snapshot.

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

Panning, Pausing and Zoom Controls

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. The **Zoom Out** and **Fit Data to Size** buttons allow you to modify the time filter and granularity of the data display.

➤ Click **Fit Data to Size** to reset any zooming in changes so that all points will be shown on the graph and the scroll bar is no longer displayed.

➤ Click Zoom Out to increment the time filter to the next granularity, for example, five minutes to 20 minutes.

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

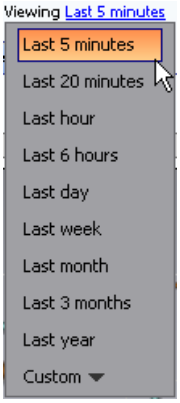
➤ Click **Pan Left** to smoothly scroll the graph from the current time period to the time period with the same duration that immediately preceded the one displayed.

➤ 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.**

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.

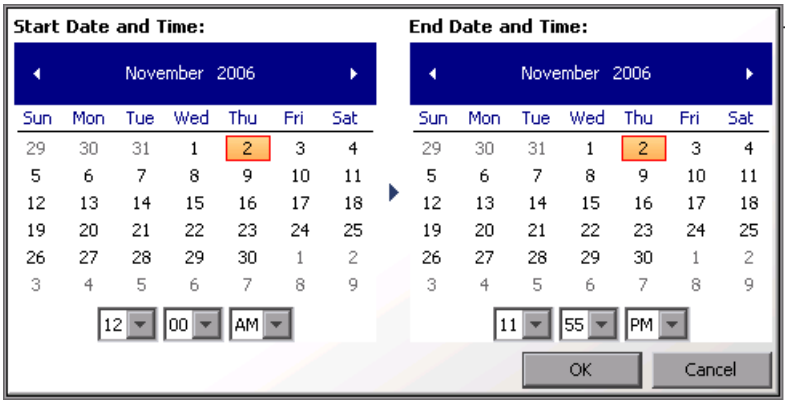


The time range that you select from the **Viewing Time** filter is used in all of the Diagnostics views unless you have locked a view for a specific time period. See “Filtering the View by Time” on page 91 for details on locking views.

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:



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 five 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 **Last 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 five 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 up to twice the duration of the selected time range.

For example, when you select **Last 20 minutes**, the metrics displayed in the view will include data for the previous 20 to 40 minutes.

You can use the **Fit Data to Size** button to see the full time range in the graph. See "Panning, Pausing and Zoom Controls" on page 63.

Using the Viewing Time Filter in Snapshot Analysis Mode

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

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, "Working with Alerts."

Turn On/Off Tooltips. A toggle button that allows you to control whether tooltips are displayed throughout the UI for your logon user.

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:

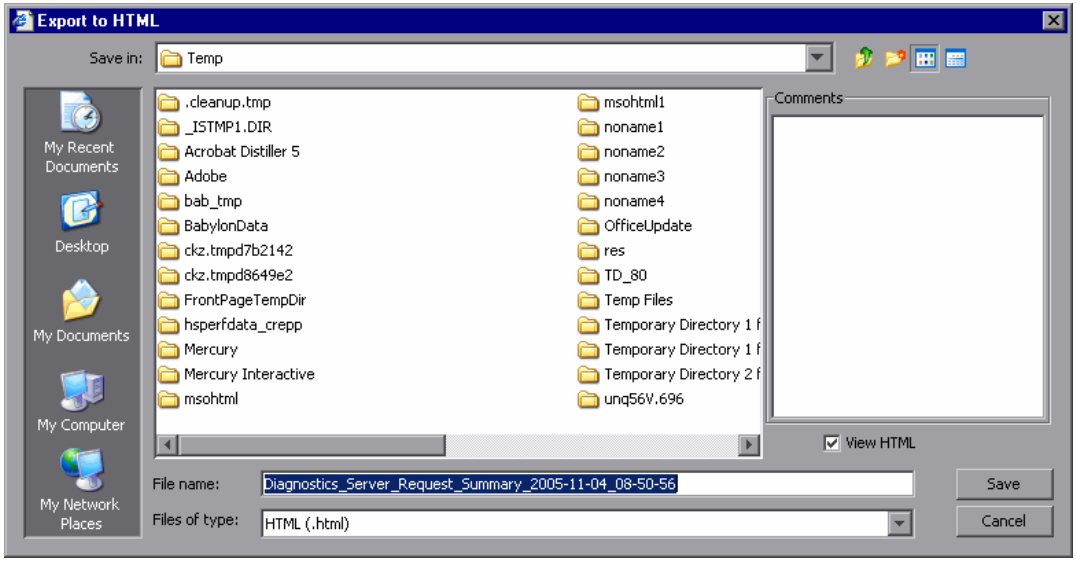
Reset Screen Layout. After rearranging the panes in the View you can select the Reset Screen Layout button to return the display to the default layout.

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.”

Save this view as an HTML Web Page. This selection allows you to save the currently displayed metrics to a report that can be recalled for viewing or sharing with others.

To save view information to an HTML Report:

1 Click **Saves this view as an HTML web page**. 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.

Note: If you select **Saves this view as an HTML web page** from a tabbed view, data from only one of the tabs is saved. For example, with the Topology view, data from the Probe Connections tab is saved.

You can use the **Data Export** function to export performance metrics directly from the Diagnostics Time Series database (TSDB), which is the repository for all persistent Diagnostics data. For more information on data export, see the *HP Diagnostics Installation and Configuration Guide*.

Diagnostics does provide an advanced Query API which you can use to create a more customized HTML dashboard. An example is provided in the **/contrib** directory of the use of the Diagnostics Query API. A readme is provided along with sample code. The Query API is accessed from the Diagnostics Welcome screen by selecting Configure Diagnostics, and then in the Components window selecting **query**.

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 first level displayed in the breadcrumb shows the application context of the Diagnostics views. You will see either Applications (Entire Enterprise) if you selected to view the Entire Enterprise or the name of the application you selected in the Applications window. Selecting the first level link in the breadcrumb takes you back to the Applications window.

The last level displayed in the breadcrumb is not a link because it represents the current Diagnostics view.

For information on navigating from one view to another, see “About the Common Tasks and Navigations Panes” on page 115.

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.

Note: If you have selected an application prior to entering the Diagnostics views, you will see just the Standard Views view group. You can right-click in the View Bar and select **Open a View Group** to unhide additional view groups.

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 Specific Information

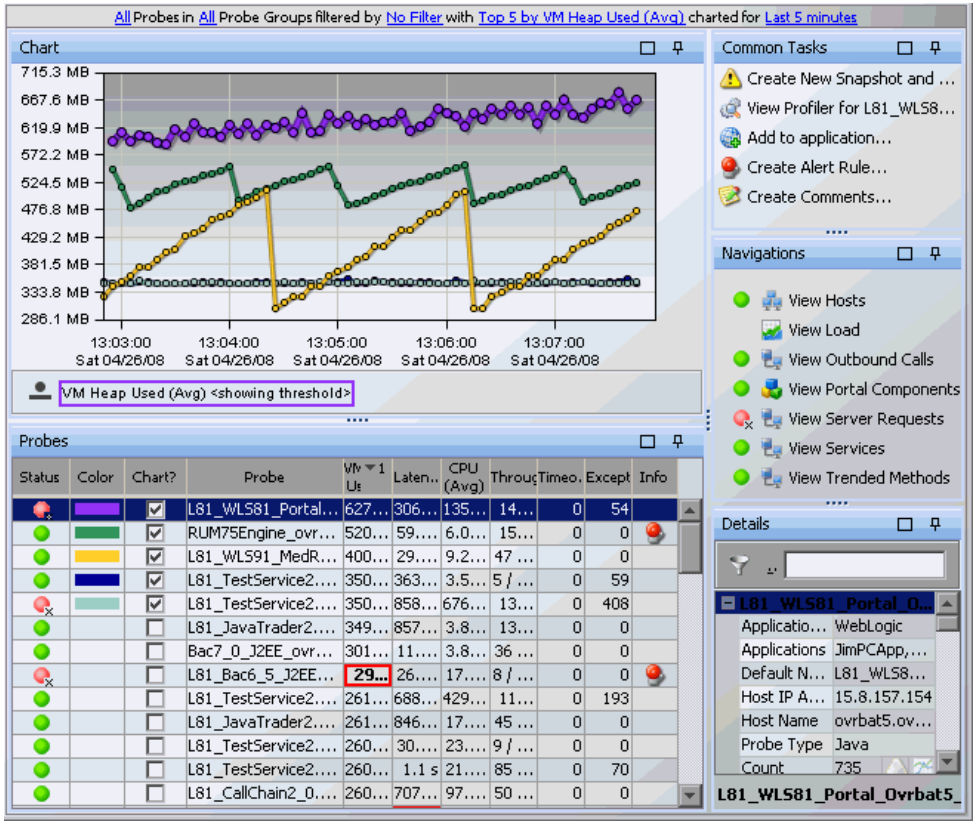
Diagnostics comes with a set of predefined **Standard Views** where each view focuses on a specific aspect of your application's performance.

Diagnostics also provides **Specialized Views** focused on specific types of applications such as Oracle Databases or BEA WebLogic.

The views show the performance metrics in both tabular and graphical formats. Diagnostics presents the performance metrics for your applications in a variety of view layouts including: detail layouts, dashboard layout, status layout, call profile layout, and topology layout. Some examples of the different view layouts are shown on the following pages.

Detail Layout

The Detail layout has a standard graph, table and details. Diagnostics views using the detail layout 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:

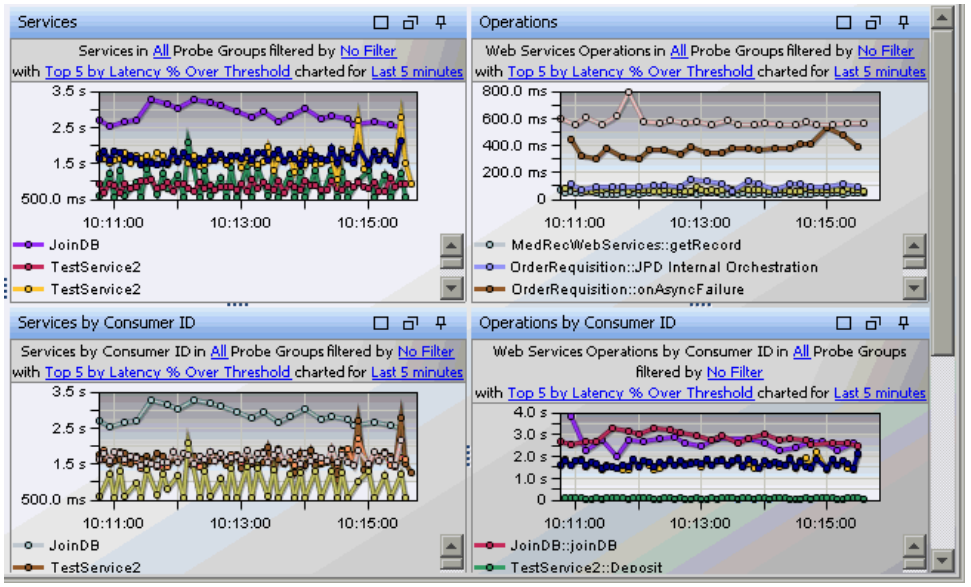


For more information on the features and controls of the detail layout, see Chapter 4, “Working with Diagnostics Data Displays”

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

Dashboard Layout

The 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 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 SOA Services - Service 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.

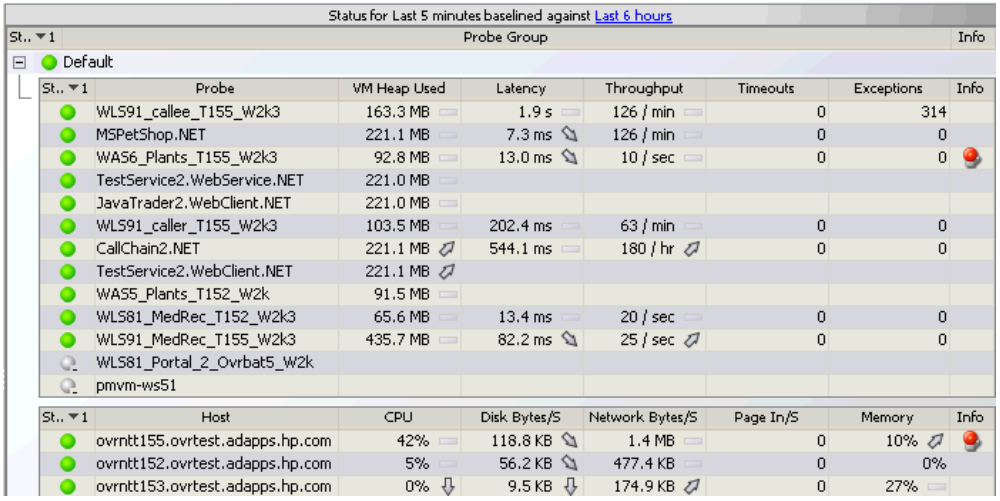
When you hold your mouse pointer over the items in the legend in the concise views, Diagnostics displays a tooltip with additional information.

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 breadcrumb to indicate that you navigated from the Dashboard view.

Status Layout

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:

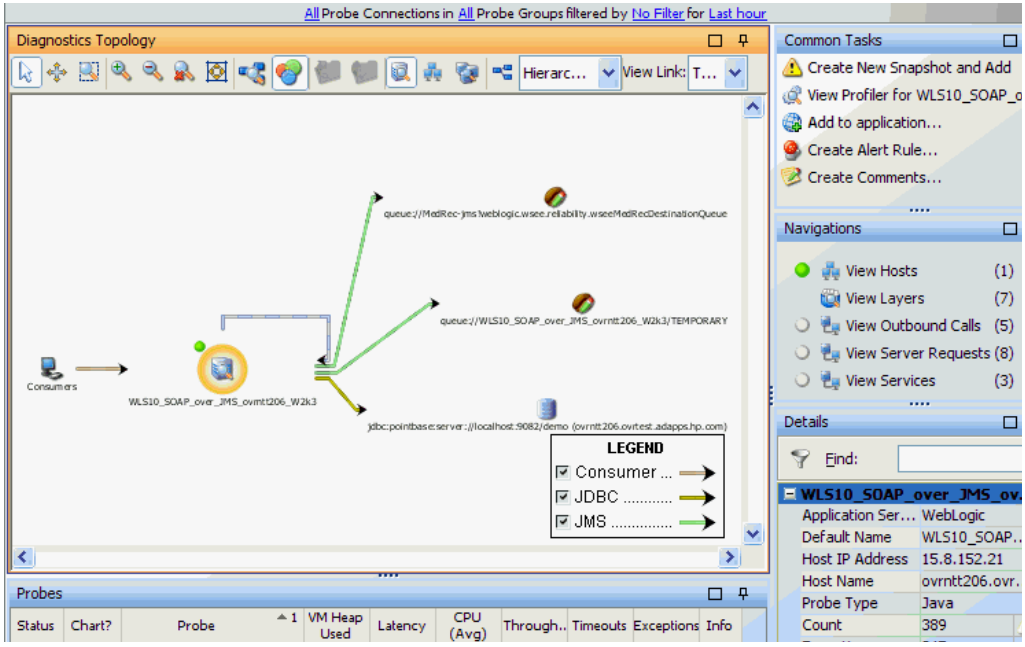


For more information on the features and controls of the Status view, see Chapter 21, “Status View.”

Topology Layout

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. For more information on the Topology view, see Chapter 6, “Working with the Topology Views”.

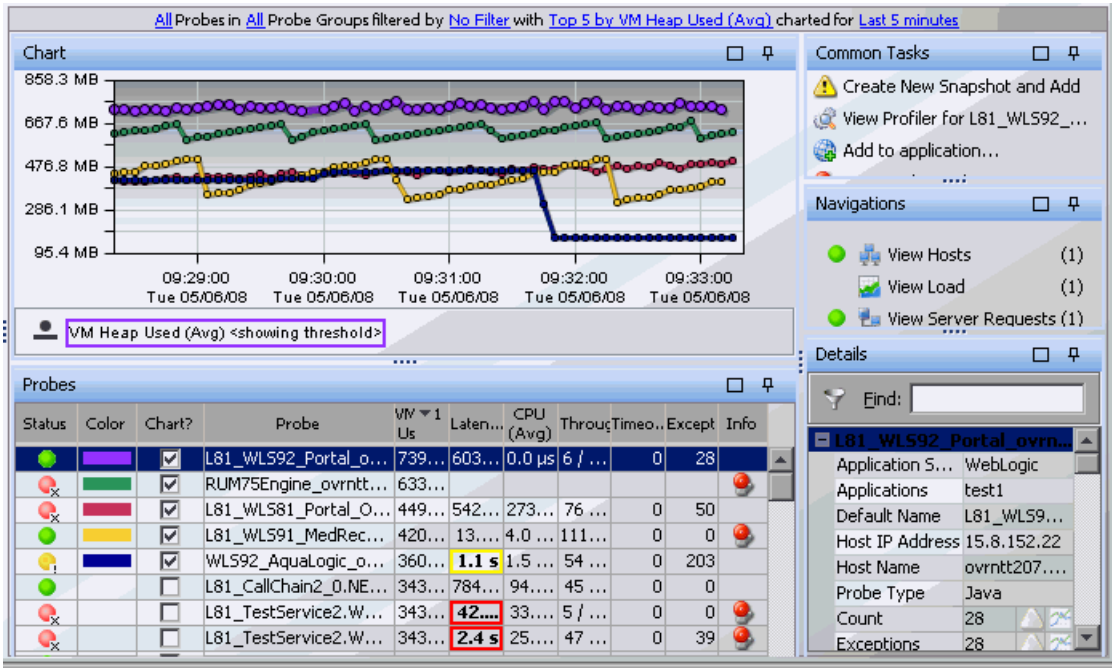
The following example shows the topology graph for a selected probe:



Resizing and Docking Windows

You can customize the look of Diagnostics views by repositioning and rearranging (docking) the different display panes to suit your needs.

Most Diagnostics views are composed of several different panes. For example, the standard view for probes is made up of 4 panes: a graph pane, a table pane, a details pane and a common tasks and navigations pane.

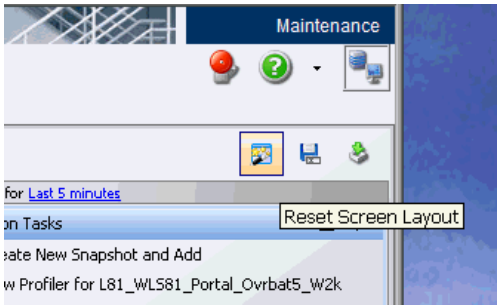


Many of the views have splitter controls that enable you to change the proportions of different areas of the view. They allow you to resize parts of the view so that areas of less interest are minimized to provide more area for the information that you want to see.

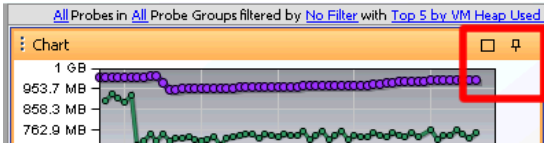
You can also select any of these panes and move and dock them independently. Select the pane by clicking at the top of the pane, the pane is then highlighted in yellow.

Click and drag the highlighted pane to move it wherever you choose in the window. You will see an outline of the pane displayed as you move it to different positions in the windows. Once you drop the pane in place, the other panes will resize to accommodate the new location and size of this window pane.

After rearranging the panes in the View you can select the **Reset Screen Layout** button to return the display to the default layout.



You can use either the Maximize/Restore icon or the Autohide icon to position the window pane.



Click the **Maximize/Restore** button to maximize the pane to fill the window or restore the pane to the previous size as illustrated in the following table.

Icon	Description
	**Maximize.** The pane is maximized to fill the window.
	**Restore.** The pane is restored to the previous size in the window.

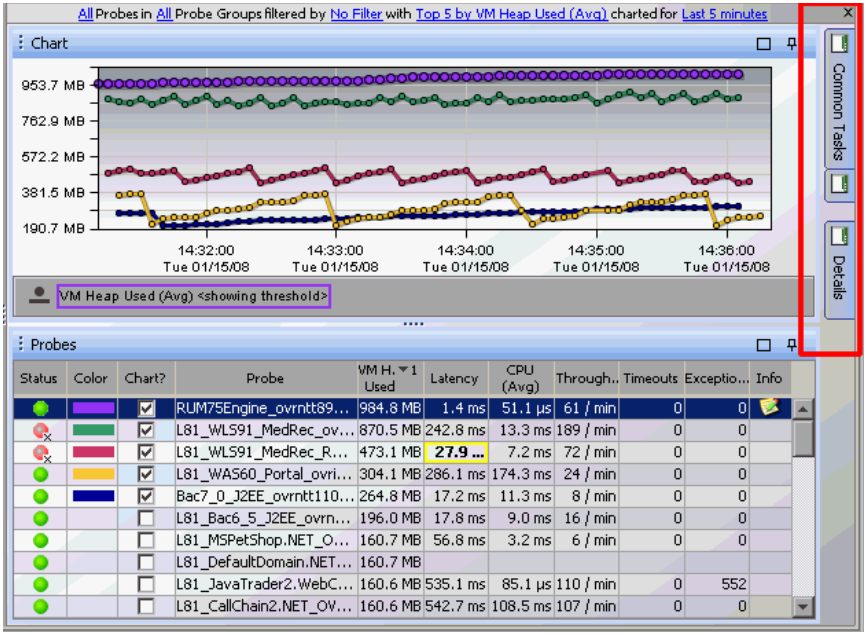
Maximize is a temporary condition - it is not considered a layout change so it is not persisted. To create a custom view with a maximized control - pin all controls that are not of interest.

You can use the Auto Hide feature to minimize windows that you don't want open. The window is minimized along the edges of the screen.

Click the **Auto Hide** button on the title bar of the window to enable or disable Auto Hide, as illustrated in the following table.

Icon	Description
	**Auto Hide Enabled.** Window is pinned (minimized) along the edges of the screen.
	**Auto Hide Disabled.** Window is displayed (open).

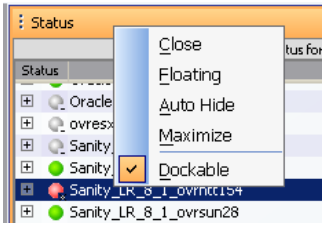
Minimized window panes are shown in the screenshot below:



You can even select auto-hide for the Views side-bar, if you need to have more room to display graphs and tables of data.

Mouse-over the hidden pane to view it and select the Auto Hide button to switch back to displaying the pane in the window.

You can also right-click in the title bar for a selected pane to Maximize/ Restore or enable/disable Auto Hide.



In some views such as the Server Summary view you can set a pane to floating mode by right-clicking in the title bar to select Floating. The window pane undocks and floats above all the other panes. To dock the window pane back into position, double-click the title bar again.

4

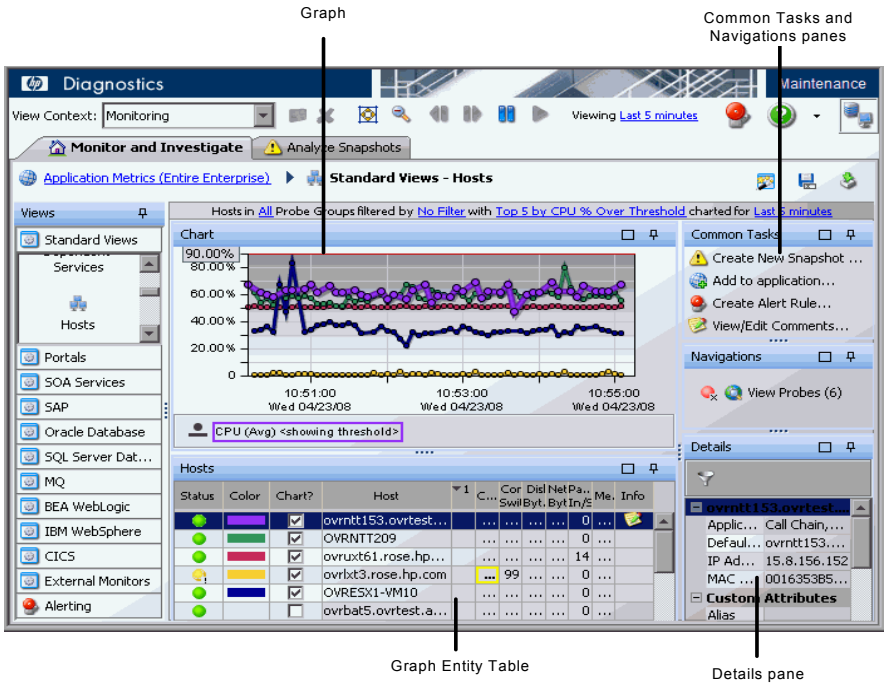
Working with Diagnostics Data Displays

Diagnostics presents performance data for monitored applications in a variety of views. Understanding the basic layout of these views will make it easier to view just the data you need, customize the display to highlight what's important to you and use the common tasks and navigations provided to help lead you through analyzing and diagnosing performance problems.

This chapter includes:

About the Diagnostics Detail Layout

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

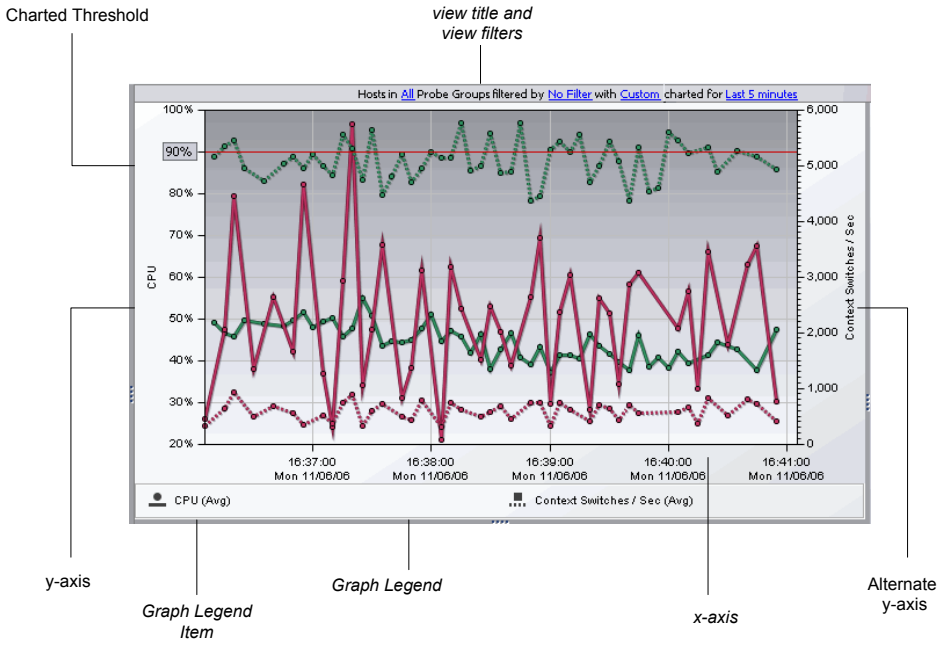


- **Graph.** The graph contains trend lines or stacked areas that represent the performance metrics. For more information about the graph in the detail layout, see “About Graphs” on page 84.
- **Graph Entity Table.** The graph entity table lists the entities associated with a particular view. For example, 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 “About the Graph Entity Table” on page 101.

- **Common Tasks and Navigations.** The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.
- **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 snapshot in Snapshot 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 5, "Working with Metrics, Thresholds and the Details Pane." For more information on investigating alert conditions for thresholds that have been exceeded, see "Investigating Threshold Violations" on page 137.

About 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 and view filters.** 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 View Filters” on page 87.
- **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 1000 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 129.

- **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.

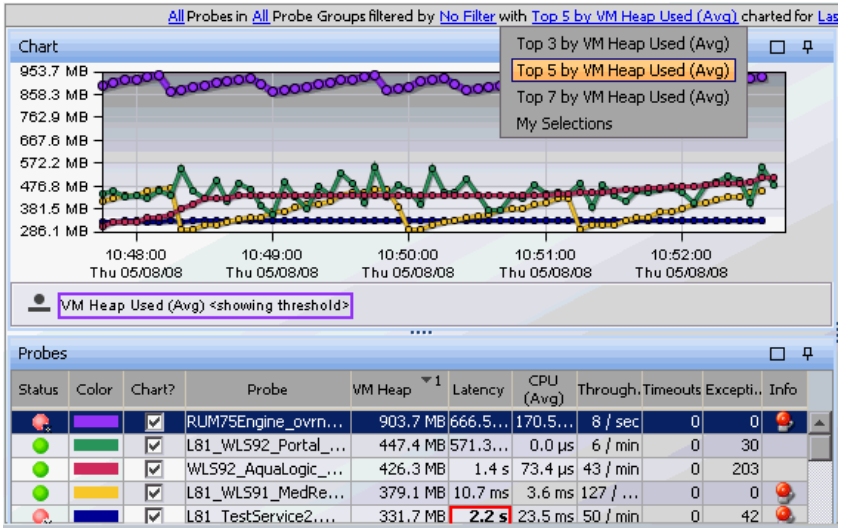
  Note the following:

  - 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.

Working With 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.

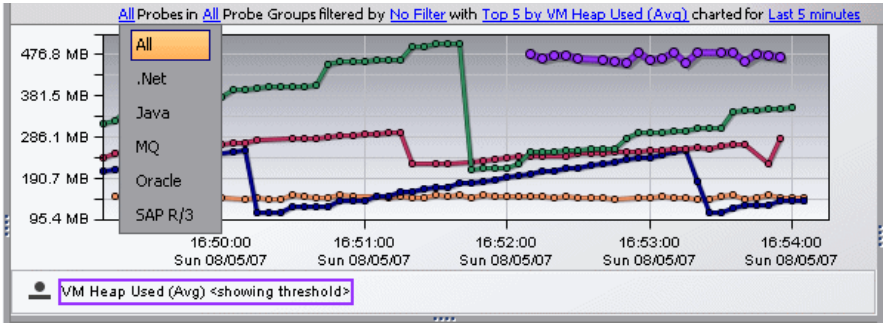
Below is an example of the view title and the view filters from the Probes view. The title includes the Probe Type **All,** Probe Group **All,** Custom Filter **No Filter,** Graph **Top 5 by VM Heap Used (Avg),** and Time Range **Last 5 minutes** menus. The **Graph** filter menu is open displaying the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



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

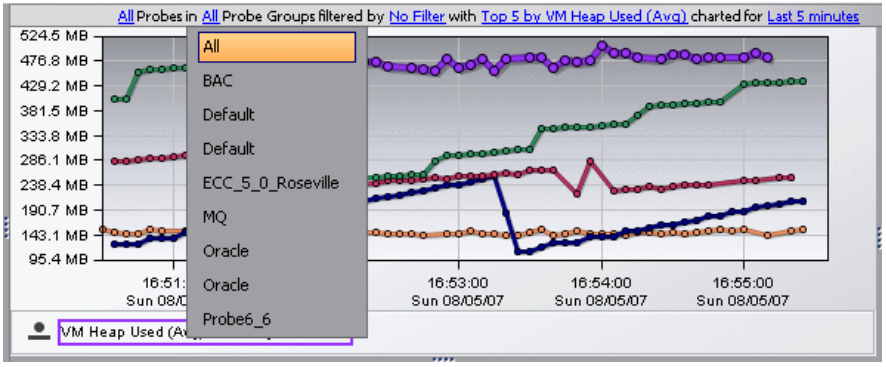
Filtering the View by Type

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:



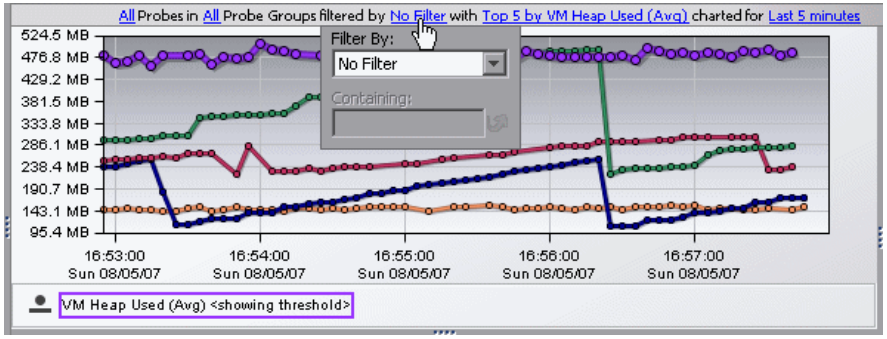
Filtering the View by Probe Group

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.

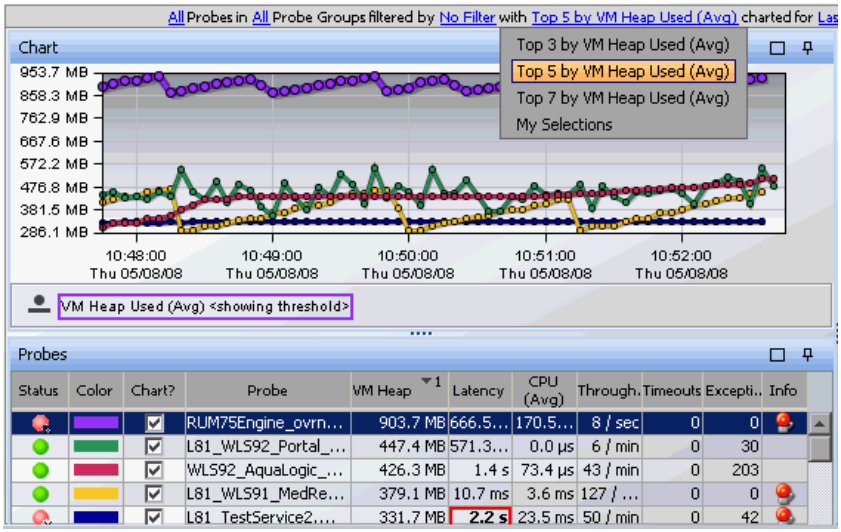
   The following image shows an example of the **Custom** filter after a filtering attribute has been selected.

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 how many entities from the graph entity table should have their metrics charted on the graph (for example Top 3 or Top 5). The choices in the **Graph** filter vary according to the type of entities in the detail layout. The metric in this filter is based on the sorted column (the metric column with a sorting control arrow) in the table.

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 Probes view:



When you select any option (except **My Selections**) 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 **My Selections**, 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 110.

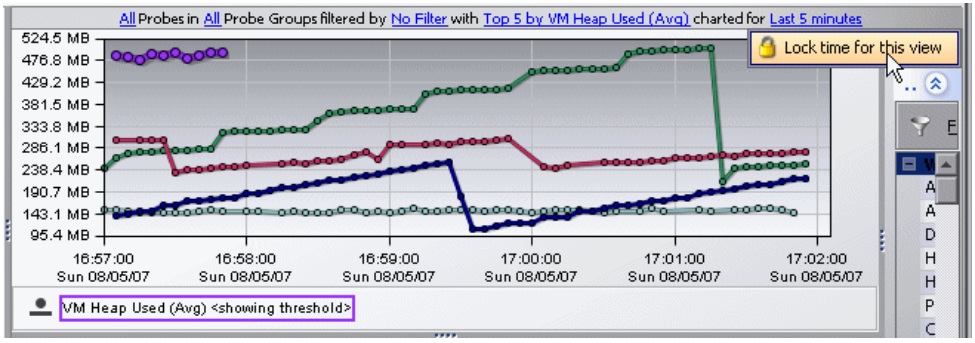
Note: An entity selected in the **Chart** column of the graph entity table may disappear from the table and the graph if no metrics for that entity were captured during the display time range.

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 "Viewing Time Filter" on page 64.

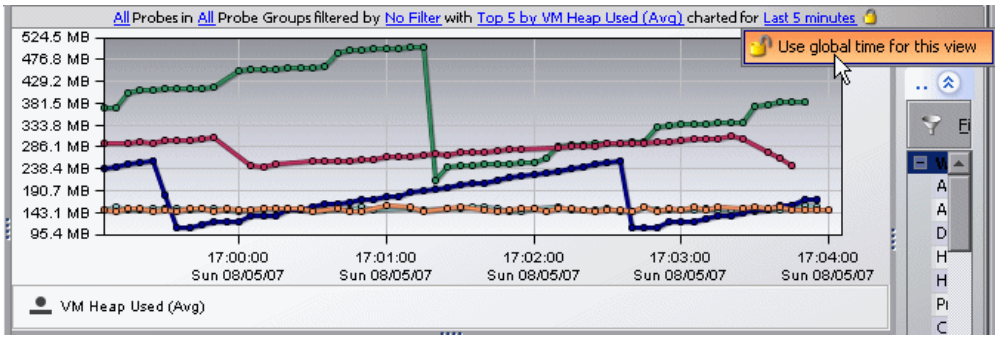
In this view, you use the **charted for** filter to lock a view at a specific time range so it will override any changes to global Diagnostics **Viewing Time** filter.

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.

Displaying Entities and Metrics in the Graph

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

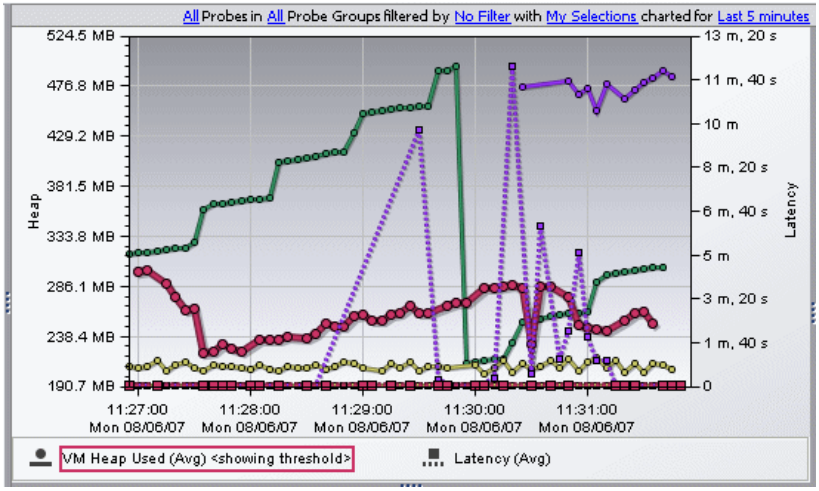
- Using the **Chart** column in the graph entity table. See "Adding and Removing Entities from the Graph" on page 110.
- From the **Graph** filter in the View Filters area. See "Filtering the Graph Metrics" on page 90.
- From the **Details** pane. See "Charting a Metric in the Graph" on page 129.

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.

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 "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.

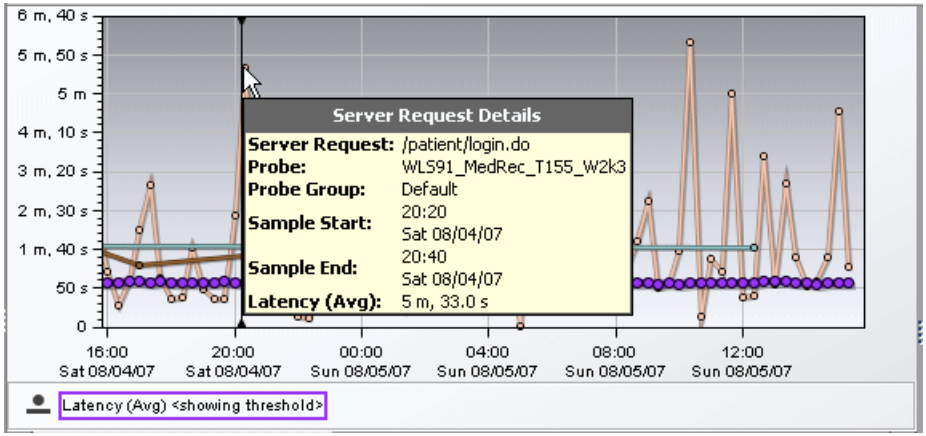
  When you select a trend line in the graph, all of the trend lines for the metrics charted for the entity are also highlighted.

  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, a tooltip appears 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 **/patient/login.do**.
- **Aggregation Period Start Time.** The **Sample Start** entry contains the starting time of the aggregation of the metrics for the data point.
- **Aggregation Period End Time.** The **Sample End** 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)**.

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.

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

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 **Last 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 **Last Hour**, the data is aggregated into one-minute aggregations. These points are initially charted so that all of the data for the last 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.

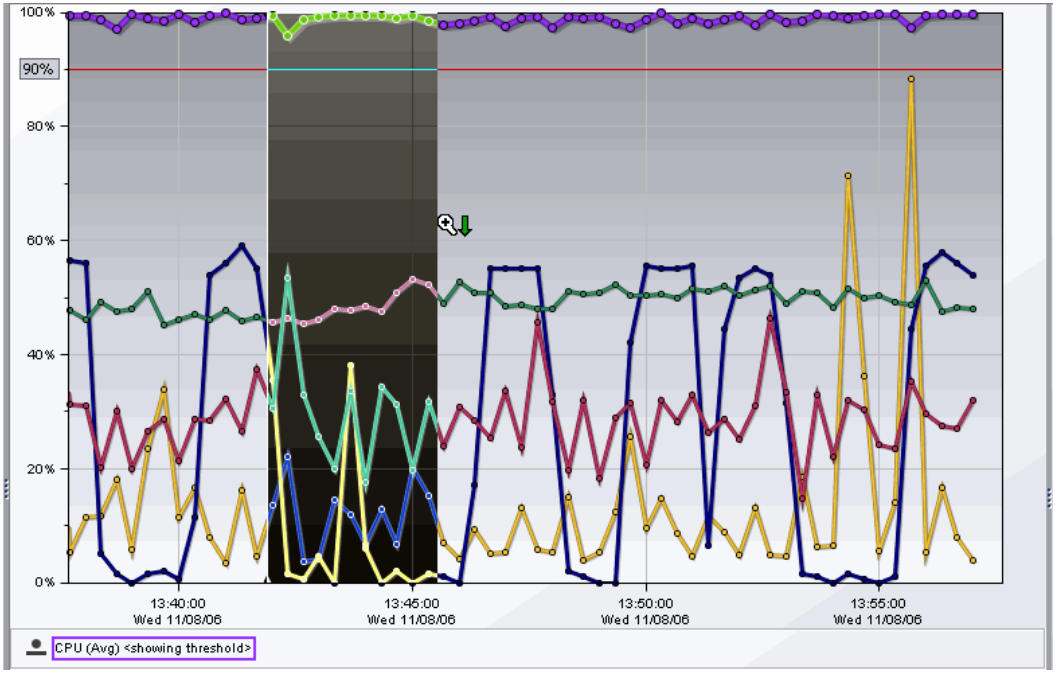
The zoom icon that you see indicates whether Diagnostics is using a magnification zoom or a resolution zoom.

➤ For a description of magnification zooming, see “Understanding Magnification Zooming” on page 96.

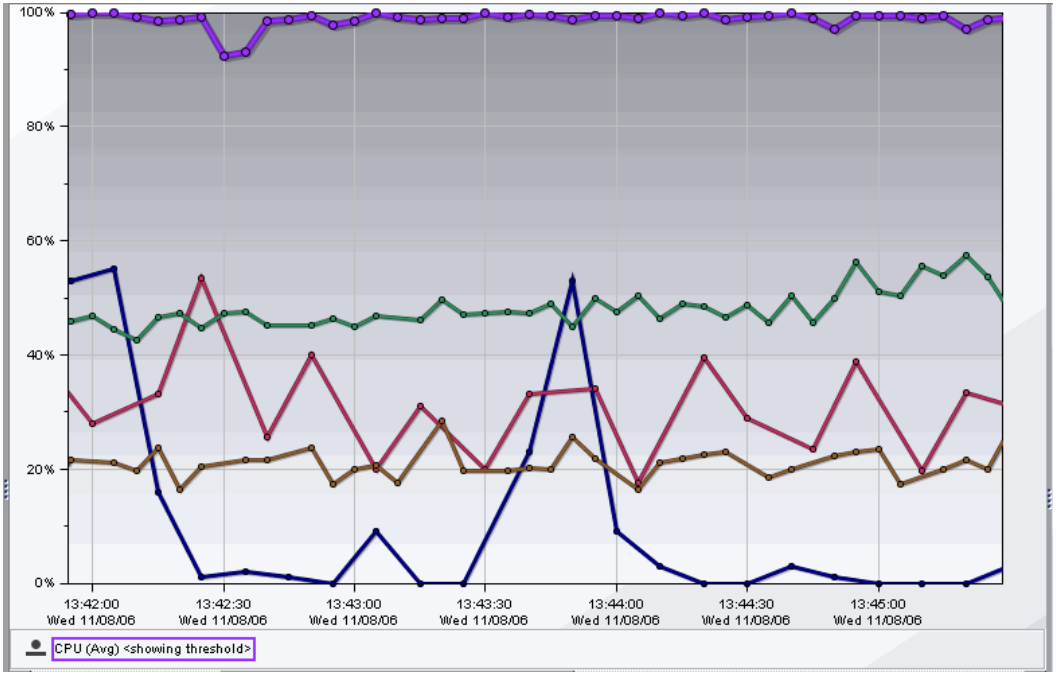
➤ For a description of resolution zooming, see “Understanding Resolution Zooming” on page 97.

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 this additional data 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.

➤ Clicking the **Zoom Out** button causes 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.

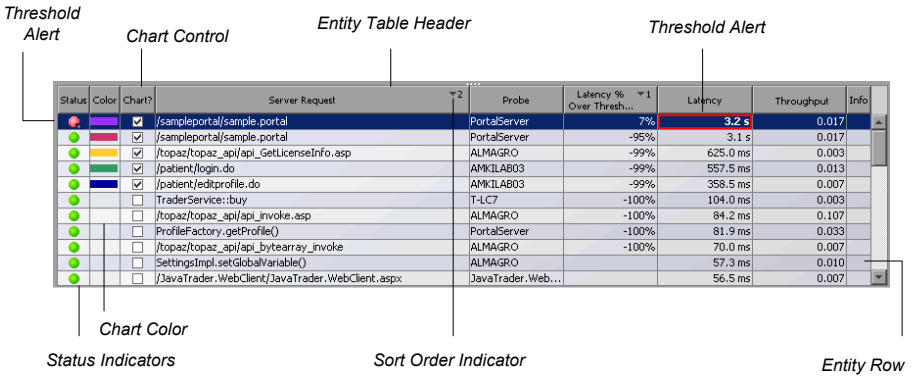
Note that the Zoom Out button will always go the next resolution in the time range filter when one of the fixed time ranges is selected, but if you are in a custom time range (as often happens when you zoom in), the zoom out function will just keep increasing the time range of the custom option. It will never go back to one of the fixed time ranges.

➤ As an alternative you can use the **Fit data to size** button which will undo any magnification zooms at the current resolution.

➤ Or you can use the time range filter to go back to whatever time range you were originally looking at before you zoomed in on the data.

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:



Status	Color	Chart?	Server Request	Probe	Latency % Over Thresh...	Latency	Throughput	Info
		☑	/sampleportal/sample.portal	PortalServer	7%	3.2 s	0.017	
		☑	/sampleportal/sample.portal	PortalServer	-95%	3.1 s	0.017	
		☑	/topaz/topaz_api/api_GetLicenseInfo.asp	ALMAGRO	-99%	625.0 ms	0.003	
		☑	/patient/login.do	AMKILAB03	-99%	557.5 ms	0.013	
		☑	/patient/editprofile.do	AMKILAB03	-99%	358.5 ms	0.007	
		☐	TraderService::buy	T-LC7	-100%	104.0 ms	0.003	
		☐	/topaz/topaz_api/api_invoke.asp	ALMAGRO	-100%	84.2 ms	0.107	
		☐	ProfileFactory.getProfile()	PortalServer	-100%	81.9 ms	0.033	
		☐	/topaz/topaz_api/api_bytearray_invoke	ALMAGRO	-100%	70.0 ms	0.007	
		☐	SettingsImpl.setGlobalVariable()	ALMAGRO		57.3 ms	0.010	
		☐	/JavaTrader.WebClient/JavaTrader.WebClient.aspx	JavaTrader.Web...		56.5 ms	0.007	

- **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.

  For more information about customizing the columns, see “Customizing the Graph Entity Table” on page 109.

  For more information about sorting the rows in the graph entity table, see “Sorting Rows in the Table” on page 107.

- **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 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.

  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 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.
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 recent 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.

The following describes, in general, how a status of RED or YELLOW is determined:

- Diagnostics compares the average value for the metric in the last five minutes to the threshold. If the threshold is crossed, then the status is RED.
- If not, Diagnostics counts the total number of points (five second averages from the probe) that violate the threshold. If the count is greater than three but the average is still under the threshold (current status is not RED), then the status is YELLOW; otherwise the status stays at GREEN.
- If none of the above, the status is GREEN.

Status indicators can progress in severity from yellow, and then to red as the metric's performance deteriorates. However, the indicators do not regress in the same way. Once a red 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 131.

- **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 110.

- **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 131.

- **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. (You can disable an alert rule in the Alert Rule dialog box by de-selecting both checkboxes for Alert Triggers.)

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

  For more information about Alert Notification Rules, see Chapter 7, "Working with Alerts." For more information on entering comments for an entity, see "Managing Comments for an Entity" on page 119.

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 Table Columns

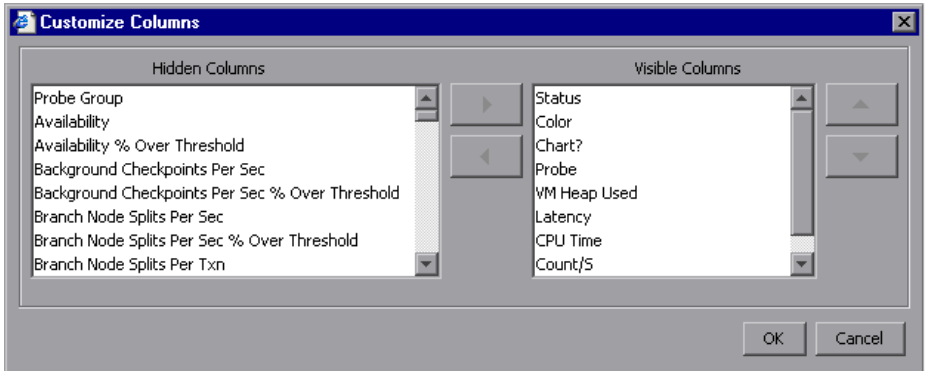
The default columns that appear in the Status view and in the graph entity table of the detail layout views help you identify the entities that are included in the view, understand the status of each entity's performance, and see some of the performance metrics that have been gathered for each entity.

Note: Graph entity tables (for probes, hosts, server-requests, etc.) only show the first 100 results. If you change the column you are sorting by, you can get a different set of 100 entities. This obviously applies only where there are more than 100 things in the dataset, which will not occur very often with probes or hosts (but can in large deployments), but will almost always occur with server requests.

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 customizations.

Note: If you customize one of the standard detail layout views, and would like to save the customizations 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 “Creating a New Custom View” on page 219.

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:

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

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

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

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

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

To create a new custom sort order for the table:

1 Click the header of the column that 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.

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 104.

Note: Graph entity tables (for probes, hosts, server-requests, and so on) only show the first 100 results. If you change the column you are sorting by, you can get a different set of 100 entities. This only applies where there are more than 100 items in the dataset - which will not occur very often with probes or hosts (but can in large deployments), but will almost always occur with server requests.

Working with Charted Metrics in the Graph Entity Table

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 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.

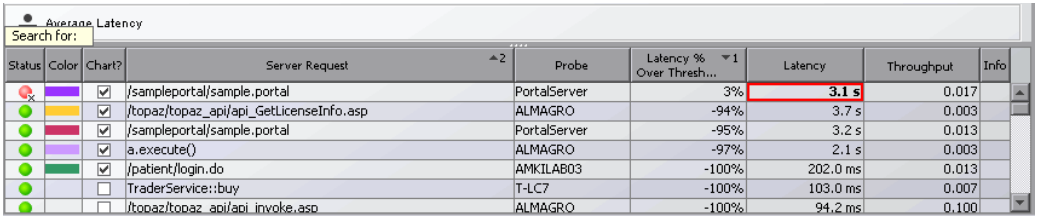
When you click the **Chart** check box for any entities in the graph entity table, Diagnostics changes the **Graph** filter in the **View Title** to **My Selections** to indicate that you have manually selected the entities. For more information about the **Graph** filter, see "Filtering the View by Custom Filters" on page 89.

Searching for Entities in Tables

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:



2 Type all or the first part of the name of the entity that you would like to locate. As you type each character, Diagnostics begins the search. To locate an entity starting with a / character you must include the / in the search field.

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**.

Selecting Multiple Rows in a Table

If you want to apply a common task to multiple items you can select multiple rows in a table and then select one of the following types of tasks:

- Add to Application
- Remove from Application
- Delete Alert Rule (delete from the Alert Rule dialog box or delete entity status rules from the graph entity table)

- Create or Replace Alert Rule (entity status rules from the graph entity table)
- Delete Comments
- Create or Replace Comments
- Create New Snapshot and Add or Add to Active Snapshot

To select multiple rows in the graph entity table:

1. Select a row in the graph entity table.
2. Scroll down to select all the rows in between and select SHIFT + left mouse click.

   Or hold the SHIFT key down and use the arrow keys to select multiple contiguous rows.

   Or press CTRL + left mouse click to select multiple individual rows in the table.

   Or press CTRL + A to select all rows.
3. Right-click to select a task from the menu of options you can apply to the multiple selected rows.

   This is supported in all tables except Alert events, status views and the probe groups of the Server Summary screen.

   Only those tasks that can be applied to all selected entities will be available in the menu.

   For example in the Probes view if you select 5 probes with active alert rules set, you will see menu items for Replace Alert Rule and Delete Alert Rule. If you select probes with no alert rules set you will see the Create Alert Rule menu option. If you select a combination of probes with alert rules set and probes with no alert rule set or disabled alert rules, you will not see any alert rule options in the menu.

   When you multiselect many entities then select **Remove from Application**, the message displayed (in hover text) is for a single entity. However what will actually be removed is - all selected entities from the application. To avoid confusion you would most likely use the multi-line remove when you want to select everything and remove from a specific application.

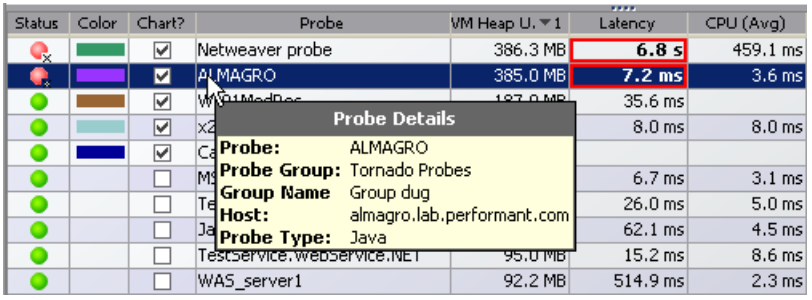
Viewing Additional Entity Information

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 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:



- 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 5, “Working with Metrics, Thresholds and the Details Pane.”

Drilling Down On Entities in the Graph Entity Table

Most of the Diagnostics views provide a way for you to drill to other Diagnostics views to see more detailed performance metrics for an entity selected in the Details pane. You can select a drill to link in the Navigations pane or right-click on the entity in the table and select a drill to option from the pop-up menu. You can also double-click an entity in the table and drill to a relevant view if a navigation is available.

For some entities you can open the Diagnostics Profiler to get more detailed information and analyze memory problems.

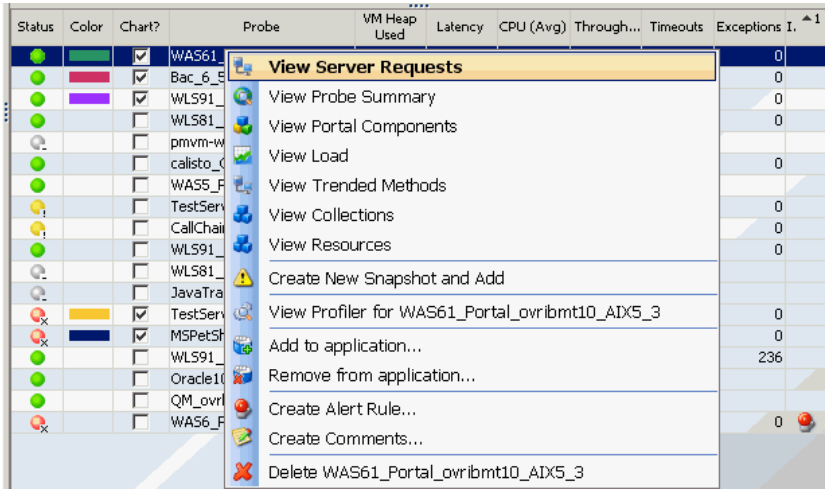
For more information on relevant navigations for an entity see "About the Common Tasks and Navigations Panes" on page 115.

About the Common Tasks and Navigations Panes

Diagnostics provides access to the common tasks and navigations that are available for an entity in the table in two ways:

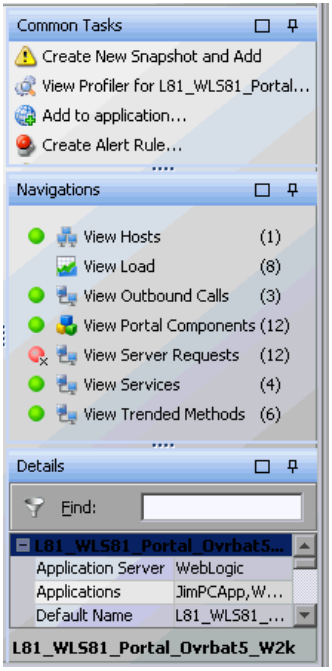
- Right-clicking on the row in the table to display the entity pop-up menu.
- Using the Common Tasks and Navigation panes above the Details pane on the right side of the Diagnostics views.

The following image shows an example of the entity pop-up menu that is displayed for a probe entity in the Probes view.



The Common Tasks pane contains tasks you may want to perform based on the currently selected entity. The Navigations pane contains links to navigations you may want to use based on the currently selected entity.

The navigation links listed also show the status of the related entities and their count, where this information is available.



When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Common Tasks

Some key Common Tasks are described in the sections that follow:

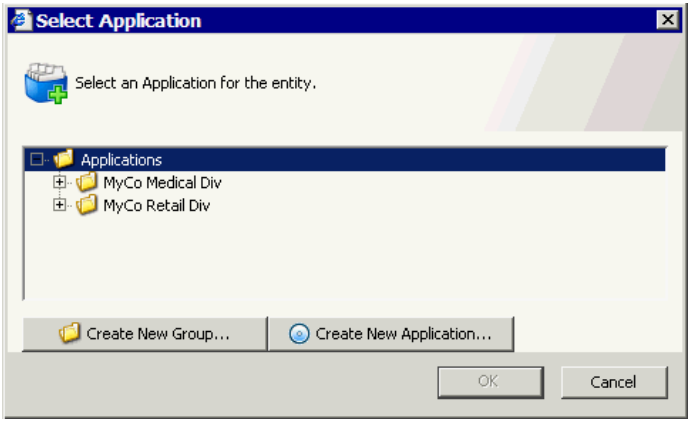
- Creating a New Snapshot
- Adding and Removing an Entity from an Application
- Managing Alert Rules for an Entity
- Managing Comments for an Entity
- Deleting an Entity

Creating a New Snapshot and Adding the Entity

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

Adding and Removing an Entity from an Application

Selecting **Add to Application** from the entity pop-up menu or from the Common Tasks pane displays the Select Application dialog box. You select an application from this list and the entity will be added to that application.



An entity can be added to any number of applications. See Chapter 2, “Working with Applications” for how to work with applications.

You can select multiple entities in the graph entity table to add or remove from an application. See “Selecting Multiple Rows in a Table” on page 111 for how to select multiple rows. When selecting multiple entities you should select groups of entities that you want to add to or remove from the same application.

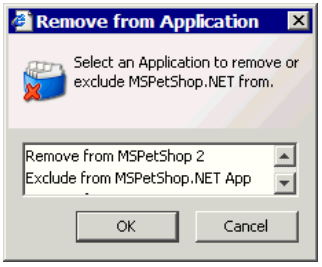
If the user you are logged in as has the appropriate permissions, you can also **Create a New Application Group** and **Create a New Application** from this dialog box. When you create a new group or application, that group or application will automatically be selected.

Note: If you create a new application (or multiple applications) from this dialog box, but then cancel out of the dialog box, the created applications will still exist.

Selecting **Remove from Application** from the entity right-click menu or from the common Tasks menu displays the Remove from Application dialog box. You can select an application to remove or exclude the entity from. The user you are logged in as, must have the appropriate permissions for you to be able to remove or exclude entities from an application.

The easiest way to remove entities you do not want is from the Topology view. Select the entity in the Topology view and right-click to select **Remove from Application**.

You cannot remove or exclude an entity if it is a parent of an entity that was manually added. For example, if you add the server request **/login.do**, you will not be able to remove or exclude the probe where this server request is running since this is the parent of the server request entity. This limitation exists to ensure that you will always be able to navigate to anything you have added to your application.



Note: In the Remove from Application dialog box you may see Remove from or Exclude from actions. This distinction is based on internal relationships between entities but both have a similar function so you can ignore the distinction.

You can remove entities from applications in either the Entire Enterprise or a selected application context. See Chapter 2, “Working with Applications” for how to work with applications.

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 options from the pop-up menu. The same menu options are also available on the Common Tasks pane. The options displayed in the menu vary depending on whether alert rules exist or not.

You can create an alert rule based on a metric collected for an entity by selecting the Alert Rule indicator icon next to the metric in the Details pane.

You can select multiple entities in the graph entity table and then select the menu item to create, replace or delete alert rules. See “Selecting Multiple Rows in a Table” on page 111 for how to select multiple rows. Typically, when selecting multiple entities you should select groups of entities that you want to have the same alert rules.

For more information on alert rules see Chapter 7, “Working with Alerts.”

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 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 pane.

You can select multiple entities in the graph entity table and then select the menu item to create, replace or delete comments. See “Selecting Multiple Rows in a Table” on page 111 for how to select multiple rows. Typically, when selecting multiple entities you should select groups of entities that you want to have the same comment.

Deleting an Entity

There are times when you may want to remove a probe or get rid of old metrics sooner than the six month automatic purge (see "Purging Data for an Entity" on page 120). The **Delete <entity>** in the entity pop-up menu in the graph entity table provides a way for you to delete a probe or old metrics from Diagnostics. The same option is available on the Common Tasks pane.

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.

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.

Purging Data for an Entity

Diagnostics automatically purges an entity from the Diagnostics data after six months have passed since data was last received from the probe. Once an entity (such as a probe) is purged, access to its trends and summaries is no longer possible.

This is also the case when a probe is renamed. Data for the probe under the old name is only available for 6 months after the rename. Even if you don't change the probe but the monitored application is updated and server requests change then the old server requests data will not be available after 6 months.

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.

The purging interval can be changed in **server.properties**, see the *HP Diagnostics Installation and Configuration Guide*.

Common Navigations

Some commonly used Navigations are described in the sections that follow:

Navigating to Other Diagnostics Views

Many of the Diagnostics views allow you to navigate from the entities listed in the graph entity table to other views so that you can analyze their performance in greater detail and diagnose performance problems. Diagnostics updates the Navigations panes so that it lists only **relevant** navigations that are appropriate for the selected entity.

As you navigate from one view to the next, the breadcrumb at the top of the view keeps track of the path that you have taken so that you can navigate backwards. If you navigate to an entity screen that is already in the breadcrumb, then the navigation will move up the breadcrumb to the appropriate level.

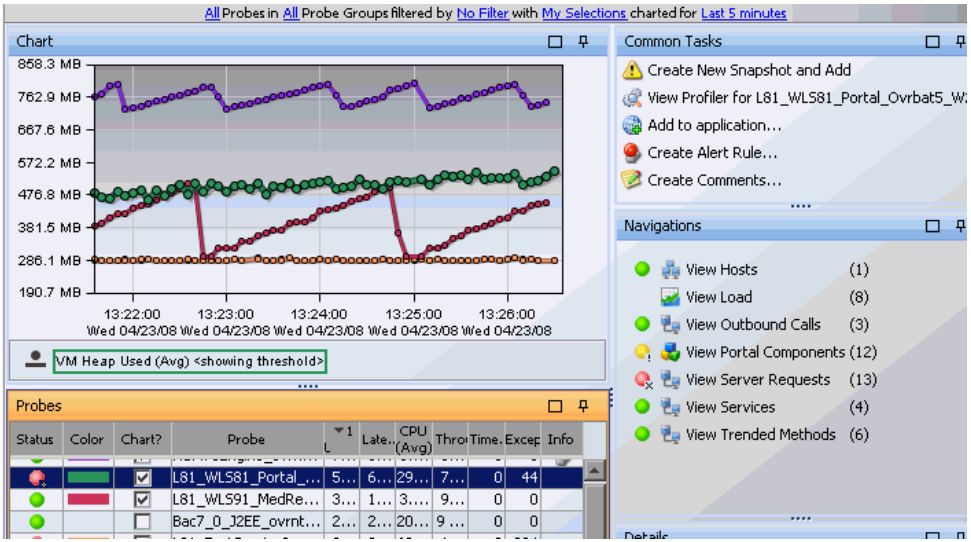
Note: Not all views have drill down navigation options.

There are three ways to navigate from an entity in the graph entity table to other views:

- Right-click on a row in the table. Diagnostics displays the pop-up menu for the entity listing the available 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 using a menu option. The default drill down option is shown in bold in the entity pop-up menu.
- Select the navigation option from the Navigations pane.

  The navigation links provide you with suggestions for how to further analyze a performance problem and diagnose the cause.

For example if you see that the status of a probe is red and in the Navigations pane you see that there are 13 server requests and the status is also red you may want to select the View Server Requests link in the Navigations pane to get more details on the server request's performance.



Selecting the Profiler for a Probe Entity

Selecting **View Profiler for...** from the entity pop-up menu or from the Navigations pane causes Diagnostics to open the Diagnostics profiler for the probe. If the entity is a Java Probe, Diagnostics opens the Java Diagnostics Profiler in a new window in the same browser (see Chapter 38, "Using the Java Diagnostics Profiler" for details). If the entity is a .NET Probe, Diagnostics opens the .NET Diagnostics Profiler in a new browser (see Chapter 41, "Using the .NET Diagnostics Profiler" for details).

5

Working with Metrics, Thresholds and the Details Pane

The Details pane in Diagnostics views displays the performance metrics collected by the probe for the selected entity. The Details pane is also where you select which metric to graph, set metric thresholds to be used in determining the status of an entity and you can also define rules for alert triggers and notifications.

This chapter includes:

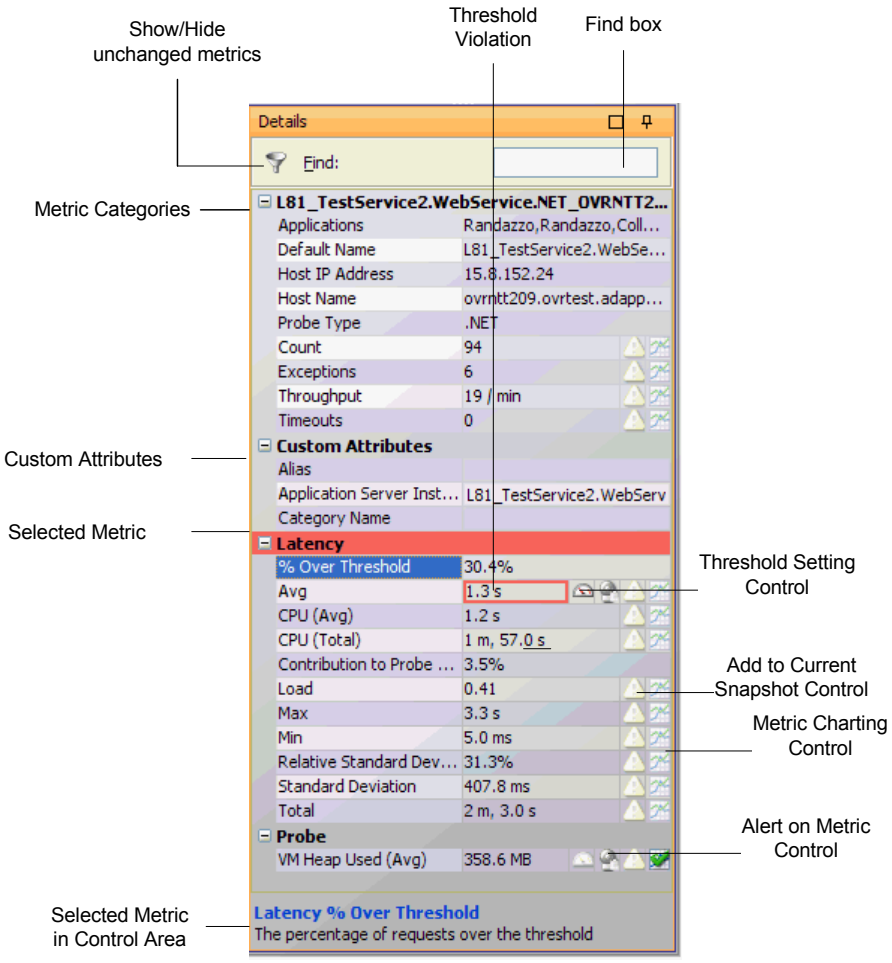
About the Details Pane

Using the Details pane, you can do the following:

- Control the metrics that Diagnostics charts in the graph.
- Specify the metrics included in a snapshot.
- Set thresholds for the metrics.

The Details pane has a table with a list of metrics and their values, grouped by metrics categories. Note that one of the metric categories represents the applications the entity you selected in the graph entity table belongs to. The metrics displayed in the table can be Java metrics, system metrics, or JMX metrics.

Note: For information on configuring the system metrics or JMX metrics that appear in the Details pane, see the *HP Diagnostics Installation and Configuration Guide*.



Selecting a Row from the Graph Entity Table

The Details pane displays the metrics for the entity you have selected.

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.

By default, the Details pane filters out the metrics that have not changed during the time period set in the **Viewing Time** filter of the graph.

➤ To display all the metrics, including those that have not changed, click the **Show all metrics** button, located at the top left corner of the Details pane.

➤ To filter out the metrics that have not changed, click the **Hide metrics which have not changed** button, located at the top left corner of the Details pane.

Searching for Metrics in the Details Pane

You can search for a metrics in the Details pane, using the **Find** box

To find a metric in the details pane:

In the **Find** box (situated at the top of the Details pane), type all or the first part of the name of the metric that you would like to locate.

➤ If a match is found, Diagnostics selects the first row in the Details pane that matches the search criteria.

➤ If no match is found, the color of the Find box changes to red.

Reviewing the Details and Settings for a Metric

When you select a metric in the Details pane, the metric's definition appears in the metric control area at the bottom of the Details pane. 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 definition for this metric.

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)**.

If the selected metric can have an associated threshold, a **Threshold** icon appears next to the metric in the table. See "Setting Metric Thresholds" on page 131 for details on how to set or edit metric thresholds.

Updating Custom Attributes

In the list of metrics there is a category called **Custom Attributes** that lists attributes associated with the entity you selected in the graph entity table.

The attributes that are listed under Custom Attributes 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 SOA Services - Operations 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 dialog boxes 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 *HP 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 Click in the column to the right of 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 a Snapshot

The rows that contain metrics that can be included in a Diagnostics snapshot are marked with the Snapshot Analysis icon. The Snapshot Analysis icon behaves differently depending on whether you are in Snapshot Analysis mode (the Analyze Snapshots tab is selected) or Monitoring mode (the Monitor and Investigate tab is selected).

Including a Metric in a New Snapshot

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

Including a Metric in an Active Snapshot

When you are looking at a Diagnostics view while in Snapshot Analysis mode, clicking the Snapshot Analysis icon adds and removes the metric selected from the Details pane from the snapshot.

When the Snapshot Analysis icon includes a check mark, clicking the icon causes Diagnostics to remove the selected metric from the active snapshot.

When the Snapshot Analysis icon does not includes a check mark, clicking the icon causes Diagnostics to add the selected metric to the active snapshot. This metric will then be charted in the active snapshot.

Alerting on a Metric

When the **Alert Rule** icon appears in a row in of the Details pane, you can set an alert rule for the metric.

There are two types of alert notification rules:

- Those based on entity status change occur when any metric for the entity violates a threshold.
- Those based on metric status change occur only when the specified metric violates a threshold.

  Alert rules based on metric status change are defined from within the Details pane. Only those metrics that can be used in an alert rule will show the Alert Rule indicator icon.

When the **Alert Rule** icon is greyed out, no alert rule has been set but the metric can be used in an alert rule. You can create an alert rule for the metric by clicking the Alert Rule icon in the row or by right-clicking on the row and selecting **Create Alert Rule for Metric**. Use the Alert Rule dialog box that is displayed to define an alert rule for the metric. See Chapter 7, "Working with Alerts" for details.

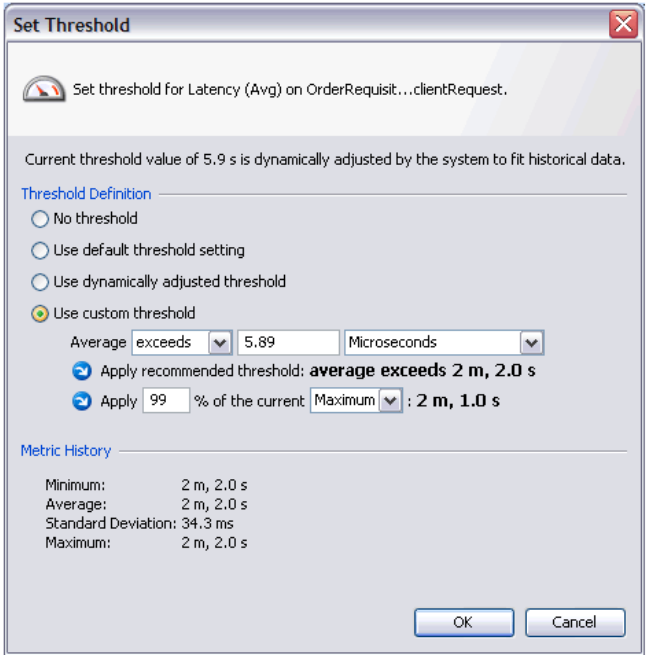
When the Alert Rule icon is highlighted (red), the metric has an alert rule set. To edit or delete an alert rule for the metric, right-click the Alert Rule indicator icon and select **Edit Alert Rule for Metric** or **Delete Alert Rule for Metric**.

If an alert rule has been set for a metric but the alert notification is disabled the Disabled Alert Rule icon is displayed.

Setting Metric Thresholds

When the **Metric Threshold** icon appears next to a metric in the Details pane, the selected metric can have an associated threshold. If the icon is dimmed, the threshold has not yet been set for the metric. If the icon is active, a threshold value has been set for the metric.

To set or edit a threshold, click the Metric Threshold icon or right-click the metric in the details pane and select **Set Threshold for Metric**. The Set Threshold window is displayed. The example shown below is for setting a latency threshold on a server request.



Threshold Definition Options

There are a number of possible threshold definition options, though not all options are appropriate for all metrics. The options are:

- **No threshold.** This radio button will be selected if no threshold has been set for a metric. If no threshold is set, there will be no alerts and no status displayed for the entity. However, if there is no threshold for a 'parent' entity a status may still be displayed based on the status of 'child' entities (for example, probes and server requests or Web services and operations).

  Once a threshold is set, you can select the **No Threshold** option to remove the threshold. After you've removed a threshold you can always come back later and set a threshold.

- **Use default threshold setting.** If selected, this radio button sets the threshold to the default value in the **<server>/etc/thresholds.configuration** file. Selecting this option clears any user-defined threshold settings. If there is no default threshold for this metric in the property file, then no threshold value is set, by default, on this metric. When there is no default, you should select another option such as Use Custom Threshold to set a threshold for the metric. After selecting the Use Default Threshold Setting option you can see what default value was used by exiting the Set Thresholds dialog box and re-entering the screen. The message at the top of the dialog will indicate, for example: Current threshold has a system-provided default value of 0.10.

  The **thresholds.configuration** file contains default threshold values for different entity types. The file also specifies DYNAMIC as the default setting for latency metrics. Changes to the default settings are made in this property file. See an example below:

```
################################################################################################
#### Default thresholds for metrics.
####
#### Syntax is
####    <metric-parent-node-classname>.<metric-name> = [-] value
#### or
####    <collector-name>.<metric-name> = [-] value
####
#### Value is interpreted as a double. Value of '0' is not allowed.
#### Negative value indicates it is a lower-bound threshold.
################################################################################################
# Threshold on avg. latency of fragments (in microseconds)
com.mercury.diagnostics.common.data.graph.node.FragmentData.latency = 60000000:DYNAMIC

# Threshold on avg. latency of web service operations (in microseconds)
com.mercury.diagnostics.common.data.graph.node.WebServiceData.latency = 60000000:DYNAMIC

# Threshold on avg. latency of R3 Server Requests (in microseconds)
com.mercury.diagnostics.common.data.graph.node.R3ServerRequestData.latency = 60000000:DYNAMIC

# Threshold on avg. latency of methods (in microseconds)
com.mercury.diagnostics.common.data.graph.node.MethodData.latency = 60000000

# Threshold on avg. latency of Portal component operations (in microseconds)
com.mercury.diagnostics.common.data.graph.node.PortletData.latency = 60000000
```

  The defaults cannot be changed in the Set Thresholds dialog box.

- **Use dynamically adjusted threshold.** This radio button is enabled only when you have selected a latency metric; for other metrics this option will be greyed out.

  A dynamic threshold is automatically adjusted by the system based on historical data. The last seven days of data is used by default. If there aren't seven days of data yet, the default value for threshold is used initially and then adjusted afterwards based on historical data.

When a dynamic threshold is set, you will see the following description in the Set Threshold dialog box: Current threshold value of <value> is periodically adjusted by the system to fit historical data.

The dynamic threshold adjustment job runs once a day. It can be run manually on the probe from the Probe Administration Welcome page, select **Advanced Options** > **Scheduler** from the Scheduler page.

➤ **Use custom threshold.** Select this radio button to set a fixed, custom threshold value. Since the value is fixed, as application usage patterns change, it may be necessary to select this button again to set a different threshold for the changed usage patterns. You enter the custom threshold on the line below:

Average <exceeds/falls below> <value> <units>

You can make your selections and enter a value directly on this line or you can use the two actions described below to help you determine a custom threshold value. Note that you cannot have a value of zero.

➤ **Apply recommended threshold: average exceeds <value>.** This action is available only for latency metrics. It uses heuristics to recommend a threshold for the metric given the recent metric history statistics.

When you select the action button, the value recommended will be entered on the line above as the custom threshold value. The metric setting is static, so it may be necessary to reset it from time to time as your application or workload evolves.

➤ **Apply <percent> % of the current <minimum/maximum/average>: <value>.** This action like the apply recommended threshold action, helps you determine a custom threshold value. It can be useful for metrics where selecting a threshold by percent makes sense, e.g., latency metrics, size metrics.

When you select the action button, the value recommended will be entered on the line above as the custom threshold value. The metric setting is static, so it may be necessary to reset it from time to time as your application or workload evolves.

Metric History

The metric history panel shows up to six hours of statistical history about the metric. The four statistical history values that may be shown are:

- Minimum value
- Maximum value
- Average value
- Standard deviation

This information is provided to give you some guidance in selecting a good threshold for the metric, that is, a threshold that will trigger when there is an emerging problem and not when there is simply a temporary spike in usage. Although these values can be helpful, you may want to look over the metric history for a longer period of time e.g., several months, to examine metric trends and behavior during incidents.

Not all metrics will have the same statistical metric values because Diagnostics does not keep certain statistics on all metrics. For example standard deviation is tracked for latency metrics, but is not tracked for host CPU utilization. If a historical statistic is not shown, it is because it is not tracked for that metric.

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

Query Page to View All Thresholds Set

The server has a query page that lists all custom and dynamic thresholds from all the mediator systems. You can access the query pages from the URL **http://<Diagnostics Server>:2006/query** or from the Diagnostics Server Administration page select **Configure Diagnostics** > **query** and then select **View Thresholds**. Note that the page may take a while to load. This query is used just for information by support in debugging alerts.

Audit Log of Threshold Changes

When a user changes a threshold, an entry is logged in the **server.log** file on the corresponding server. For example:

```
2008-03-03 90:27:18,071: INFO configure : User admin set latency metric
threshold to 15 seconds for entity L81_WLS91_MedRec_ovrnett155_W2K3
```

Using Thresholds in Monitoring Business Applications

Metric thresholds should be set to monitor the performance of important business applications. It is not necessary or desirable to set thresholds on every possible metric; in fact setting too many thresholds may be counter-productive, overloading the IT staff with alarms when there is no real problem, or distracting them with trivial issues when there is a more important problem to solve.

When set correctly, thresholds can be useful for:

- Verifying whether service level objectives are being met or not.
- Providing advanced warning when the application workload is changing due to an increasing demand that may result in degraded performance if application provisioning changes are not made.
- Notifying the IT staff when application service level objectives are in danger of not being met, or are not being met due to infrastructure problems, application problems, or workload changes.

To achieve these objectives, the business requirements for the application and its recent workload history should be considered. The business requirements should be considered first. Some questions to consider are:

- Is this an important application to the business customer?
- What is the expected average transactions/second? What is the expected minimum and maximum transaction rate?
- What is the desired average transaction response time? What is the maximum acceptable transaction response time?
- What is the business impact of failure to meet transaction rates and response times?
- How willing or able is IT able to deal with false alarms and at what cost? An overly aggressive threshold can result in more false alarms; an overly lax threshold can result in delaying recognition of emerging problems. There is an important cost/benefit trade-off to consider.

Once the business requirements are understood, the current application characteristics should be considered:

- How is the application structured?
- What are the key transactions? What are the key components supporting these transactions e.g., service requests, SQL statements, hosts, etc.
- How should the business requirements be mapped to lower level component threshold requirements? What thresholds should be set on which components? Not every component requires a threshold for good application performance management.
- What is the workload history over the past months? Can the business requirements be met with the current deployment or are changes required?
- How frequently should thresholds be reviewed and updated?

This stage will likely require a team effort from applications and operations experts.

Investigating Threshold Violations

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 highlight in the Details pane indicating a threshold violation determines its severity:

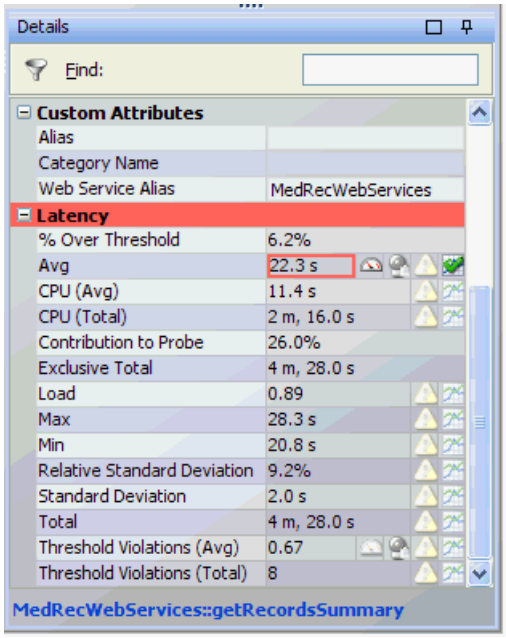
- **No outline around the metric or highlighting in the table.** Normal – There has been no threshold violation or there is no threshold defined for the metric.
- **Yellow.** Warning – The component is occasionally but not consistently exceeding the defined threshold.
- **Red.** Critical – The component is consistently exceeding defined thresholds.

The following describes, in general, how a status of RED or YELLOW is determined:

- Diagnostics compares the average value for the metric in the last five minutes to the threshold. If the threshold is crossed, then the status is RED.
- If not, Diagnostics counts the total number of points (five second averages from the probe) that violate the threshold. If the count is greater than three but the average is still under the threshold (current status is not RED), then the status is YELLOW; otherwise the status stays at GREEN.
- If none of the above, the status is GREEN.

Status indicators can progress in severity from yellow, and then to red as the metric's performance deteriorates. However, the indicators do not regress in the same way. Once a red 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 (Latency) has a red background because the metric value, **22.3s**, exceeded the threshold.



If you have set a CUSTOM latency metric threshold and there have been threshold violations, you will see the **Threshold Violations (Avg)** and **Threshold Violations (Total)** metrics in the details pane. This shows the average and total number of latency threshold violations across ALL instances. (You may need to select the Show All Metrics button in the Details pane to see these two metrics if the values haven't changed in the last five minutes.)

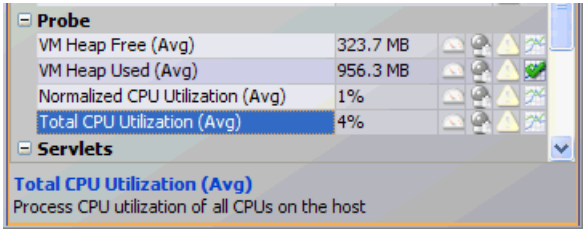
You can set a threshold on the Threshold Violations (Avg) metric, as well as chart the metric and set an alert rule for the metric.

Note: You have to set a CUSTOM latency threshold, then after 5 minutes when the probe picks up this value, the Threshold Violation metrics can be calculated. If you want to set a threshold for the Threshold Violations (Avg) metric it must be set as a custom threshold because there is no default value.

For example: if for a 5 minute period there are 12 samples and each sample has 2 instances and twelve of the instances have a violation then we have total=12, and avg=12/24 or .50. You could set a custom threshold that triggers an alert if the Threshold Violations (Avg) metric exceeds .50.

Probe CPU Utilization Metrics

Diagnostics collects probe CPU utilization metrics per Java process and displays them under the Probe metrics section of the Details pane.



For a probe on a supported system you'll see the following metrics:

- **Normalized CPU Utilization (Avg).** The value for process CPU utilization normalized by the number of CPUs on the host.

  This metric is useful when you want to compare the CPU utilization of two or more probes running on the same host and compare the probe CPU utilization with the system CPU utilization. In this case, the probe CPU utilization metric needs to be adjusted by the number of CPUs of the host so that each of the metrics is less than 100% and also consistent with the host system CPU utilization metric.

- **Total CPU Utilization (Avg).** The value for process CPU utilization for all CPUs on the host.

This metric is useful when you want to compare the CPU utilization of the same application or application server running on different hosts. These hosts may have different numbers of CPUs. So in this case, using the Probe's total CPU utilization (total of all CPUs) is useful for comparisons. Note that this total process CPU utilization metric is not adjusted by the number of CPUs so it may be greater than 100% if the probe is running on a multi-processor system.

The probe CPU Utilization metrics are supported on the following platforms: Windows, Solaris, AIX, HP-UX and Linux (kernels 2.6.10 or later).

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 CPU time 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 *HP Diagnostics Installation and Configuration Guide*.

Collection of CPU Time Metrics

The CPU Time metrics appear in the Details pane for various views such as the Server Requests view, the Transaction view, the Probes view, the Call Profile view, the Portal Components view and the Diagnostics Profiler. You can enable, disable and configure the collection of CPU time metrics. The CPU time metrics are **CPU (Avg)** and **CPU (Total)**. If collection of CPU time metrics is disabled or not configured for methods, you will see N/A in the Details pane for these metrics.

The CPU Time metrics rely on CPU timestamping which is generally supported on the following platforms: Windows, Solaris, AIX, HP-UX and Linux kernels 2.6.10 or later (for example RedHat 5.x, SUSE 10.x).

Note: Support for CPU timestamping can vary however, not only by operating system, but also by platform architecture (for example SPARC versus x86).
Please see the Product Availability Matrix at http://support.openview.hp.com/sc/support_matrices.jsp. for information on support for CPU Time on specific platform versions and architecture.

Important: In VMware, the CPU time metric is from the perspective of the guest operating system and is affected by the VMware virtual timer. See the VMware whitepaper on timekeeping at http://www.vmware.com/pdf/vmware_timekeeping.pdf. Also refer to the *HP Diagnostics Installation and Configuration Guide* section on Time Synchronization for Probes running on VMware.

CPU Timestamping for Server Requests

By default, collection of CPU time metrics is enabled for server requests. You can disable CPU time metric collection and configure collection of CPU time metrics for both Java and .NET probes as described below.

Java Probe

Collection of CPU time metrics can be configured in property files or using the Java Profiler UI.

For a Java Probe, configure the following property files for the collection of CPU Time metrics:

- **use.cpu.timestamps** property in **<probe_install_directory>\etc\capture.properties**.

This property is set to **true** by default which enables collection of CPU time metrics. But what, if any, CPU timestamps are collected is controlled by a second property listed below. If you set the use.cpu.timestamps property to false, the CPU time metrics will not be collected for any server request or method reported by the probe

- **cpu.timestamp.collection.method** property in **<probe_install_directory>\etc\dynamic.properties**.

Important: Use caution when configuring 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.

Cpu.timestamp.collection.method can be set to one of the following:

- **0 –** No CPU timestamping
- **1 –** CPU timestamps will be collected only for server requests.
- The default value is **1** which means that CPU times can be reported at the server request level but not at the transaction level. However, if the setting is removed or commented out of the properties file, the default is then 0.
- **2 –** CPU timestamps will be collected for All server requests and ALL methods.
- **3** - CPU timestamps will be collected for ALL server requests and the life cycle methods instrumented for Portal Components.

Or you can set the cpu.timestamp.collection.method property using the Configuration tab in the Java profiler as follows:

1. In the **Profiler** UI select the **Configuration** tab. The profiler does not need to be started to make this probe configuration change.

2 In the Configuration screen select a **Collect CPU Timestamps** option from the drop down list.

CPU Timestamp Collection Method	Description
None	No CPU Timestamps.
For Server Requests Only	CPU timestamps will only be collected for server requests.
For Server Requests and Portlet Methods	CPU timestamps will be collected for ALL server requests and the life cycle methods insrumented for portal components.
For Server Requests and All Methods	CPU timestamps will be collected for ALL server requests and ALL methods.

3 When you have completed making changes to the Configuration tab, click **Apply Changes.**

Note: Your changes take effect immediately. There is no need to restart the Probe.

.NET Probe

For a .NET probe the collection of CPU time metrics is controlled by the XML element **cputime** in **probe_config.xml**. The cputime element can be set to the following: none, serverrequest, method. The default is serverrequest.

BPM Transactions

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

6

Working with the Topology Views

Diagnostics has a number of Topology Views that provide you with a visual representation of the interactions and relationships between key application components. Understanding the common layout and controls in the Topology views makes it easier to customize the diagram and select the way you want data displayed.

This chapter includes:

About the Topology Layout

The Topology layout provides a graphical representation of the way your applications or business processes are working. It generates a diagram that shows interactions and relationships between key application components, such as server requests, application servers, and databases.

A Topology diagram is displayed in the following views:

- Standard Views has a **Topology view** for probe connections, probes, probe groups, consumer connections and hosts
- **Application Explorer view** has a topology layout
- SOA Services specialized view group has a **Service Topology view**

To access the different Topology views:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2.  In the View bar, click **Standard Views** to open the Standard Views view group. Click **Topology**. Select an entity in the graph entity table to generate a topology diagram.

   OR

3.  In the View bar, click SOA Services to open the Services specialize view group. Click **Service Topology**. Select an entity in the table to generate a topology diagram.

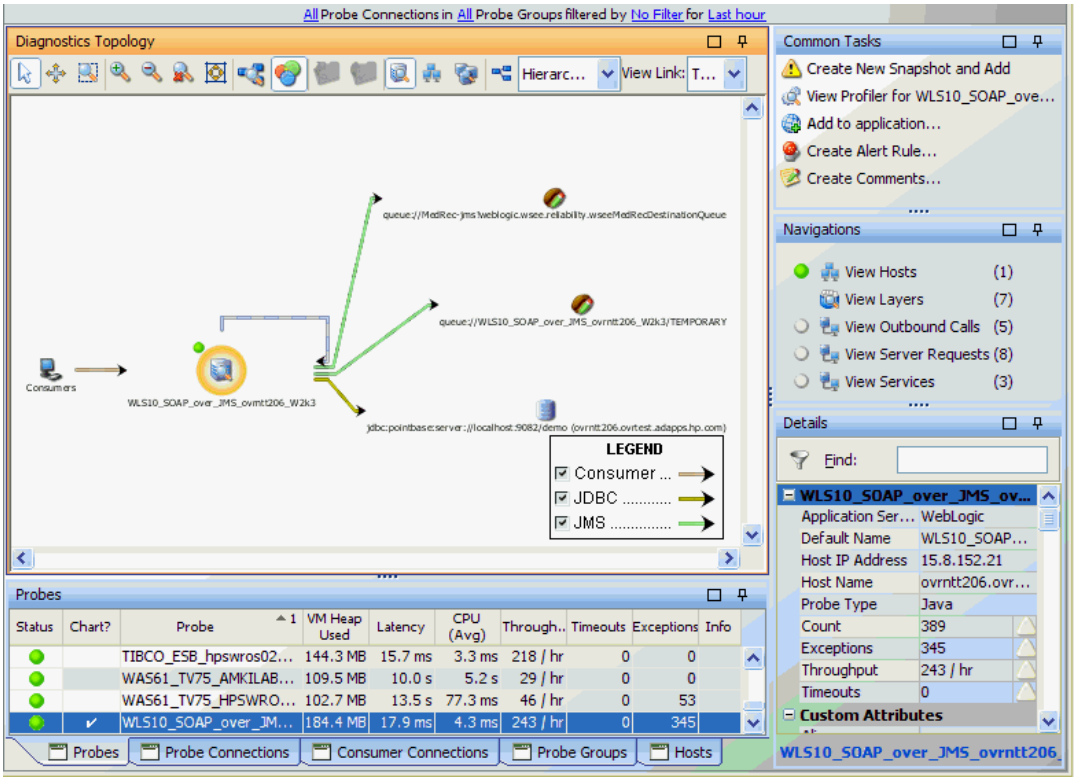
   OR

4. In the Applications window select an application from the Applications list. In the View bar, click **Application Explorer**. The Application Explorer view is not available when you select the Entire Enterprise application.

Note: You can also access the standard Topology view directly from the Diagnostics Applications window by selecting the **viewing the application topology** link.

Description of the Topology Layout

The Topology layout has the appearance and controls of a detail layout except the topology diagram is displayed instead of a graph. The following figure shows an example of the Topology layout.



The topology diagram includes a toolbar with controls that allow you to pan, zoom, and alter the way the diagram is drawn by grouping entities, changing diagram type, or changing connection (link) information.

Note: The Application Explorer includes a topology display that is similar, but some functions are different. Please see Chapter 12, “Application Explorer View” for details.

Topology Diagram

The topology is a mapping of entities in an application and the connections between entities. The topology diagram provides you with a visualization of your application or business process. See "The Topology Diagram" on page 149 for details.

When you first access the Standard Topology view, you see a message in the place of the topology diagram instructing you to select a entity in the table to generate a topology.

Entity Table

The entity table in a topology layout is used like the table in any detail layout. You select an entity in the table to generate a topology for that entity. All other entities with connections to the entity you select in the table are marked with a check mark in the chart column and are also included in the topology. You can filter the data by **Probe Connections**, **Probe Group** and **Time Range**.

In the Application Explorer topology view there is no entity table. Instead you will see a status and navigation pane and a metrics summary pane.

A series of tabs is available at the bottom of the view. In the Standard Topology view you can select the following tabs: Probe Connections, Probes, Probe Groups, Hosts, Consumer Connections.

Common Tasks and Navigations

The common tasks and navigations pane in the Topology view and the Service Topology view displays the links that are available for the entity you select in the entity table. The relevant tasks and navigations will vary depending on the type of entity you select.

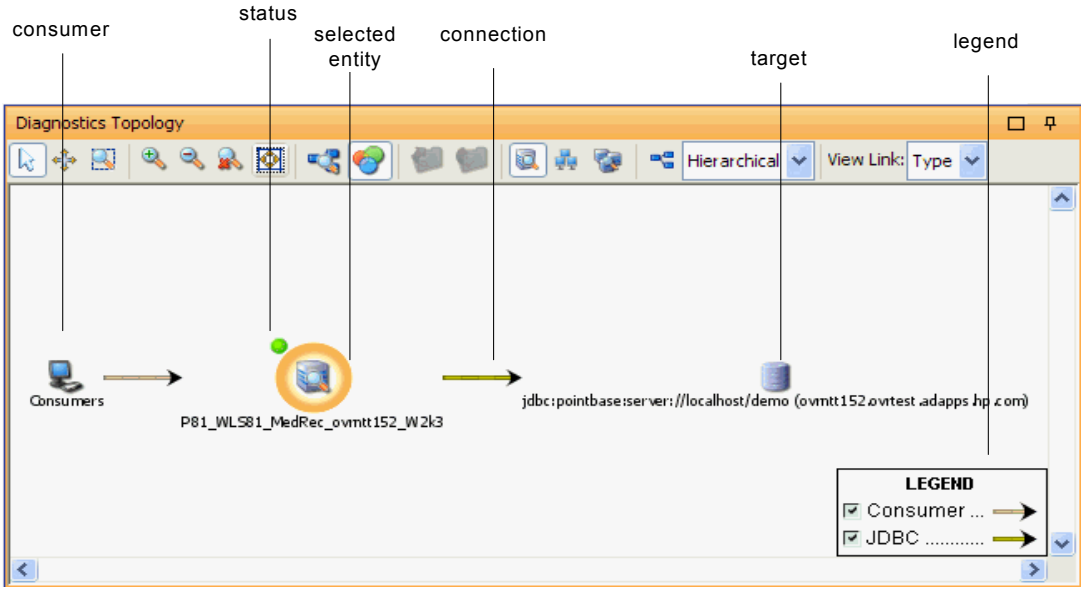
In the Application Explorer topology view the relevant navigations are shown in the status and navigations pane.

Details

The Details pane in the Topology view and the Service Topology view lists the metrics for the selected row in the entity table.

The Topology Diagram

The objects that appear in the topology diagram are shown in the following figure, and are described in the section that follows. See “The Topology Toolbar” on page 151 for a description of the items in the toolbar.



Selected Entity. Entities with a blue ‘halo’ are selectable in the topology. When an entity is selected, it is displayed with a yellow halo. Metrics for a selected entity are displayed in the details pane. You can right-click entities on the diagram to see a tooltip containing information on that entity.

Status. The colored dot next to an entity in the topology indicates the status of the entity. Status is based on a variety of factors, including metric thresholds. Status is only available when you view time ranges that are an hour or less. Status is shown as gray when no status is available, or when the time range exceeds an hour. See “About the Graph Entity Table” on page 101 for more information on how status is calculated.

Connection. The connection arrows represent aggregated calls to applications within your environment. Connections can be shown between various entities such as probes, probe groups, Web services and consumers, or targets. The arrow on the connection indicates the direction of the communication.

The color of the connection arrow can indicate the type of call or performance status and load, depending on the selection from the View Link drop-down menu in the topology toolbar. When you click a connection in the topology, the line is emphasized with a black stripe (indicating that you have selected it).

Consumer. The consumer icon represents calls to a probe (or other entity) coming from a non-instrumented source (not monitored by a probe). These calls are grouped together in the topology under the consumer node. You can't select the consumer icon but if you want to see which consumers are making the calls represented in the topology you can select the consumer connections arrow coming from the consumer icon. The Consumer Connections tab then opens to display details about all the different consumers making calls to a particular entity.

You can configure consumer IDs to identify specific consumers. The default consumer ID is the IP address but you can configure consumers in a number of ways. "About Consumer IDs" on page 433.

Target. The target icon represents calls to a non-instrumented source (not monitored by a probe) or when multiple connections have the same origin or destination. 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. You cannot select targets in the diagram since targets do not contain any metric information.

You could see the following entities in a topology, each represented by different icons.

- Probes
- Probe Groups
- Hosts
- Web Services

- Consumers
- Connections
- Targets.

The Topology Toolbar

The topology toolbar appears at the top of the topology display, below the view title. The toolbar allows you to control the appearance of the topology by scrolling and zooming, and by toggling the appearance of useful aids, such as the topology overview box and the legend.

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 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. 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 the mouse button. 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 **Select Mode**.

Zooming

Diagnostics provides several controls that allow you to zoom in 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 no longer affects the appearance of the diagram.

Zooming In

To zoom in a selected area of the diagram, click **Zoom Mode** to put the pointer into zoom mode. When in zoom mode, the pointer is shaped like a magnifying glass. 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 Mode** again, or click **Select Mode**.

To zoom in the entire diagram to get a closer look at an area of the diagram, click **Zoom In**. Diagnostics zooms in 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.

Zooming Out

To zoom out to get a more complete view of the diagram, click **Zoom Out**. Diagnostics zooms out 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.

Fit Contents to View

To zoom to fit the topology diagram into the view, click **Zoom Contents to Fit View**. Diagnostics zooms 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.

Restoring the Diagram to Primary Zoom

Diagnostics sizes the diagram for the best fit to your browser window when it first displays a diagram. You can choose to zoom the diagram in or out. Then you can restore the diagram to this primary zooming resolution by clicking **Reset Zoom**. The custom layout changes that you made to the position of probes and probe connections are not impacted.

Diagram Overview

The Topology diagram overview provides you with a view of the entire diagram. The diagram overview helps you to scroll and zoom the diagram, so the information that you need is visible in the Topology view. The diagram overview is especially helpful when you have zoomed in on the 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 then dragging it 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 lower-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 currently displayed diagram. It shows the specific color used to draw each probe connection type.

To display the diagram legend, click the **Diagram Legend** button in the diagram toolbar. Diagnostics displays the legend in the lower-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 or load are to be diagramed. 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, in which the number of probe connections can make it difficult to understand.

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

Grouping Probes

You can change the way the diagram is drawn by selecting to group probes by hosts or by probe groups. Grouping probes can simplify the appearance of the diagram.

Grouping Probes by Hosts

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

Grouping Probes by Probe Group

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

Ungroup Connected Probes

To cause the probes to again be displayed as individual probes in the diagram, click **Don't Group Connected Probes** in the toolbar.

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 this icon, or right-click the group icon, the group is opened so you can see each of the probes in the group, along with the probe connections to the targets.

With a large topology it may be necessary to click the Layout button after ungrouping probes to properly display overlapping connections.

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.

Adjusting the Layout

The topology diagram can be drawn in several different layouts. 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** drop-down menu. Diagnostics redraws the diagram using the selected layout.

The drop-down menu for Layout allows you to select layout types including:

- **Hierarchical.** Organizes nodes without overlaps in horizontal or vertical levels. Arranges the graph such that the majority of links are short and flow uniformly in the same direction (from left to right, from top to bottom, and so on). Reduces the number of link crossings. Most of the time, produces drawings with no crossings or only a small number of crossings.
- **Tree.** Takes into account the size of the nodes so that no overlapping occurs. Optionally, reshapes the links to give them an orthogonal form (alternating horizontal and vertical line segments).
- **Uniform.** Often provides a drawing without any or with only a few link crossings and with approximately equal length links for small- and medium-size graphs having a small number of cycles. The maximum number of nodes for which you can use the algorithm depends on the connectivity of the graph and is difficult to predict.
- **Spring.** Often provides a drawing with no link crossings or few crossings for small- and medium-size graphs. The maximum number of nodes for which you can use this layout depends on the connectivity of the graph: hundreds for trees, dozens for cyclic graphs.

Customizing the Diagram Layout

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

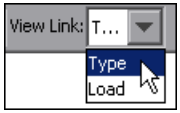
Note: Diagnostics redraws the diagram when an entity has been detected that must be added to the diagram, or when an entity must be removed from the diagram. If the view is zoomed in, the addition of new nodes to the diagram does not cause the diagram to zoom to fit the contents.

Restoring the Layout

When you have customized the diagram layout by repositioning probes and targets, you can have the diagram redrawn clicking **Layout all Nodes** in the diagram toolbar. Diagnostics redraws the diagram, keeping your zoom and pan settings as you set them.

View Link: (Type or Load)

The View Link drop-down menu shows information about probe connections in the current Topology diagram.

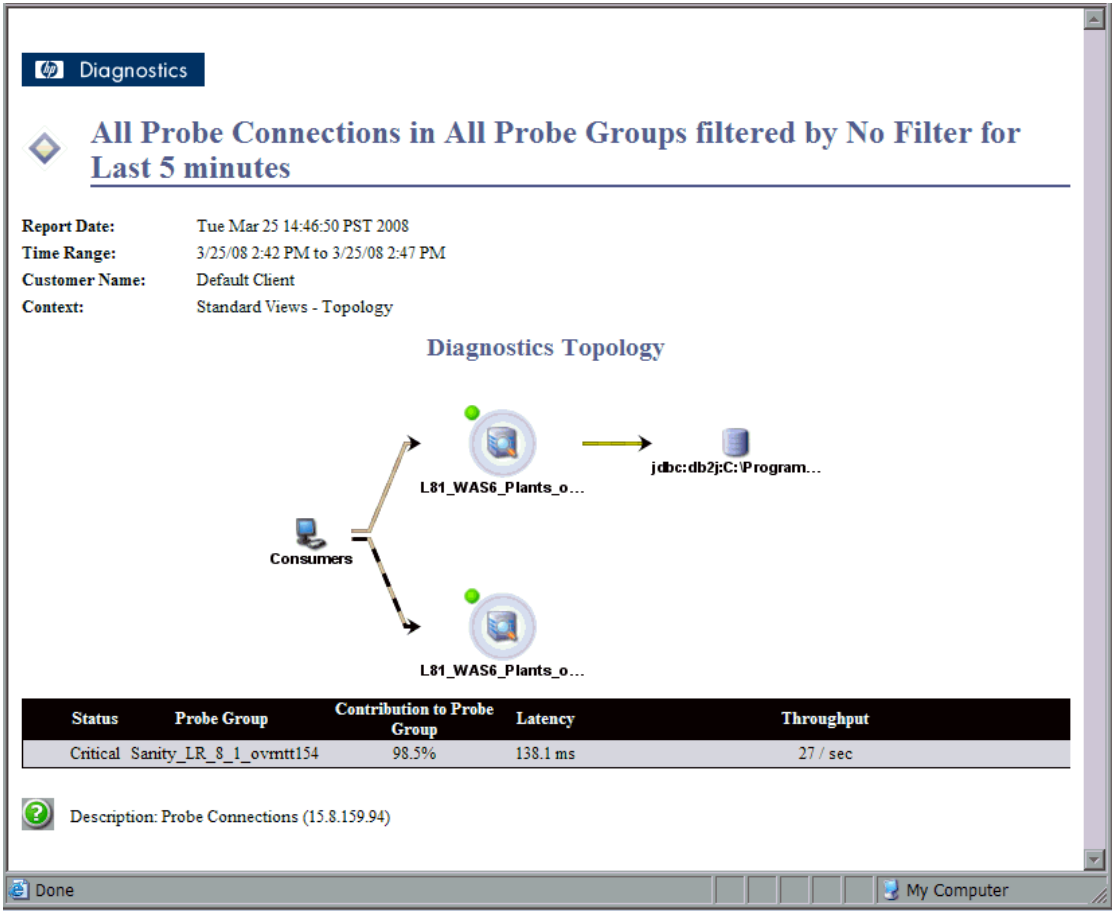


By default, the connections are initially colored according to the Type of call (for example, one color for ADO, another for JDBC, and so on).

You can use the View Link drop-down menu to select **Load**, which uses line thickness and color to indicate the connection's frequency of use and latency time combined (thick as heavy, thin as light) as well as its performance level status (green as good, yellow as warning, and red as critical problem).

Saving the Topology to an HTML Web Page

You can save the current Topology as an HTML web page. When you have the view you would like to capture, click the **Saves this view as an HTML web page** button in the View toolbar (upper-right side of the view). The exported view shows the information in the current display as a web page that is also saved to an HTML file you name. The following screen is an example of a Topology HTML web page.



Note: Whatever diagram display resolution appears at the time of your export is duplicated in your report.

For the Standard Topology view, the Probe Connections tab of the data entity table is what is saved to your HTML file.

7

Working with Alerts

In Diagnostics you can create alert rules and notifications to be triggered when entity status or metric values exceed a specified threshold.

This chapter includes:

About Alert Notification

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 131.

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.

There are two types of alert notification rules:

- Those based on entity status change occur when any metric for the entity violates a threshold.
- Those based on metric status change occur only when the specified metric violates a threshold.

Diagnostics can trigger alert notifications when the status of the entity (or metric) 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. An entity that remains in critical status will not continuously cause the generation of alert notifications.

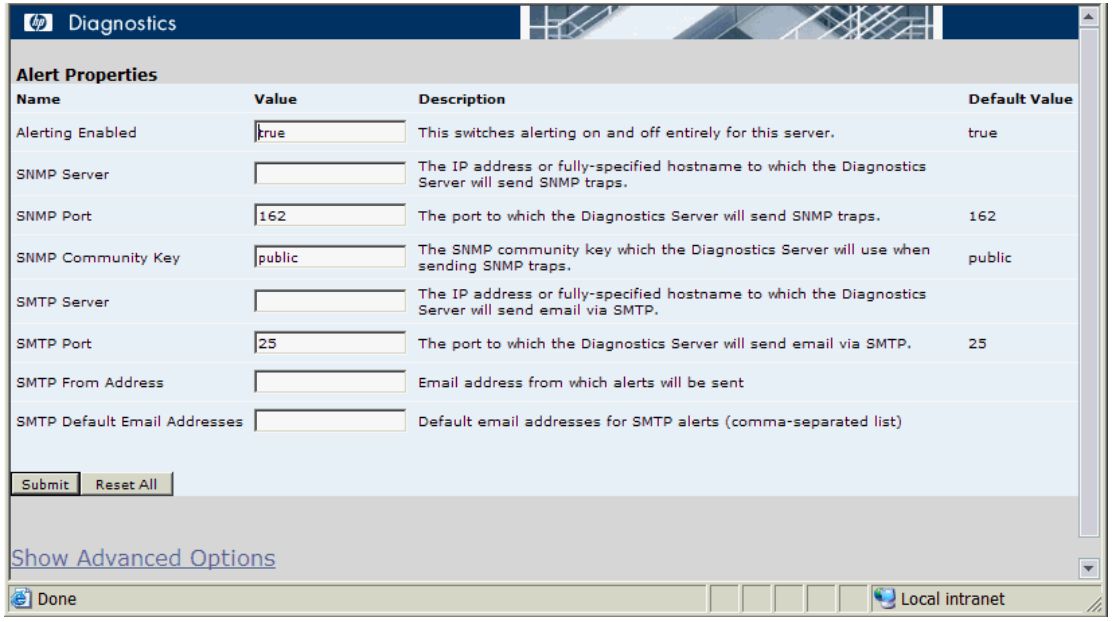
Configuring Alert Notification

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 (select Show Advances Options to see the Alert Properties page).

Important: 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.

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:



Accessing the Alert Properties Page

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 UI:

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** > **HP 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 164.
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 164.
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.

Notes:

- 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.
- You configure Diagnostics alert notification properties from the Alert Properties page for each Diagnostics Server in Mediator mode.

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 164.
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 164.
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 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, for all views, 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 entities such as applications, 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. Alert rules can be based on the status of an entity or the status of a metric collected for an entity.

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 alert notification rules in a number of ways:

- **Create an alert rule based on entity status.** Right-click on the entity in a graph entity table or status view and select one of the alert rule options from the pop-up menu. Or you can also use the Common Tasks pane to maintain alert rules for an entity. The options displayed in the menu vary depending on whether alert rules exist or not. Only one alert notification rule can be created for each entity.

➤ **Create an alert rule based on a metric status.** Select the Alert Rule indicator icon next to the metric in the Details pane.

Note: You will receive two alert events if both an entity status change and a metric status change rule are set.

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 163.

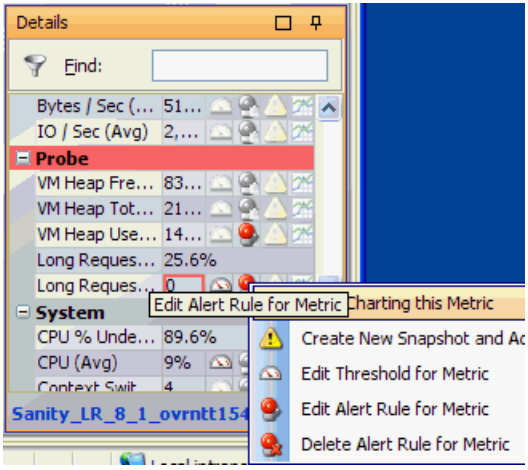
To create or maintain an alert notification rule for an entity:

1 Right-click on an entity in 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 box 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**. To delete an existing alert rule for an entity select the **Delete Alert Rule** menu option and select Yes in the Delete Alert dialog box.

To create or maintain an alert notification rule on a metric:

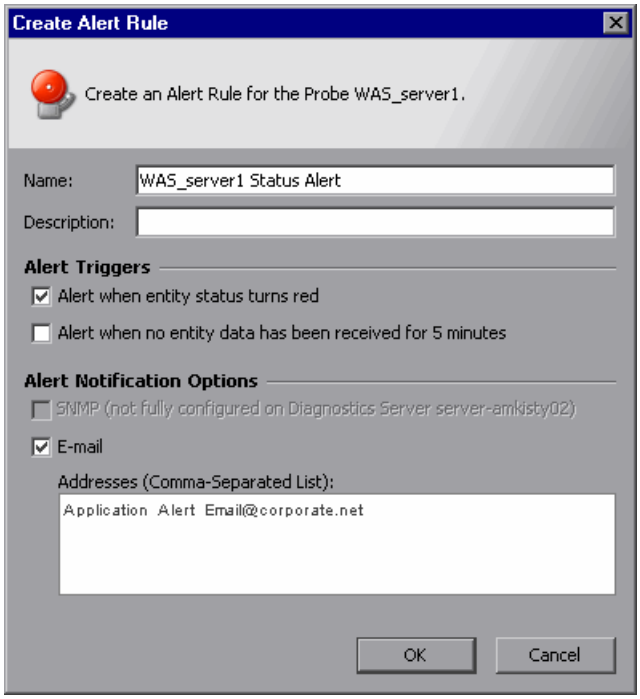
1 In the Details pane select the Alert Rule indicator icon next to the metric you want to use in an alert rule.



2 The Create Alert Rule dialog box is displayed allowing you to create a new alert rule based on this metric. Only those metrics that can be used in an alert rule will show the Alert Rule indicator icon. If an alert rule has already been created for the metric the Alert Rule indicator icon will be highlighted (red, Active Alert Rule). See “Understanding the Columns in the Alert Rules View” on page 173 for a description of the Active and Disabled Alert Rule indicator icons. To edit or delete an alert rule for the metric, right-click the Alert Rule indicator icon and select **Edit Alert Rule for Metric** or **Delete Alert Rule for Metric**.

Note: For entity status alert rules you can select multiple entities in a graph entity table and then select the menu item to create, replace or delete alert rules. See “Selecting Multiple Rows in a Table” on page 111 for how to select multiple rows. Typically, when selecting multiple entities you should select groups of entities that you want to have the same alert rules.

The Create Alert Rule dialog box as shown in the following example:



1 Enter a **Name**. The name is either for the entity (to help you to remember the entity and the reason why you wanted to receive alert notifications) or the metric name.

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.

2 Enter a short **Description** for the rule.

3 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.

If you do not select at least one Alert Trigger, the alert notification rule is created but is disabled. So to **disable an alert rule**, de-select both of these checkboxes.

4 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.

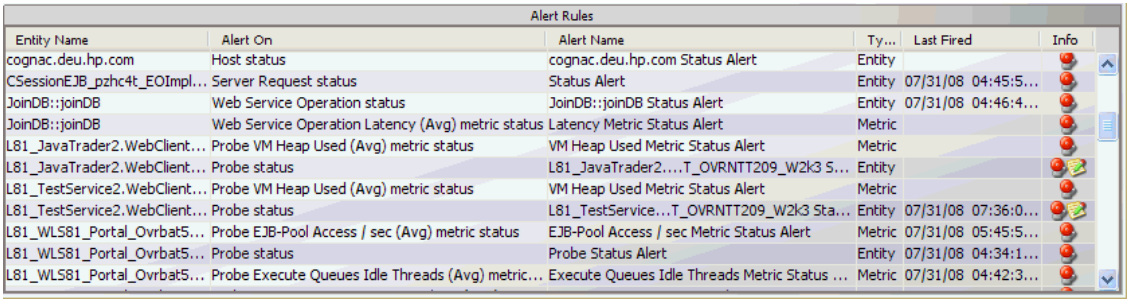
Reviewing Alert Notification Rules

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.

Note: Alert Rules cannot be filtered by application so in an application context you will not see the Alert Rules selection in the view bar under Alerting.

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 select right-click menu options to view, edit, delete the alert notification rules and navigate to the Diagnostics views that are associated with the entities that have alert notification rules. You can select multiple rows in the table to delete multiple alert rules.

The following is an example of the Alert Rule view:



In this example, eleven alert notification rules are displayed: six based on entity status and five based on metric status. All of the alert notification rules are currently enabled. In addition, the example shows that two of the entities have comments associated with them.

Understanding the Columns in the Alert Rules View

The columns in the Alert Rules view provide the same information and navigation options as their 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.** The name of the entity listed in the view. When the alert rule is on a metric, this column shows the parent entity name. 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 On.** What status change the alert rule is based on. For example, Probe status, Host status or Probe VM Heap Used (Avg) metric status.
- **Alert Name.** The name of the alert notification rule.
- **Type.** The type, either entity or metric, listed in the row.
- **Last Fired.** 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 or 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. (You can disable an alert rule in the Alert Rule dialog box by de-selecting both checkboxes for Alert Triggers.)

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

Reviewing Alert Notification Events

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

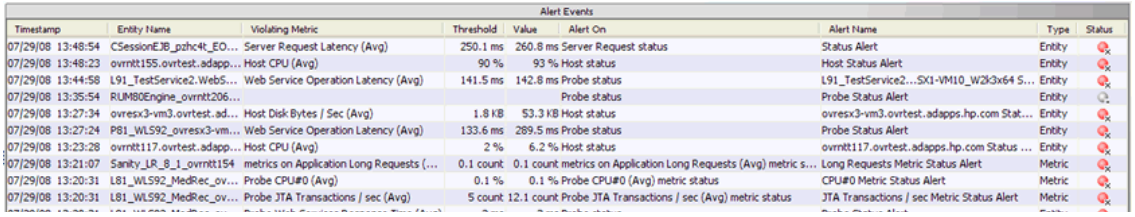
Using the Alert Events View

Diagnostics lists each of the events that triggered an alert in the Alert Events view. By default, 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.

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

Understanding the Columns in the Alert Events View

The following is an example of the Alert Events view.



Alert Events

Timestamp	Entity Name	Violating Metric	Threshold	Value	Alert On	Alert Name	Type	Status
07/29/08 13:48:54	CSessionEJB_pzhc4t_EO...	Server Request Latency (Avg)	250.1 ms	260.8 ms	Server Request status	Status Alert	Entity	
07/29/08 13:48:23	ovrntt155.ovrtest.adapp...	Host CPU (Avg)	90 %	93 %	Host status	Host Status Alert	Entity	
07/29/08 13:44:58	L91_TestService2.WebS...	Web Service Operation Latency (Avg)	141.5 ms	142.8 ms	Probe status	L91_TestService2...SX1-VM10_W2k3x64 S...	Entity	
07/29/08 13:35:54	RUM80Engine_ovrntt206...				Probe status	Probe Status Alert	Entity	
07/29/08 13:27:34	ovresx3-vm3.ovrtest.ad...	Host Disk Bytes / Sec (Avg)	1.8 KB	53.3 KB	Host status	ovresx3-vm3.ovrtest.adapps.hp.com Stat...	Entity	
07/29/08 13:27:24	P81_WLS92_ovresx3-vm...	Web Service Operation Latency (Avg)	133.6 ms	289.5 ms	Probe status	Probe Status Alert	Entity	
07/29/08 13:23:28	ovrntt117.ovrtest.adapp...	Host CPU (Avg)	2 %	6.2 %	Host status	ovrntt117.ovrtest.adapps.hp.com Status ...	Entity	
07/29/08 13:21:07	Sanity_LR_8_1_ovrntt154	metrics on Application Long Requests (...	0.1 count	0.1 count	metrics on Application Long Requests (Avg) metric s...	Long Requests Metric Status Alert	Metric	
07/29/08 13:20:31	L81_WLS92_MedRec_ov...	Probe CPU#0 (Avg)	0.1 %	0.1 %	Probe CPU#0 (Avg) metric status	CPU#0 Metric Status Alert	Metric	
07/29/08 13:20:31	L81_WLS92_MedRec_ov...	Probe JTA Transactions / sec (Avg)	5 count	12.1 count	Probe JTA Transactions / sec (Avg) metric status	JTA Transactions / sec Metric Status Alert	Metric	

The columns are:

- **Timestamp.** The time at which the alert event was triggered.

- **Entity Name.** The name of the entity listed in the view. When the alert event is on a metric, this column shows the parent entity name. 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.
- **Violating Metric.** This indicates the entity type and metric that caused the event.
- **Threshold.** 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.** The value for the metric when the alert was triggered.
- **Alert On.** The entity status or metric status change the alert rule is based on. For example, Probe status or Probe VM Heap Used (Avg) metric status.
- **Alert Name.**The name of the alert notification rule that was triggered by the alert event.
- **Type.** The type, either entity or metric, listed in the row.
- **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. See "About the Graph Entity Table" on page 101 for details on how Status is calculated.

  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	**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 recent 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.

The following describes, in general, how a status of RED or YELLOW is determined:

- Diagnostics compares the average value for the metric in the last five minutes to the threshold. If the threshold is crossed, then the status is RED.
- If not, Diagnostics counts the total number of points (five second averages from the probe) that violate the threshold. If the count is greater than three but the average is still under the threshold (current status is not RED), then the status is YELLOW; otherwise the status stays at GREEN.
- If none of the above, the status is GREEN.

Viewing Alert Event and Alert Notification 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** > **HP 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/status_changes.log** to view the log messages for all status change events for the Diagnostics Server.

   You can also view a log of the alert notifications sent on status change events. Select the link for **<diagnostics_server_install_dir>/log/alerting.log**.

Viewing Threshold Violations

From the Alert Events 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 select 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. The threshold violation is shown with a red plus sign icon.

You can also double-click an alert event row to go to the relevant view of the threshold violation.

8

Performing Snapshot Analysis

When diagnosing a performance problem you can create snapshots with relevant data for further analysis.

This chapter includes:

Snapshot Analysis

When you see a performance anomaly, you can use Diagnostics to capture the performance of your application, and then store the information as a snapshot. A snapshot 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 snapshot was captured. With the information captured in a snapshot, you can investigate threshold violations and other performance issues, using the Analyze Snapshots view and the other Diagnostics views. Snapshots provide you with the metrics for the specific slice of time in which the performance of your application was not as you expected. These metrics help you understand what is causing the performance issue.

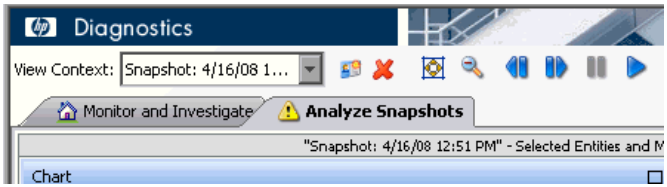
You control when Diagnostics captures a snapshot. When you see a performance anomaly that you want to analyze, you can use the controls in the Diagnostics views to create a snapshot that includes the performance metrics for the specified time period.

Data for the time range of a snapshot are preserved. They are exempt from routine data purging. This time-range preservation enables you to save a snapshot that shows the symptoms of a problem. Later, you can use the snapshot to help troubleshoot similar problems. When you delete a snapshot, the time range is removed from the global list of preserved times. The data is purged.

If you have multiple snapshots in the same time range (or overlapping time ranges), the data is preserved in the overall time range, independent of snapshots. This snapshot-independent preservation enables you to delete a snapshot without removing data that is a part of another snapshot.

Monitoring Performance Versus Analyzing Snapshots

The Monitor and Investigate tab and the Analyze Snapshots tab behave differently, depending on your selection from the **View Context** drop-down menu.



When you select **Monitoring f**rom the **View Context** drop-down menu, Diagnostics is in Monitoring mode. 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.

When you select a snapshot from the **View Context** drop-down menu, Diagnostics is in Snapshot Analysis mode. As soon as you select a snapshot, Diagnostics opens the **Analyze Snapshots** view in the Analyze Snapshots tab for the selected snapshot. The view displays the entities and metrics that you specified shown in the tables and graph.

You can modify the snapshot to control the entities and metrics that are included in the Analyze Snapshots view. You can then visually compare and correlate the metrics from multiple entities of different types. See “Working with Entity-Metric Pairs” on page 187 for details.

In general, the controls in Snapshot Analysis mode work the same as the controls in Monitoring mode. However, you should be aware of the following differences in Snapshot Analysis mode:

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

- When you are in Snapshot 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 Snapshot 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 snapshot by clicking the **Use Time of Snapshot...** 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 Snapshot Analysis mode or in Monitoring mode. These views are not impacted by your choices in the **Viewing Time** filter and the **View Context** drop-down list.

Creating and Managing Snapshots

If you see a performance issues while you are monitoring your application's performance in Diagnostics, you can create a snapshot capture of the data associated with the performance issue. These snapshots are available for analysis in the Analyze Snapshots view.

Create a New Snapshot

You can create a snapshot capture using the controls of the graph entity table and the details pane. When Diagnostics creates a new snapshot, it creates a default snapshot name, based on the starting date and time of the period for which the snapshot was captured. Diagnostics also sets up the Analyze Snapshot view to display the entities and metrics that are appropriate for the trigger that caused the snapshot to be captured.

To create a new snapshot 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 Snapshots view of the new snapshot. Note that you can select multiple rows in the graph entity table to add to the snapshot.
2. Select the **Create New Snapshot and Add** menu option either from the right-click pop-up menu for the entity, or from the Common Tasks menu.

Diagnostics creates a new snapshot for the time period displayed in the view. The new snapshot is selected automatically from the **View Context** drop-down menu as Diagnostics switches to Snapshot Analysis mode. The Analyze Snapshots view opens with the selected entity listed in the graph entity-metric table, and with the same metric charted in the graph as was charted on the view from which you captured the snapshot.

To capture a new snapshot from the details pane:

1 From the details pane, select the row for the metric that you want to include in the Analyze Snapshots view of the new snapshot.

2 Select the **Create New Snapshot and Add** menu option from the right-click pop-up menu for the metric.

Diagnostics creates a new snapshot for the time period displayed in the view. The new snapshot is automatically selected from the **View Context** drop-down menu as Diagnostics switches to Snapshot Analysis mode. The Analyze Snapshots view opens with the metric you selected charted in the graph, and with the entity to which the metric belongs listed in the graph entity-metric table.

Rename a Snapshot

You can rename a snapshot listed in the **View Context** drop-down list to make it easier to locate in the drop-down list, or to make it easier to recognize the performance issue that the snapshot represents.

To rename a snapshot:

1 Select the snapshot to be renamed from the **View Context** drop-down list.

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

2 Click **Rename Snapshot** to the right of the **View Context** drop-down list.

Diagnostics displays the Rename Snapshot dialog box with the current name of the selected snapshot.

3 Enter the new snapshot name.

4 Click **OK** to rename the snapshot.

Delete a Snapshot

You can delete a snapshot listed in the **View Context** drop-down list so a snapshot that you have already resolved, or a snapshot in which you are no longer interested, does not clutter the list in the drop-down list.

To delete a snapshot:

1. Select the snapshot to be deleted from the **View Context** drop-down list.

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

2. Click **Delete Snapshot** to the right of the **View Context** drop-down list.

   Diagnostics deletes the selected snapshot, so it is not longer listed in the **View Context** drop-down list.

About the Analyze Snapshot View

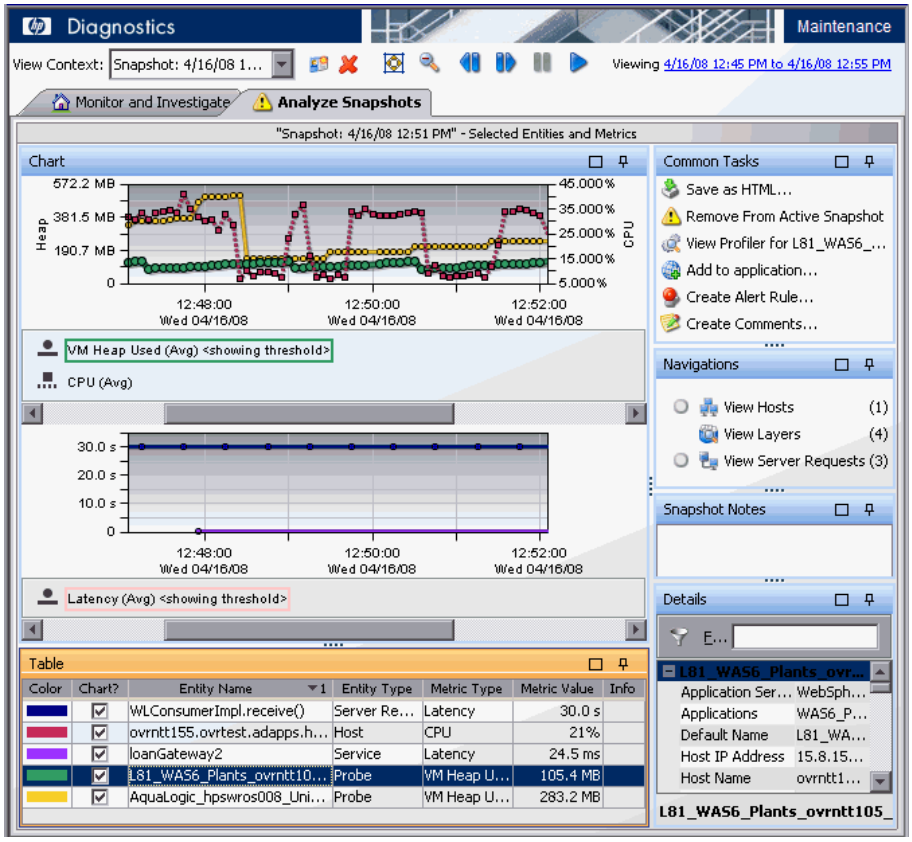
The Analyze Snapshot View on the Analyze Snapshot tab displays the entities and metrics from a captured snapshot that you have selected to be included in this view. This view provides a way for you to compare, contrast, and correlate entities and metrics of different types that would normally only be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

Note: The Analyze Snapshots view is displayed in the Analyze Snapshots tab only when Diagnostics is in Snapshot Analysis mode. When Diagnostics is in Monitoring mode, the Analyze Snapshots tab contains instructions for creating a new snapshot.

The Analyze Snapshots 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 4, “Working with Diagnostics Data Displays.”

Description of the Analyze Snapshots View

The following figure shows an example of the Analyze Snapshots view.



Graph

By default, the charts (graph) in the Analyze Snapshots view contain the charted metrics for the entity-metric pairs listed in the graph entity table. These metrics are the metrics that you have included in the snapshot.

Graph Entity Table

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

Common Tasks and Navigations Panes

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take and relevant navigation links you can follow to further analyze a performance problem and diagnose the cause. The navigation links listed, also show the status of the related entities and their count, where this information is available.

You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed.

For more information see "About the Common Tasks and Navigations Panes" on page 115.

Snapshot Notes

The **Snapshot Notes** area enables you to record the reason you created the snapshot, the steps you took, and the progress you made in understanding the cause of the performance problem captured in the snapshot.

Details Pane

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

Accessing the Analyze Snapshots View

To access the Analyze Snapshots view, Diagnostics must be in Snapshot Analysis mode. Diagnostics is in Snapshot Analysis mode when you select a snapshot from the **View Context** drop-down menu, or when you create a new snapshot from a view while in Monitoring mode. The Analyze Snapshots view displays on the Analyze Snapshots tab when Diagnostics is in Snapshot Analysis mode. For more information on creating a new snapshot see "Monitoring Performance Versus Analyzing Snapshots" on page 181.

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

To access the Analyze Snapshot view using the View Context drop-down list:

Select a snapshot listed in the **View Context** drop-down list:

- 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 snapshot. You can click the Analyze Snapshots tab at any time to see the Analyze Snapshots view for the selected snapshot.
- If you were looking at the Analyze Snapshots 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 snapshot. And the time range changes to that of the selected snapshot.

Customizing the Analyze Snapshots View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Analyze Snapshots view. For more information about the ways you can control how performance metrics are presented in this view, see "Working with Diagnostics Data Displays" on page 81.

Working with Entity-Metric Pairs

As you analyze a snapshot, you discover entities and metrics that help explain the application's behavior. You want to include this information in the Analyze Snapshots view. You also notice that some of the entities and metrics shown in the view are no longer helpful, and should no longer be included in the view. Diagnostics provides a number of ways for you to manipulate the entities and metrics that are included in the Analyze Snapshots view.

Add a Metric to the Analyze Snapshots View

You can choose additional metrics to be charted in the graph of the Snapshot Analysis view, either from the views in the Monitoring and Investigate tab, or from the Analyze Snapshots view.

To add a metric to the Analyze Snapshots view:

1. Select a snapshot from the **View Context** drop-down menu so Diagnostics is in Snapshot Analysis mode.
2. From the details pane in either the Analyze Snapshots view, or in a view displayed in the Monitor and Investigate tab, select the row for the metric that you want to include in the active snapshot.
3. Select the **Add to Active Snapshot** menu option from the right-click pop-up menu for the metric, or click on the icon for the metric.

Diagnostics displays the Analyze Snapshots view. The selected metric is charted in the graph, and an entry for the selected entity-metric pair is listed in the details pane. In the details pane, the icon for the metric is changed to indicate that the metric is included in the view.

Add an Entity to the Analyze Snapshots View

When Diagnostics is in Snapshot Analysis mode, you can chose additional entities to be included in the Snapshot 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 Snapshot Analysis view. You can select an entity to add to the Snapshot Analysis view from the views displayed on the Monitoring and Investigate tab.

To add an entity to the Analyze Snapshots view:

1. Select a snapshot from the **View Context** drop-down menu so Diagnostics is in Snapshot 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 Snapshots view for the active snapshot.
4. Select the **Add to Active Snapshot** menu option, either from the right-click pop-up menu for the entity, or from the Common Tasks pane.

Diagnostics displays the Analyze Snapshots view. The selected entity is included in the view, so each of the metrics that were charted for the entity in 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 table. In the details pane, the icon for each of the charted metrics is changed to indicate that the metric is included in the view.

Delete a Metric from the Analyze Snapshots View

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

Note: When you delete a metric from a view, the entity is no longer included in the view after you delete the last metric for the entity.

To delete a metric from the Analyze Snapshots view:

1. Select a snapshot from the **View Context** drop-down menu so Diagnostics is in Snapshot Analysis mode.
2. From the details pane either in the Analyze Snapshots view, or in a view displayed in the Monitor and Investigate tab, select the row for the metric that you want to remove from the Analyze Snapshots view of the active snapshot.
3. Click the icon for the metric on the selected row, or select the **Remove from Active Snapshot** 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 Snapshots view, the selected metric is no longer charted in the graph, and the entry for the selected entity-metric pair is deleted from the entity-metric table. In the details pane, the icon for the metric is changed to indicate that the metric is no longer in the view.

Delete an Entity from the Analyze Snapshots View

When Diagnostics is in Snapshot Analysis mode, you can remove entities so they are no longer included in the Snapshot 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 Snapshot Analysis view. You can select an entity to remove from the Snapshot Analysis view from the views displayed on the Monitoring and Investigate tab, and from the Snapshot Analysis view.

To delete an entity from the Analyze Snapshots view:

1. Select a snapshot from the **View Context** drop-down menu so Diagnostics is in Snapshot 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 Snapshots view.
3. From the graph entity table, select the row for the entity that you want to remove from the Analyze Snapshots view for the active snapshot.
4. Select the **Remove From Active Snapshot** menu option, either from the right-click pop-up menu for the entity, or from the Common Tasks pane.

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

Maintaining Snapshot Notes

You can enter notes and comments about a snapshot in the Snapshot Notes pane, located above the Details pane on the Analyze Snapshots view. Any text that you enter here is saved when you leave the view. The text is redisplayed the next time you work with the snapshot in the Analyze Snapshots view.

9

Instance Trees

Diagnostics uses an algorithm to select the most interesting instance trees for each server request. Instance tree markers allow you to drill down to the call profiles.

This chapter includes:

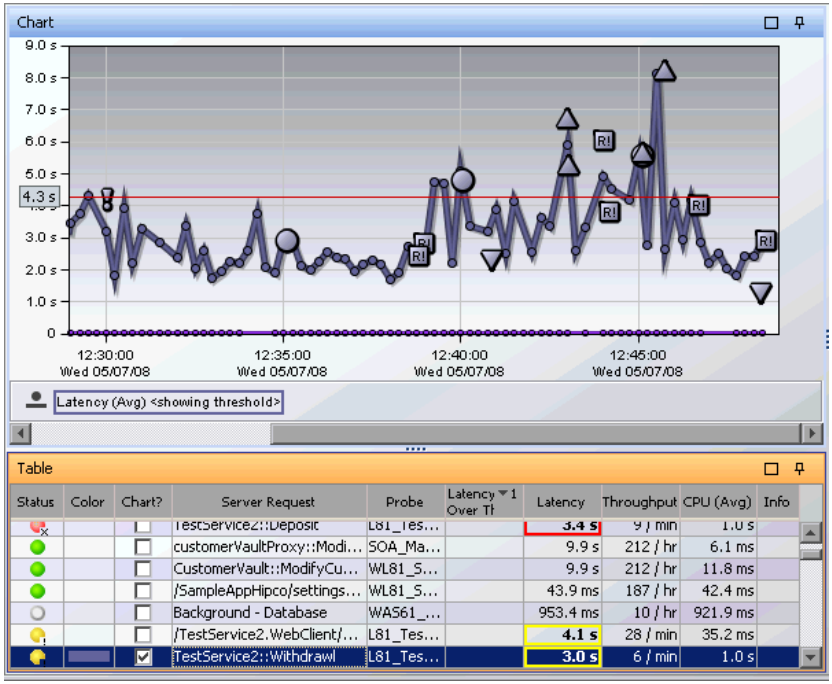
Introducing Instance Trees

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.

Diagnostics does not record every captured trace because of storage considerations. Instead, at five-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, slowest, second slowest and median or average invocation of that server request. Diagnostics also saves 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. And if methods throw exceptions or have SOAP faults, this information is saved and details provided in exceptions instance trees.

An example of a Server Requests view showing some of the instance tree markers is shown below. 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 saved:

- **Maximum Instance Tree**. The maximum instance tree for a server request represents an instance tree with the largest latency for the time period. This tree 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 tree depicts one of the best-performing invocations of the selected server request and resulting root method. It 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 tree depicts a typical invocation of the selected server request and resulting root method call. It can be useful for comparison to instances that perform poorly.

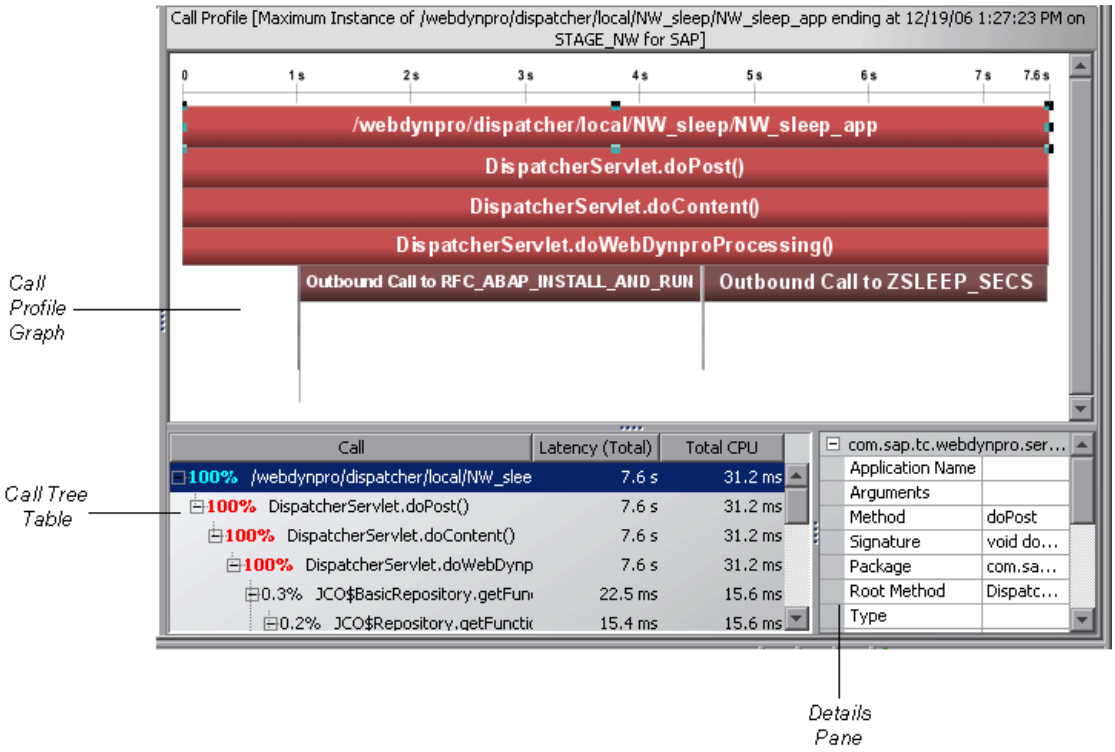
➤ **Cross VM Instance Tree**. The cross VM instance tree for a server request represents an instance tree that includes a call to another server request by using a technology such as RMI or Web services.

The following icons are used to represent exceptions and faults. Additional information is collected by the probe when exceptions or SOAP faults occur. You can drill down to the call profile graph and view exception or fault details for a method, including the stack trace.

➤ **Exception.** The exclamation mark indicates where an exception occurred. When you click the icon, a call profile opens. This call profile has the methods that threw the exceptions marked with a yellow dashed outline and an **Exceptions** tab providing details on the exception. A stack trace for the exception is included with the details.

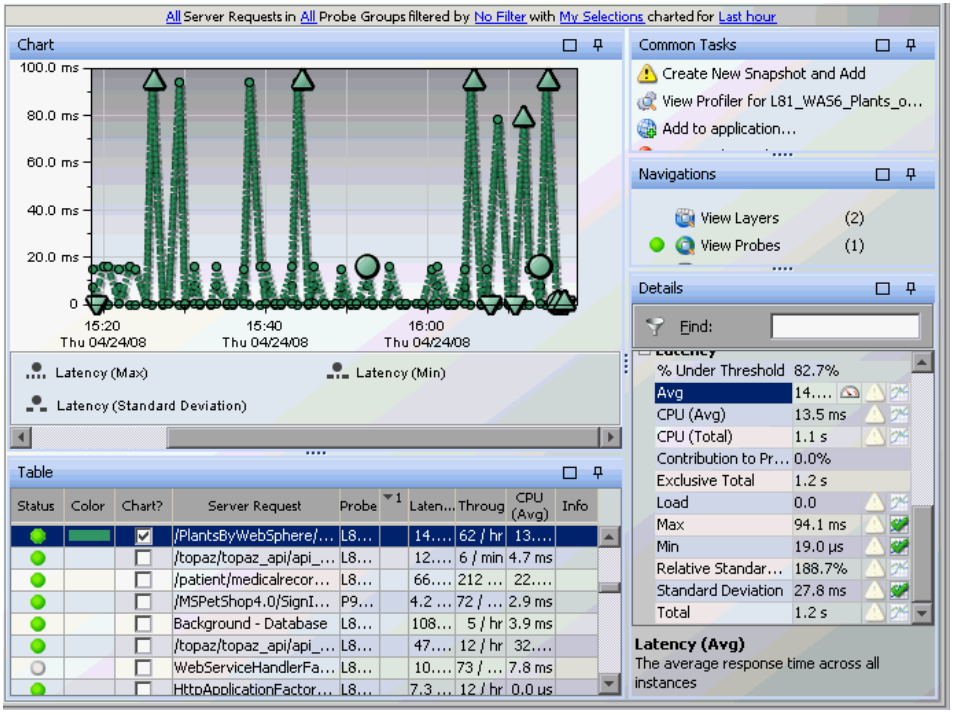
➤ **SOAP Faults.** The icon indicates where an soap fault occurred. See “Types of SOAP Faults” on page 447 for details on the icons used to represent different types of faults. When you click the icon, a call profile opens. This call profile has a SOAP Faults tab providing details on the fault, including the stack trace.

When you select an instance tree icon, the program trace is presented in a call profile visualization as shown in the following example:



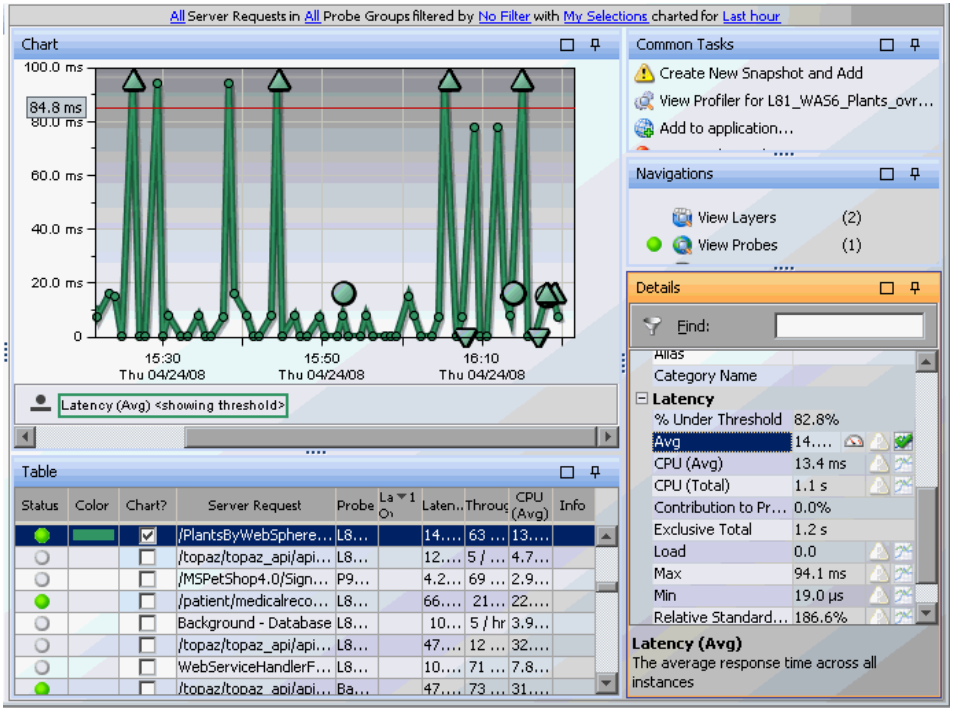
Solving Performance Problems Using Instance Trees

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 **/PlantsByWebSphere/serverlet/ShoppingServlet** server request generally varies between about 10 ms and 20 ms, and sometimes spikes above 90 ms.

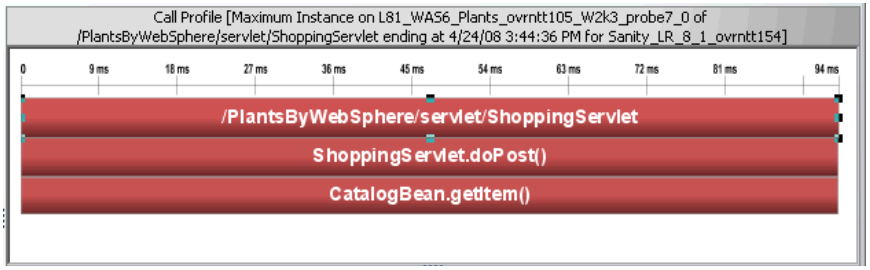
The following image is less noisy. It shows the charted trend for the Average Latency without trend lines for the minimum and maximum latency:



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

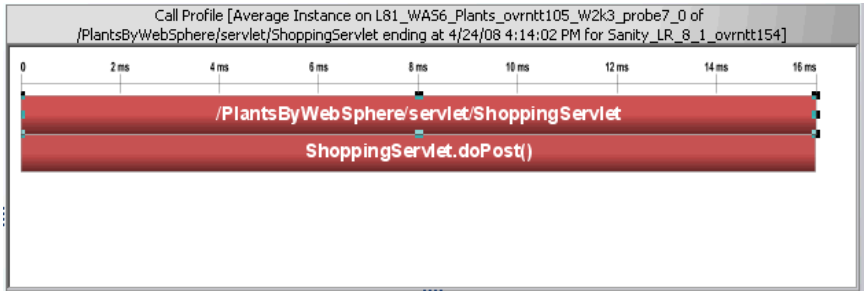
To understand why **/PlantsByWebSphere/serverlet/ShoppingServlet** is sometimes four times slower than average (or five times slower than the best case), compare the captured instance trees against one another to see what is different about each case:

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 getItem method (**CatalogBean.getItem**) 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:

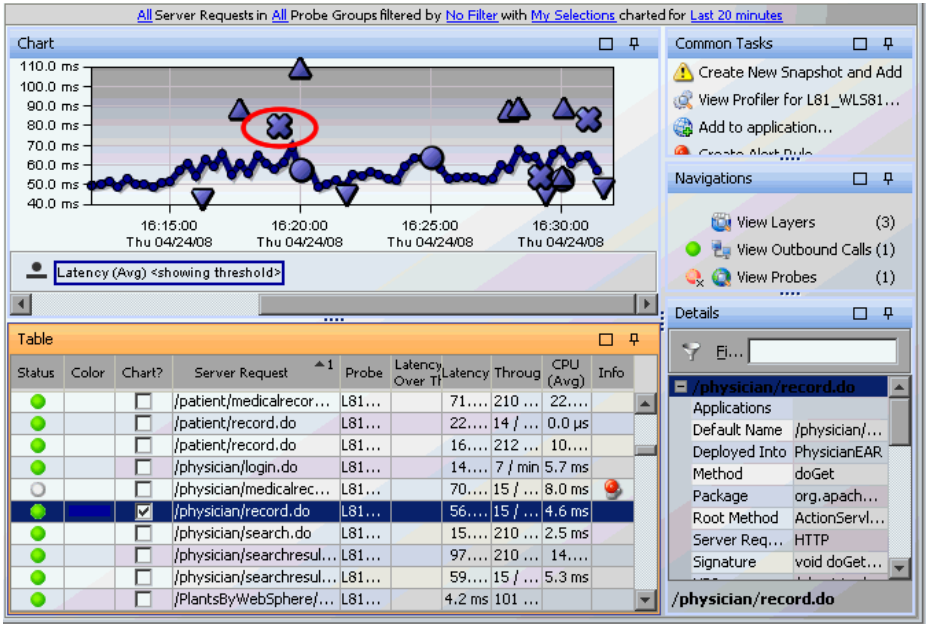
- The EJB is cached in the application server memory and the retrieval is nearly instantaneous. This would be the minimum call profile.
- The EJB must be retrieved from the database. Depending upon usage statistics, this could be the average call profile.
- The server request must wait a significant amount of time for a pooled database connection prior to requesting the EJB data. This could be the maximum call profile.

Cross-VM Trees

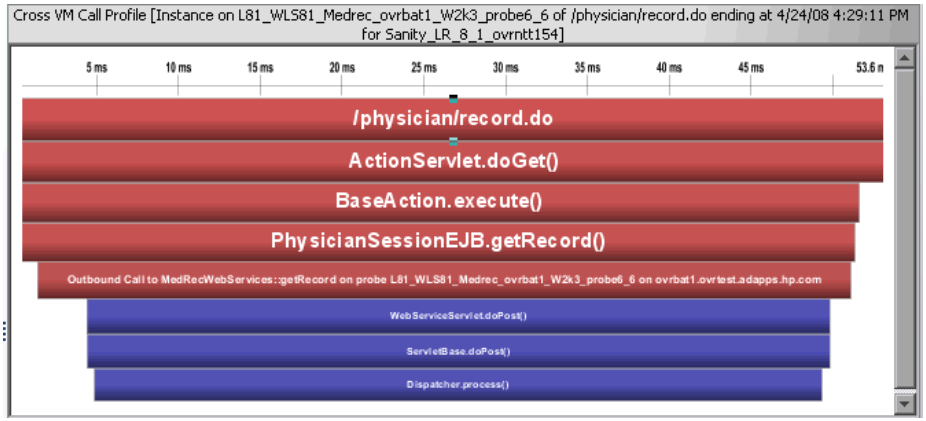
Cross-VM trees are collected to help you quickly understand average application behavior when multiple virtual machines are involved.

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. When you click the icon, the following cross-system call profile opens:

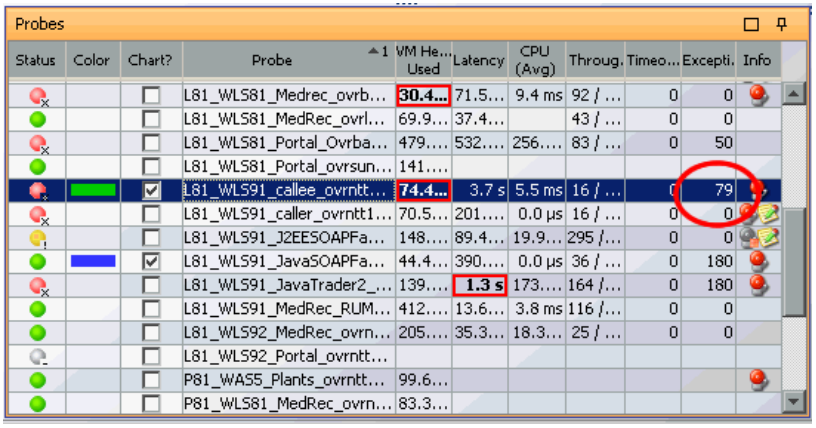


The red nodes in the profile are the processing that took place on the origination system, and the blue nodes are the processing that took place on the child system, **ovrbat1.ovrtest.adapps.hp.com**.

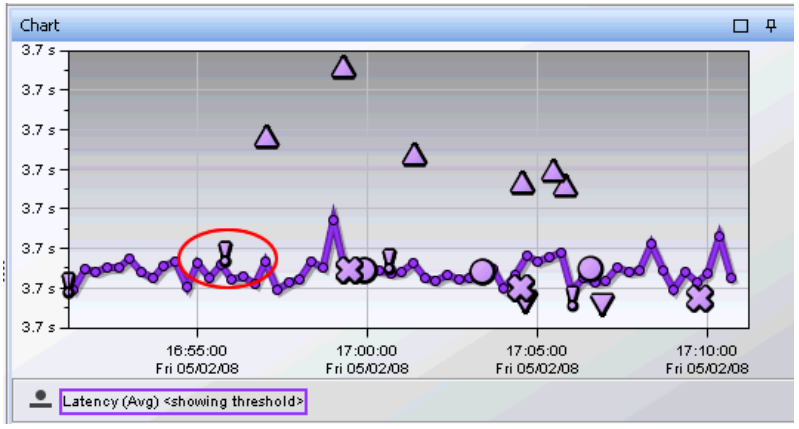
This call tree indicates that over half of the elapsed time for this server request was caused by processing in the child system.

Exception Trees

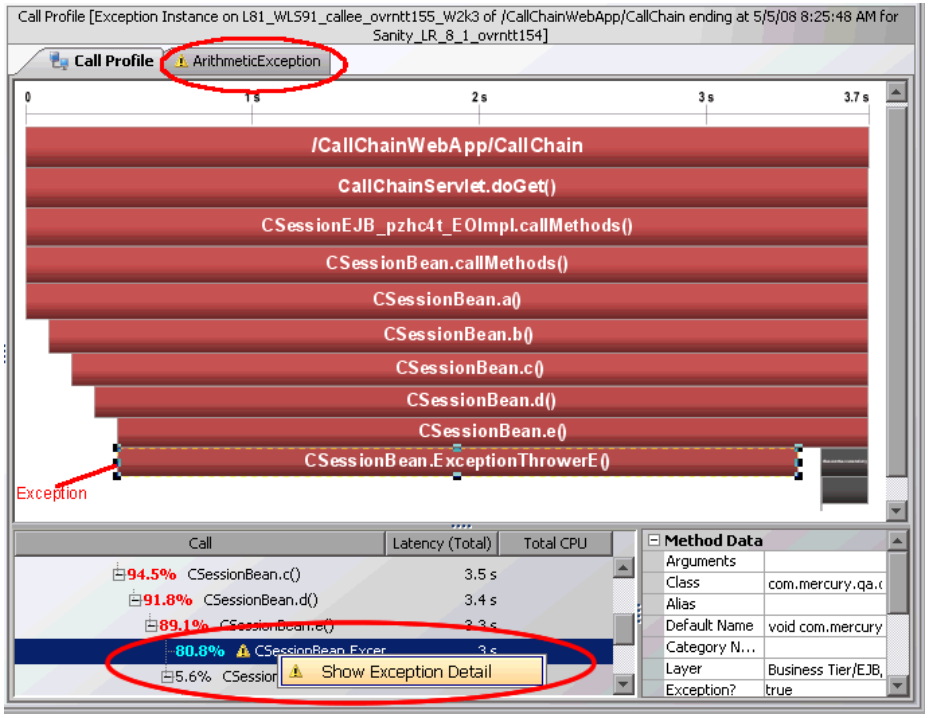
Information on exceptions is also collected by the probe. An example of the Probes view graph entity table is shown below where you can see the number of exceptions thrown for each probe.



If you drill down to the server requests for this probe collection you will see the charted trend with exceptions indicated by an icon.

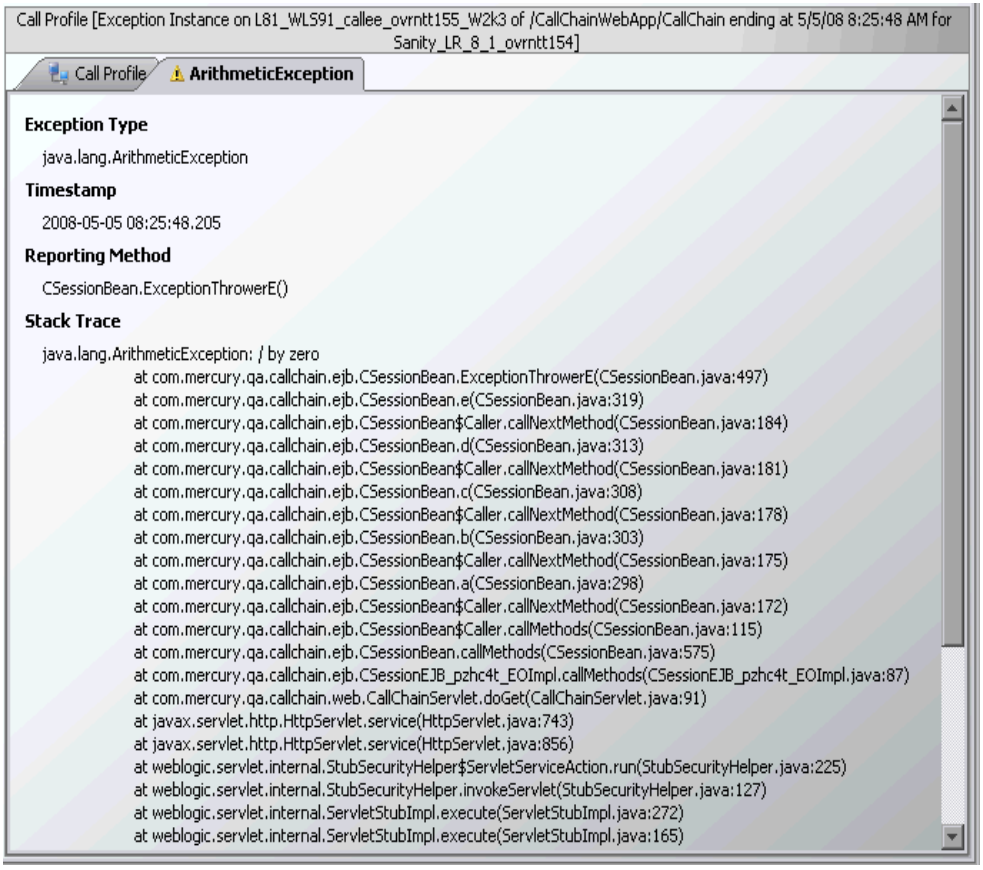


The exclamation mark indicates where an exception occurred. When you click the exception instance icon, a call profile opens.



The call profile has the method that threw the exception marked with a yellow dashed outline. In the call tree the method throwing the exception is marked by a small exclamation mark icon.

You can select the Exceptions tab to get details on the exception including the stack trace. Or you can right-click the exception in the call tree and select Show Exception Detail. This is useful when there are several tabs with detail data. The Exception tab is labeled with the exception name (not including the package and class information).



You can copy and paste the stack trace listing in this view to send to others for use in diagnosing a problem.

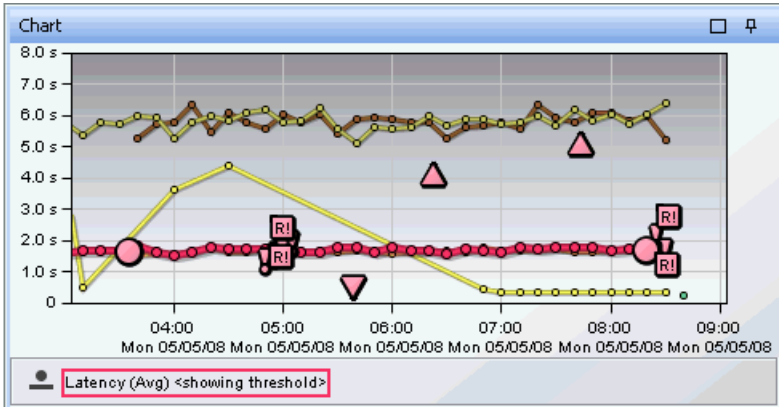
Configuring Collection and Display of Exception Data

By default, every exception that occurs in the monitored application is a candidate for the exception instance tress. Collecting all exception information is usually undesirable however, because exceptions that are not of interest overload the display as well as the data collection and transfer operations. You can therefore limit the exception types for which data is collected by the probe.

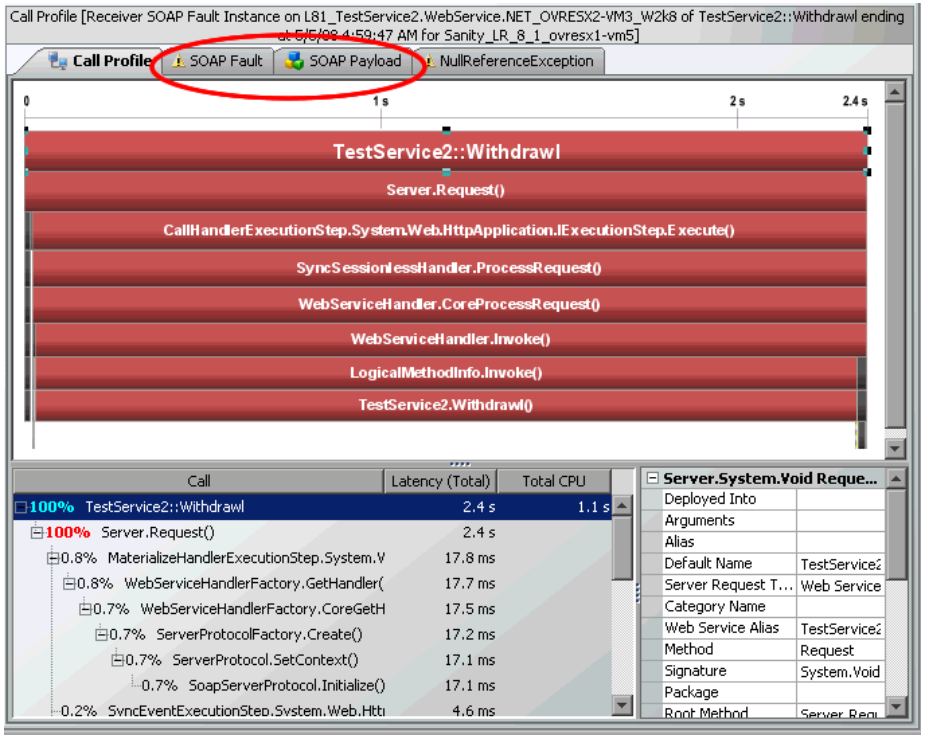
See the *HP Diagnostics Installation and Configuration Guide* for details on limiting exception tree data.

Soap Faults

If methods throw SOAP faults, these are marked in the Server Requests view with an icon.



This is an example of the type of icon used to indicate where an soap fault occurred. See “Types of SOAP Faults” on page 447 for details on the icons used to represent different types of faults. When you click the icon, a call profile opens.



You can select the **SOAP Fault** tab to get details on the fault including the stack trace. You can select the **SOAP Payload** tab to get details on the payload.

The Diagnostics SOAP message handler is required for probes to collect payload on SOAP faults. See “Payload on SOAP Faults” on page 449 for more information. See the *HP Diagnostics Installation and Configuration Guide* for details on configuring the SOAP message handler as well as configuring the probes to capture SOAP fault data.

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 Java 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 HP Software 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 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 five-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 Using 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.

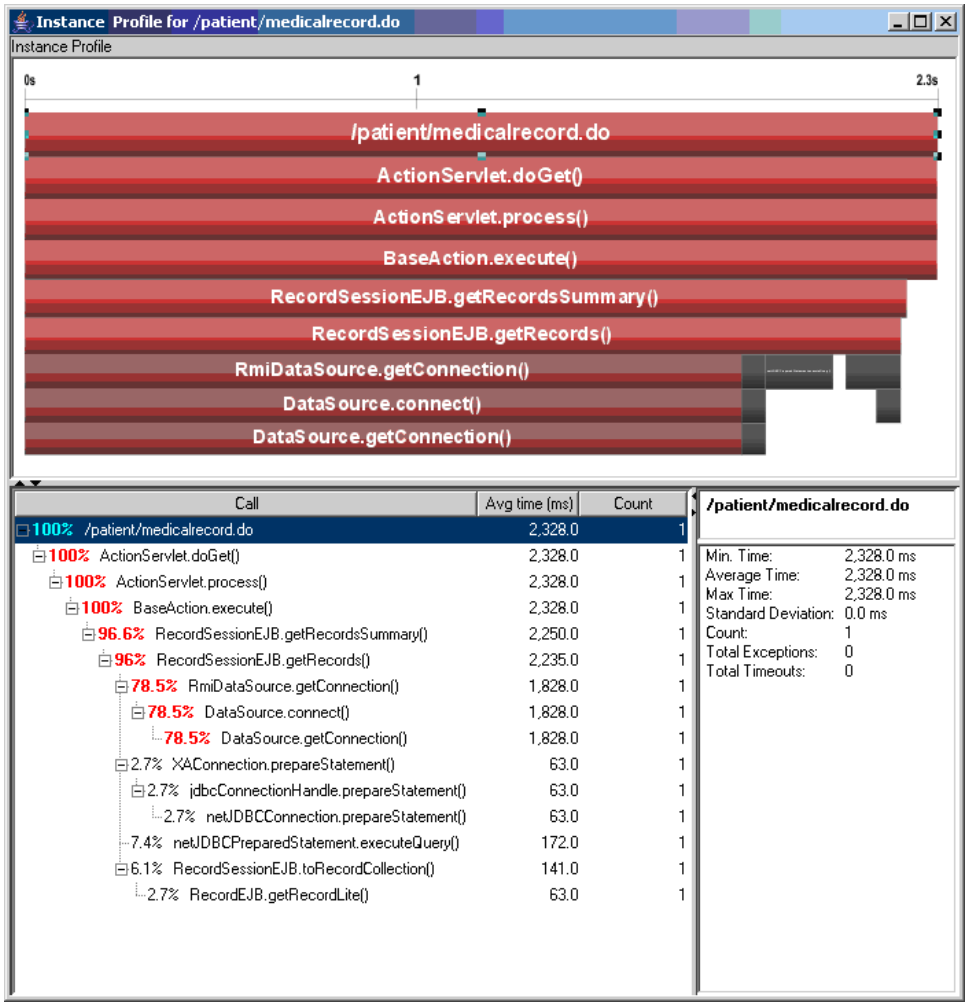
Another disadvantage is that while aggregate trees do well for general performance tuning (such as what occurs during a load test), they tend to hide information about relatively infrequent slow cases. Unfortunately, it is often these cases that cause problems for the customer.

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 Java application.

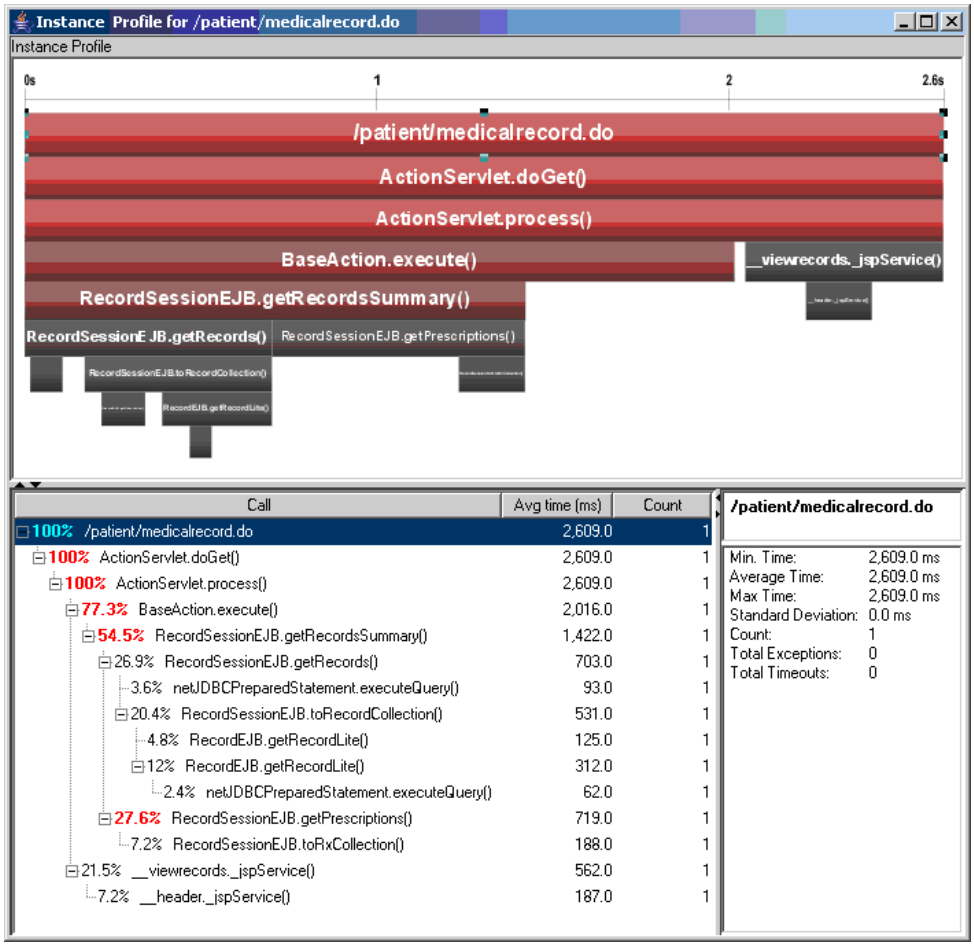
Example Instance Tree #1

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?)



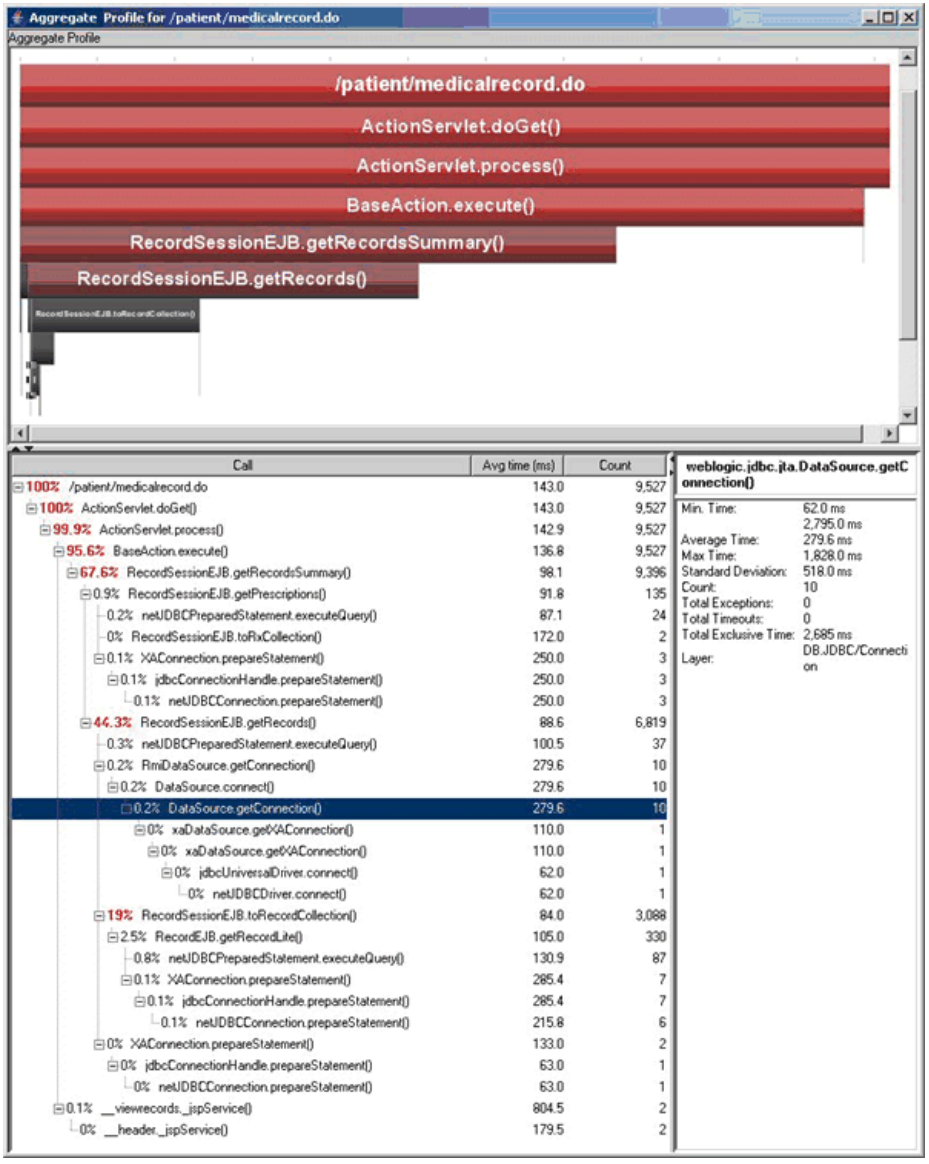
Example Instance Tree #2

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

You can create your own customized views in Diagnostics so that you can focus on the data you need to monitor specific applications and situations.

This chapter includes:

About Diagnostics Custom Views

Diagnostics lets you create your own custom views and 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 leave Diagnostics or exit back to the Application Overview screen, the view is displayed in its default configuration the next time you access it.

Customized Drill Down 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 breadcrumbs. 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 create. Custom views cannot be stored in any of the view groups installed with the product except for **My Views**.

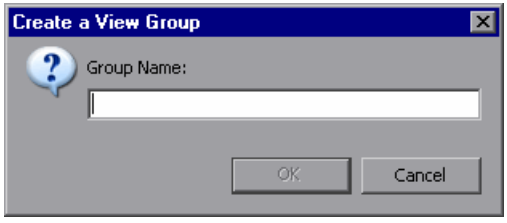
Note: Custom view groups created within the context of an application can only be viewed within that context. And these custom view groups are shared across all users of that application.

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 or Opening 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 close them so that Diagnostics will not display them in the View bar. You may open View groups that have been hidden so that they are once more available in the View bar.

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

To close 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 **Close 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.

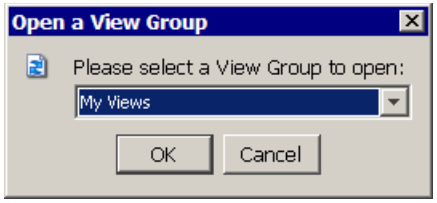
Diagnostics hides the view group so that it is no longer visible in the view group displayed in the View bar and opens the View group that preceded the view group that you hid.

To open a view group:

1. Right-click the View bar inside of any view group, and select **Open 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 Open a View Group dialog box as shown in the following image:



2 Select the view group that you want to open 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.

Creating a New Custom 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 navigate to the underlying metrics just as you can in any other view in Diagnostics.

Note: Custom views created within the context of an application can only be viewed within that context. And these custom views are shared across all users of that application.

To create a view:

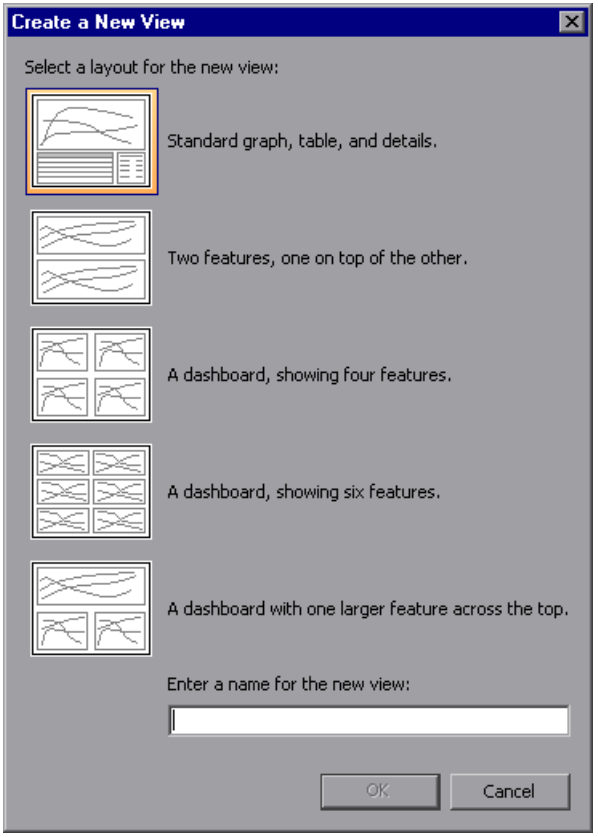
1 Select the custom view group where you want to store the new view from the view bar.

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

---

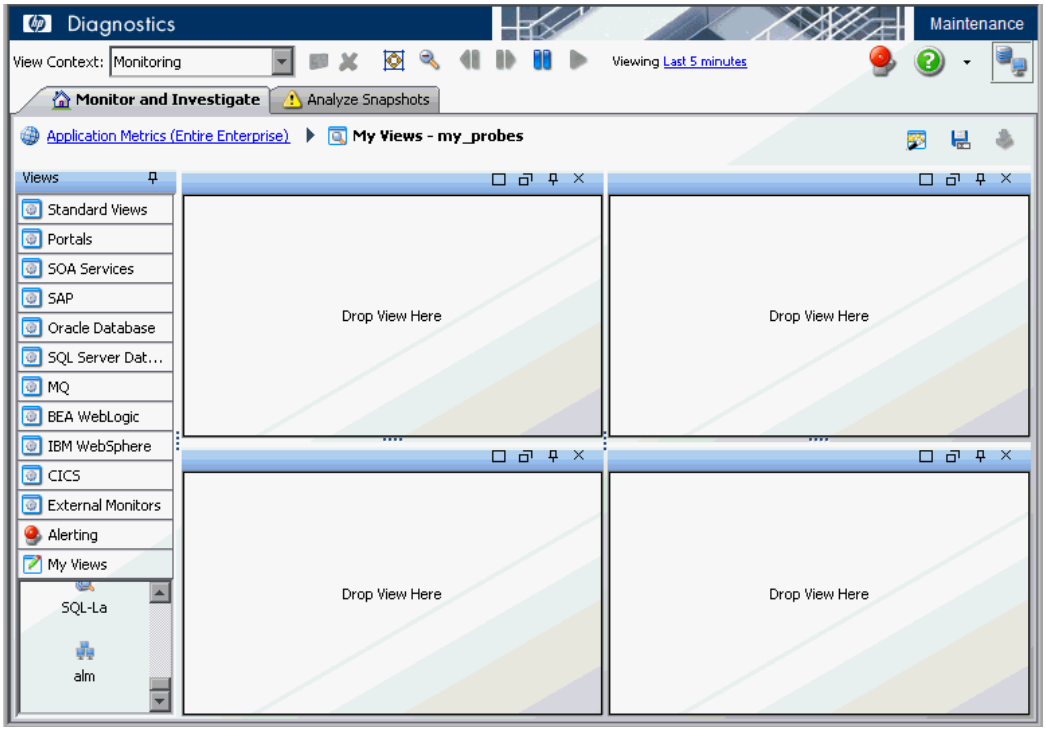
2 Right-click the View bar inside the custom view group, and select **Create a New View** from the pop-up menu.

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



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**.

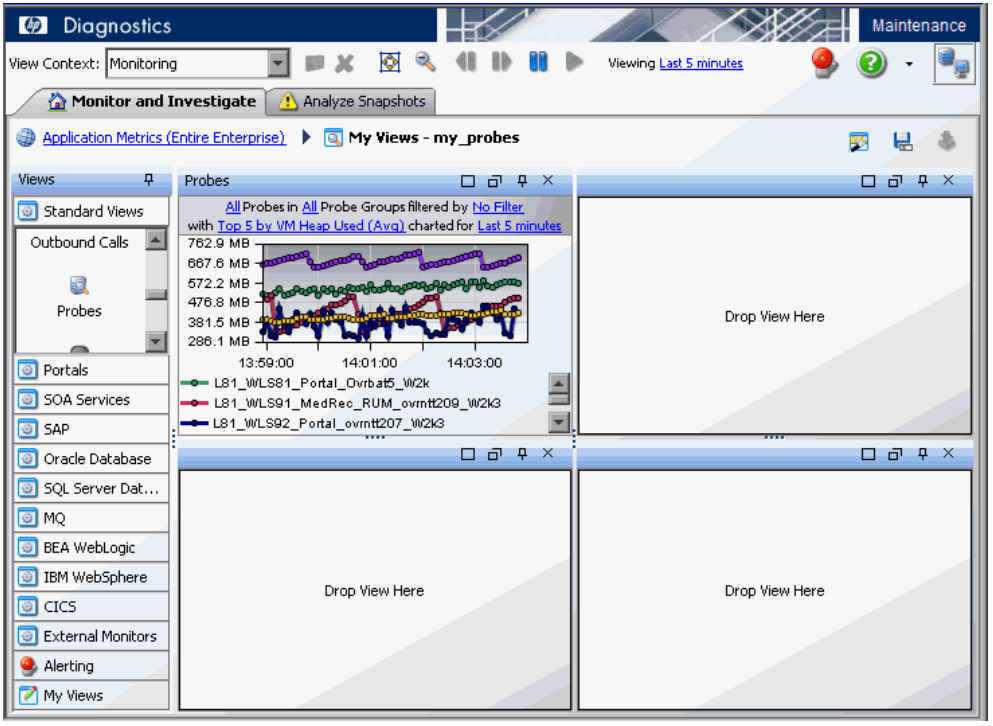
An icon for the new view is added to the view group, and the selected view layout is displayed on your screen with **Drop View Here** placeholders for each of the views that make up the new view. The example below shows the view layout for the four-feature dashboard.



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.

Saving a Customized View

As you work with a view, you may find that your customization to the view has 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 Business Availability Center or Performance Center, the customized views are saved for the 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**.

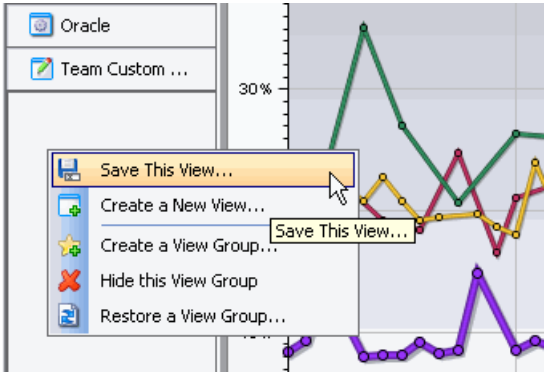
There are two ways to save a custom view once you have configured the view in the desired manner: using the pop-up menu in a custom view group and using the Save this View button in the toolbar.

To save a view in a custom view group:

1 Open the custom view group where you want to store the saved view.

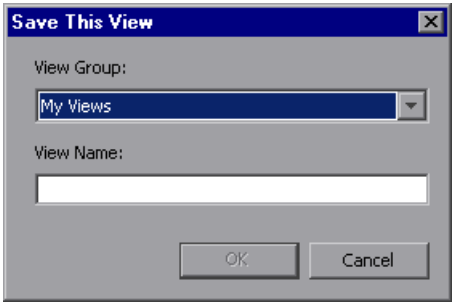
Note: Custom views can only be saved in the view group, My Views, or in a view group that you have created.

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 box.



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.

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 box.

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 two ways for you to share the custom views that you have created, with other users.

➤ When you create a custom view and you are not in the context of an application (Entire Enterprise views selected in the Applications window), 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.

➤ When you create a custom view in the context of an application, the XML file for that view is stored in **<diagnostics_server_install_dir>/storage/userdata/<customer name>/<application identifier>**.

Any user who logs into Diagnostics and has permission to view that application will share this view. Note, however, that with the new application permissions, only users with the Modify Screens permission for the application will actually have their changes saved.

This does, however, present a potential for multiple users to be viewing the same custom view and make changes. In this case, only the first user's changes will be saved. Other users (or even the same person logged in multiple times) will see an error message that their changes could not be saved, and they should save the view as a new view if they want to preserve the changes.

Upgrading Custom Views From Previous Diagnostics Versions

When you open the custom views that were created in earlier versions of Diagnostics for the first time in the latest version of Diagnostics, 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 Diagnostics data refer to the *HP Diagnostics Installation and Configuration Guide* appendix on upgrades.

For instructions on using individual custom views from a backup or from another user see “Sharing Custom Views” on page 227.

Part III

Understanding Diagnostics Standard Views

11

Server Summary View

The **Server Summary** view provides a high-level overview of your enterprise and allows you to easily check the status. This is the default view for the Entire Enterprise application when Diagnostics is run by itself or is integrated with Business Availability Center. If you selected an application, the Server Summary view gives a high-level overview of that application.

The Server Summary view shows a Status table, a graph of aggregated server requests, and a list of alert events.

This chapter includes:

Accessing the Server Summary View

You can access the Server Summary view from the View bar in the Diagnostics views.

To access the Server Summary view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

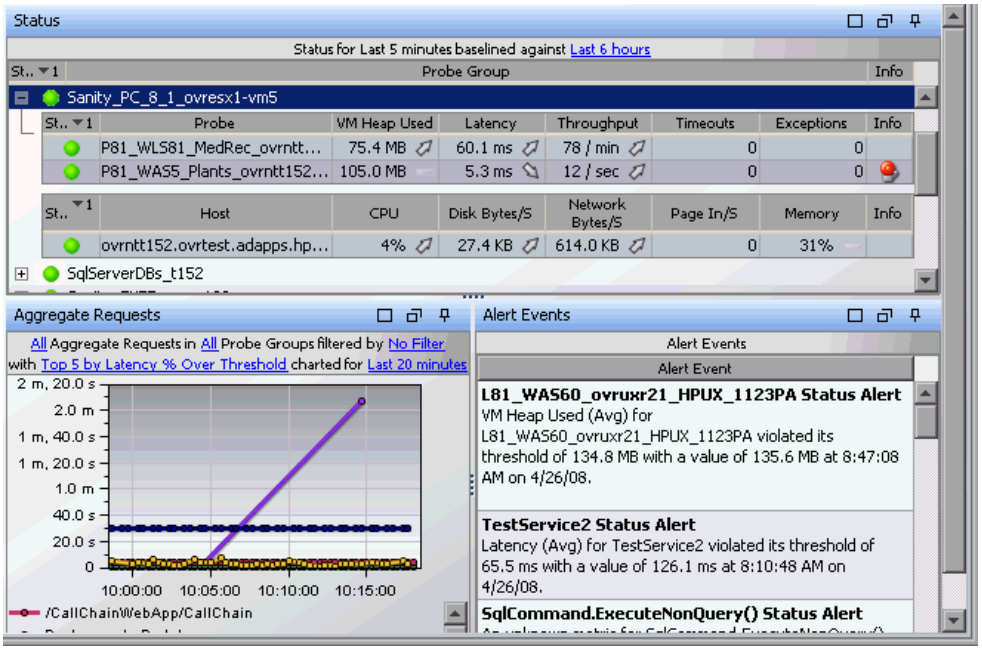
3. In the Standard Views view group, click **Server Summary**.

   Diagnostics shows the Server Summary view with the default settings.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select Entire Enterprise in the navigation pane. And then select the **application summary** link.

Description of the Server Summary View

The following figure shows an example of the Server Summary view.



Status Table

The Status table provides you with a quick indication of the status and metrics values of hosts and probes. The Status table contains the probe groups that you defined when you installed the probes. Within each probe group, there are several sub-tables.

- **Probe Table.** Lists all of the probes in the probe group.
- **Host Table.** Lists the hosts for each of the probes in the probe group.
- **Monitors Table.** If you have configured an integration with SiteScope to forward data to Diagnostics you will see the Monitors table, which lists the SiteScope monitors in the probe group.

  By default, the first column in the Status table presents the status for each probe group, probe, and host. The status reflects how the performance of the items compares with their performance thresholds.

The table also shows key metrics and attributes for each item.

You can collapse or expand a probe-group entry to show or hide the probes in the group.

The title of the status table contains the baseline control. You use this control to set the time that Diagnostics uses as the baseline performance level with which to compare the current performance level. The result of the comparison of the baseline performance to the current performance is shown by using the metric trend indicators (arrows that are displayed following the metric values in the status tables).

See Chapter 21, “Status View,” to find out how to use the information in this display.

Aggregate Requests Graph

The Aggregate Requests graph in the Server Summary view shows the trends of the aggregated server requests across your enterprise (or application). Server requests are aggregated at the probe-group level. When multiple probes monitor the same application, this aggregation enables you to see related data from multiple probes in one view.

By default, the Aggregate Requests graph displays the top five aggregated server requests with the highest latency percent over threshold during the previous five minutes. By default, the data is based on all aggregated server requests for all probe groups. You can filter the data by aggregated server requests, or by an individual probe group.

The Aggregate Requests graph displays latency, using a trend line. The x-axis of the graph shows actual chronological time. The y-axis of the graph shows latency in seconds.

See “Aggregate Requests View” on page 312, to find out how to use the information in this display.

Alert Events Table

The Alert Events table lists recent alerts that have been triggered. By default, the table of events in this view is sorted with the most recent alert events at the top. The list can contain a maximum of 20 events. For a complete list go to the Alerting view.

See “Reviewing Alert Notification Events” on page 175 to find out how to use the information in this display.

Customizing the Server Summary View

You can use the standard Diagnostics view controls to adjust the amount and type of data that Diagnostics displays in the Server Summary view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 4, “Working with Diagnostics Data Displays.”

The table headers in the Status table contain controls that enable 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.

Interpreting the Server Summary View

Using the information displayed in the Server Summary view status table, you can get an immediate understanding of the performance of your application from the probe groups, probes, and host being monitored.

Using the information displayed in the Server Requests (Aggregated) graph, you can get an immediate understanding of the requests that may be experiencing performance problems.

If the metrics displayed in the Server Summary view raise 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 a probe group in the status table, you can drill down to aggregated server requests for that probe group.

Checking Status Indicators

The status indicator shows how the listed entity or any of its child entities is performing relative to the thresholds that you set for their metrics. A tooltip for each indicator gives a description of the current status. See “Investigating the Status of Selected Entities” on page 332 for details on status indicators.

Displaying Probe Group, Probe, and Host Details

You can view details for a probe group, probe, or host listed in the Status table by holding your mouse pointer over that entry until the tooltip appears.

The **Probe Group Details** tooltip displays the probe group name.

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.** Java, .NET, Oracle, SAP, MQ, SQL Server.

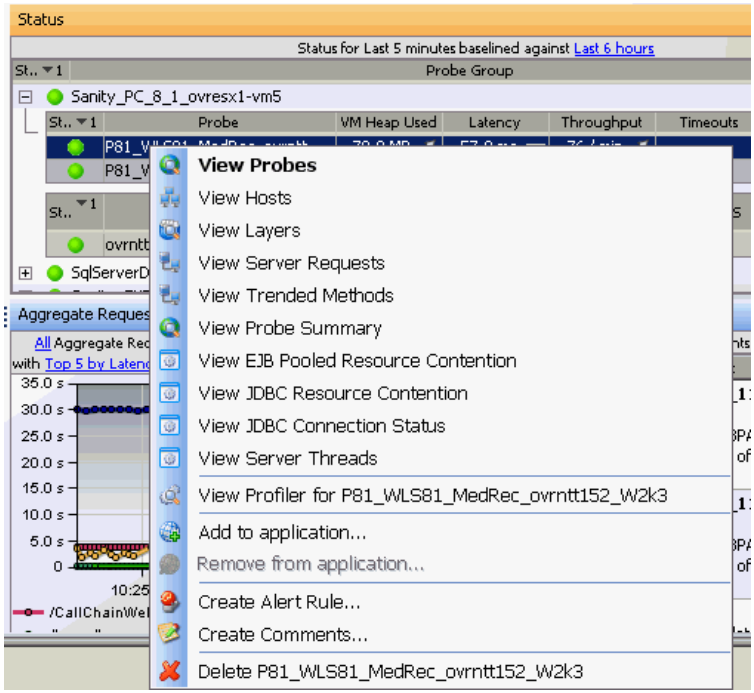
The **Host Details** tooltip displays the fully qualified host name.

Drilling Down and Taking Action from the Status Table

From the Status table, you can drill down into the details for the probe groups, probes and hosts that are behind the status that you see in the view. And you can take actions that are relevant to the entity selected.

Right-clicking a row in the Status table displays a pop-up menu of suggested actions you can take to further analyze a performance problem and diagnose the cause.

When you select a row in the table, Diagnostics updates the right-click menu items so that they contain only **relevant** navigations and actions that are appropriate for the selected entity. An example of the relevant navigations and actions for a probe is shown below:



In general, the menu items can be described as follows:

- Some navigation links (such as View Probes, View Server Requests, View Load) provide a drill down to more detailed information in a standard view such as the Probes view.
- Some navigation links (such as View JDBC Connection Status) are specific to the type of probe selected and display more detailed information in a specialized view group such as the IBM WebSphere view group.
- The View Profiler... navigation link opens the Java profiler or the .NET profiler which you can use to analyze method latency and monitor memory usage in your application.
- Common actions are also available in the pop-up menu. Some typical actions you might want to take when analyzing a performance problem include: Create Alert Rule, Create Comments, Add to Application. See “Common Tasks” on page 116 for more information on these actions.

Navigation links for a probe group, probe or host are filtered by the selected probe group, probe or host and display data for the previous five minutes, regardless of the global time range.

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

When you double-click a row in the host table, Diagnostics displays the Hosts view for the selected host.

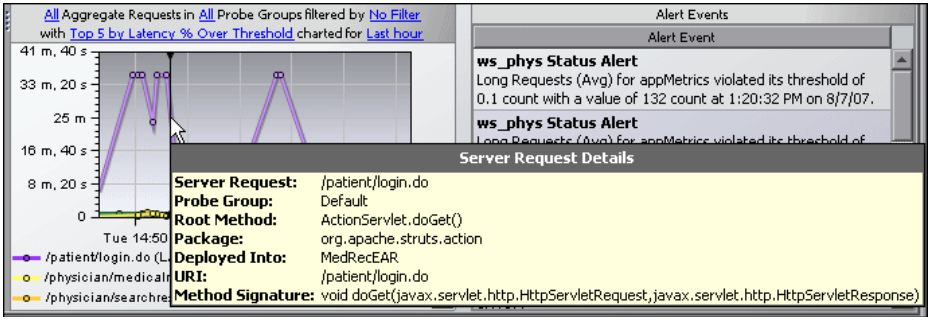
Drilling Down from the Aggregate Requests Graph

You can drill down to the Aggregate Requests view by double-clicking the concise Aggregate Requests graph or its legend. The Aggregate Requests view gives the full-featured view for the graph, rather than the summary. See “Aggregate Requests View” on page 312 for details.

Displaying Server Request Details from the Charted Metrics

You can get additional details about a server request whose metrics are charted in the graph by viewing the tooltips that Diagnostics displays in the Aggregate Requests graph.

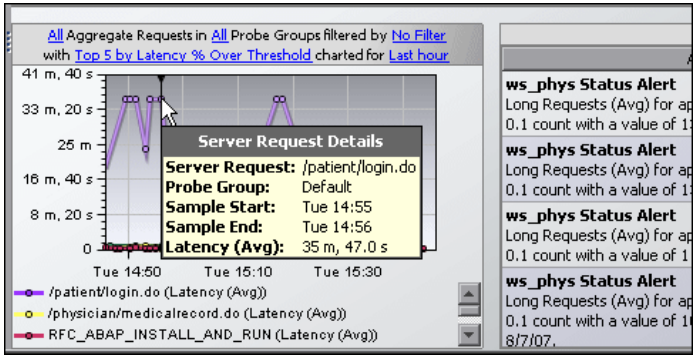
When you hold your mouse pointer over a point on a charted trend line that is between two of the round nodes on the trend line, Diagnostics displays a **Server Request Details** tooltip, as shown in the following figure.



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

- **Server Request.** The name of the selected server request.
- **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 method may be a portion of a URL. In the case of an RMI call, it may be a class and method representing the name of the server request itself.
- **Package.** The name of the package that contains the class from which the method was called.
- **Method Signature.** The signature of the root method.

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 **Server Request Details** tooltip, as shown in the following figure.



The additional information shows the information that was used to plot the selected node on the trend line.

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

- **Server Request.** The name of the selected server request.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- **Sample Start.** The start time that the selected data point sample began.

- **Sample End.** The end time that the selected data point sample ended.
- **Latency (Avg).** By default, the average CPU utilization for the data point (the y-axis value).

Drilling Down from the List of Alert Events

You can drill down from an alert event listed in the alert events table to the relevant detailed view by double-clicking the alert event. If a threshold on a probe is violated, the drill down is to the Probes view.

For information about alert notification rules, see Chapter 7, "Working with Alerts."

12

Application Explorer View

The Application Explorer gives you a topological view of your application, and a set of tabs showing data for various aspects of the selected entity's performance. As you select different entities in the topology, the data is updated to reflect the current selection.

You can use the Application Explorer as the starting point to determine how your application is performing. For example, if some aspect of an application is taking longer to complete than expected, you can start with the Application Explorer and examine each entity in the application to locate the performance issues. If you find that a server request has unusually high latency then you can drill into more details. for that request

This chapter includes:

Accessing the Application Explorer View

You can access the Application Explorer view from the View bar in the Diagnostics views.

To access the Application Explorer view:

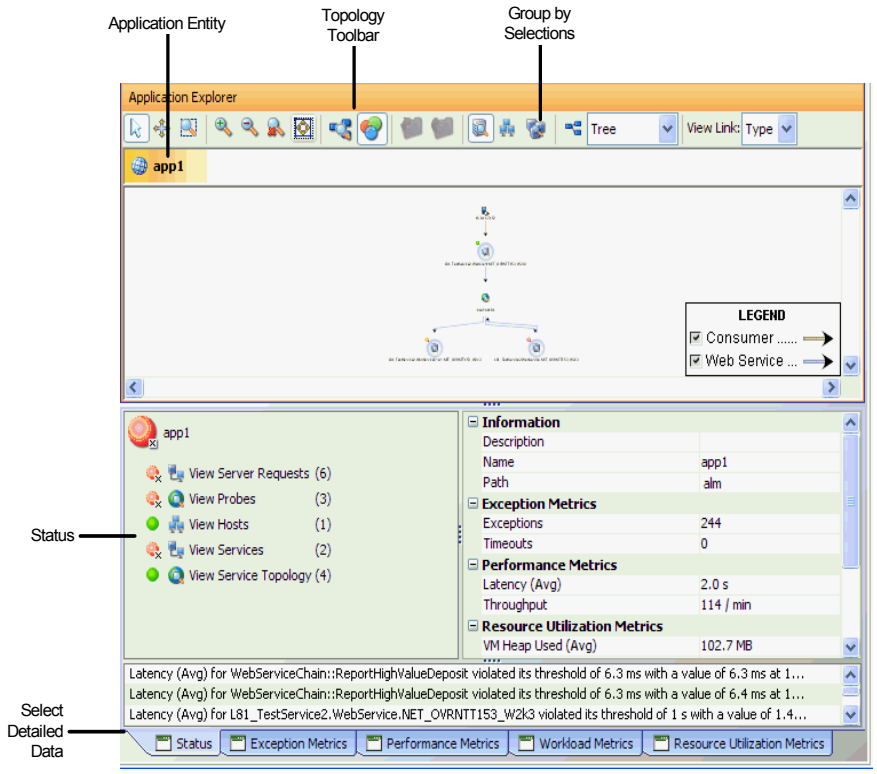
1. In the navigation pane of the Diagnostics Applications window, double-click an application name from the Applications groups. The Diagnostics views open with the Application Explorer displayed. This is the default view for an application.
2. Alternately, from the Applications window you can select an application and then select **Open application explorer** and this will display the Application Explorer.
3. Or, from the View bar, click **Standard Views** to open the Standard Views view group. In the Standard Views view group, click **Application Explorer**.

Note: The Application Explorer view is not available if you select the Entire Enterprise.

You can navigate to all other relevant views from within the Application Explorer or you can select another view from the View Bar.

Application Explorer Entry Screen

The following image is an example of the entry screen in the Application Explorer.



When you enter the Application Explorer, the application entity is selected by default in the topology (the application is shown at the top, above the topology). The entity selected in the topology determines what is displayed in the lower panes.

The topology provides a visual representation of your application. The types of entities that may be selected in the topology are as follows:

- Application

- Probe group
- Host
- Probe (Java. .NET, SAP NetWeaver-ABAP collector)
- Collector (Oracle, SQL Server, MQ)
- Connections

Other entities (like consumer) are shown in the topology but cannot be selected.

You can also see the status for an entity in the topology, represented by a green, yellow, red or grey icon. See "About the Graph Entity Table" on page 101 for an explanation of the status icons.

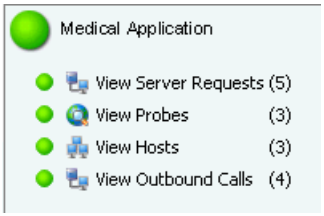
You can use the topology toolbar to control the appearance of the topology by scrolling and zooming, and by toggling the appearance. See Chapter 6, "Working with the Topology Views" for details.

You can select to group data in the topology by hosts and by probe groups. Use the Group Probes by Host and Group Probes by Probe Group buttons on the toolbar.

You can use the tabs at the bottom of the display to access performance information for the entity selected. The tabs available will vary slightly based on the entity you select in the topology.

Status Tab

When you select the Status tab, the pane on the lower left shows the status of the entity selected in the topology. And under this you'll see the entities found and the relevant navigations for the selected entity. So for example you may see the following when an application entity is selected:



This indicates that for the Medical Application you should see three probes, three hosts, five server requests and four outbound calls.

See "Investigating the Status of Selected Entities" on page 332 for more information on the status indicators and how status is calculated. See "Setting Metric Thresholds" on page 131 for details on metric thresholds.

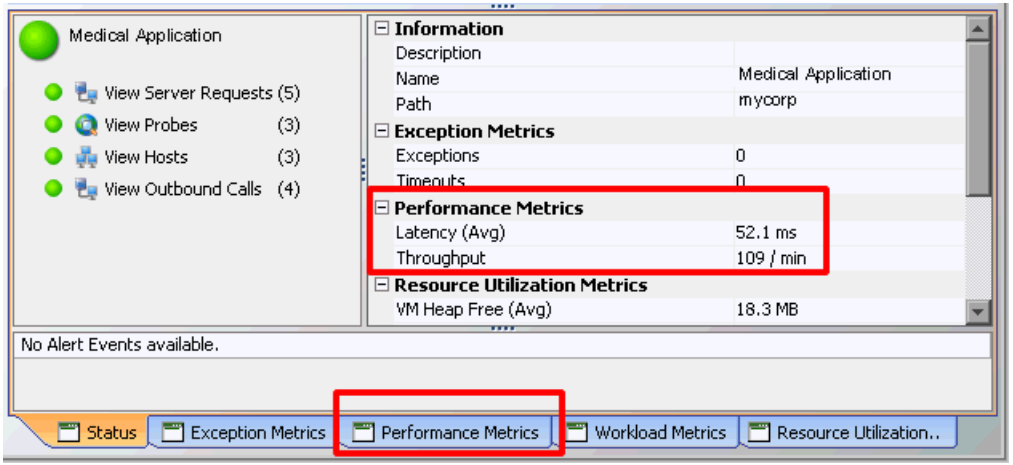
Note: When you have selected an application, the time period shown is always five minutes. For other entities you can select a time period.

You can select any of the links in the Status and Navigations pane and navigate to a detail view.

The pane on the lower right shows summary information for the entity selected in the topology. This is designed to give you a high level view of the performance metrics for the entity selected. Each of the summary groups is associated with a different tab at the bottom of the screen. Threshold violations for a metric (such as Latency (Avg)) are shown in red or yellow in the summary.

The metrics shown will vary depending on what type of entity you've selected in the topology. See "Metrics Details" on page 251 for details.

For example the Performance Metrics summary shows Latency and Throughput for the selected application. Selecting the Performance Metrics tab at the bottom of the display gives you performance metrics trend graphs for the entity selected.



Alerts for the selected entity are shown at the bottom of the display. Only the last three alerts for the entity selected are shown. The alerts are not cumulative for all related entity, just for the selected entity.

Exception Metrics Tab

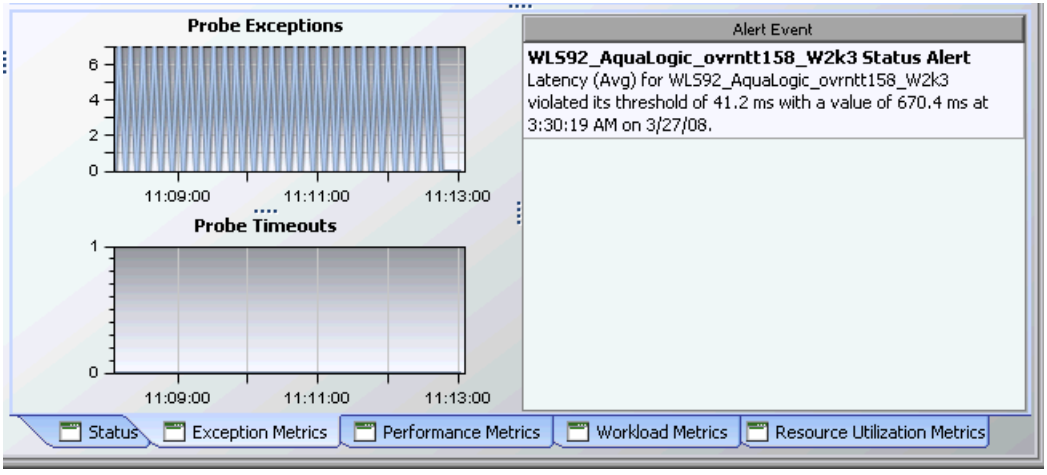
To get more detailed information on exceptions you can select the Exception Metrics tab.

The Exceptions Metrics tab is available when you select the following types of entities: applications, probes (Java and .NET), probe (ABAP), probe group and probe connection. No Exceptions Metrics tab is available when a host, or collector entity (Oracle, SQL Server, MQ) is selected in the topology.

The Exception Metrics tab displays the following data.

- A trend graph of exceptions for the entity
- A trend graph of timeouts for the entity
- A trend of alerts, if any, for the entity

For example, for a probe entity the following is displayed:



Double-clicking on the graphs allows you to drill down to more detail.

If there are no exceptions, the graphs on this tab will show a zero trend line.

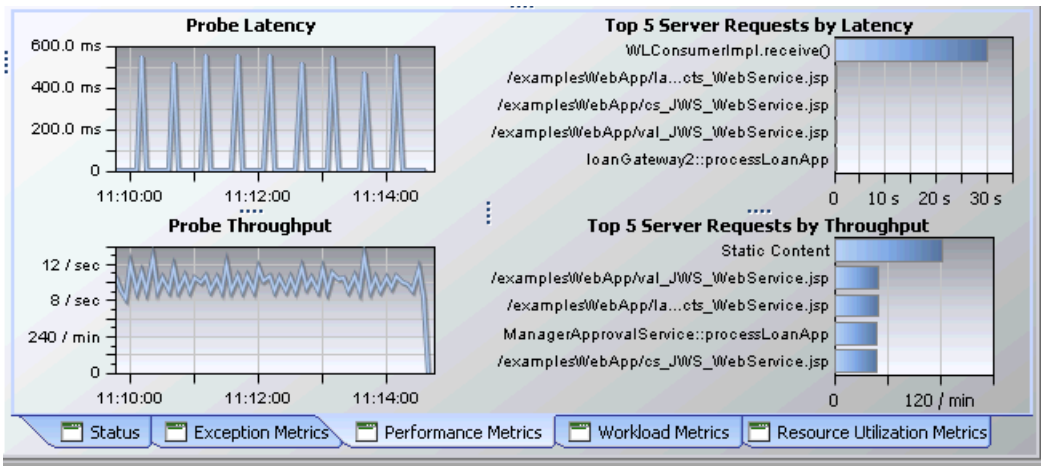
Performance Metrics Tab

To get more detailed information on performance you can select the Performance Metrics tab. The lower pane then displays additional data.

The specific data varies depending on the type of entity selected. See “Metrics Details” on page 251.

For example, for a probe entity the following is displayed:

- A trend graph of latency for the entity
- A trend graph of throughput for the entity
- A bar chart of top 5 requests by latency
- A bar chart of top 5 requests by throughput



If you select a connection in the Application Explorer topology, the bar graphs show Top 5 Calls by <metric> allowing display of both outbound and inbound calls.

Double-clicking on a graph or chart allows you to drill down to more detail.

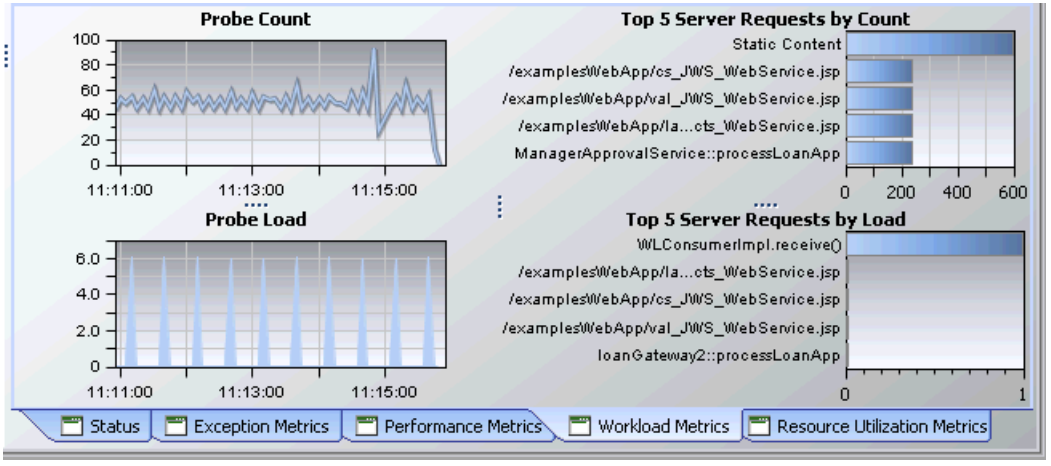
Workload Metrics Tab

To get more detailed information on workload you can select the Workload Metrics tab. The lower pane then displays additional data.

The specific data varies depending on the type of entity selected. See “Metrics Details” on page 251.

For example, for a probe entity the following is displayed:

- A trend graph of the count for the entity
- A trend graph of the load for the entity
- A bar chart of the top 5 server requests by count
- A bar chart of the top 5 server requests by load



Double-clicking on a graph or chart allows you to drill down to more detail.

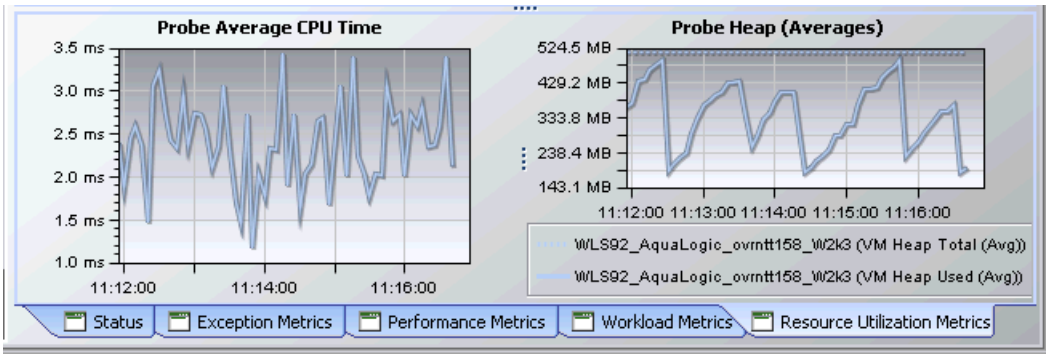
Resource Utilization Metrics Tab

To get more detailed information on resource utilization you can select the Resource Utilization Metrics tab. The lower pane then displays additional data.

The specific data varies depending on the type of entity selected. See “Metrics Details” on page 251.

For example, for a probe entity the following is displayed:

- A trend graph of the average CPU time
- A trend graph of the Heap averages



Double-clicking on the graphs allows you to drill down to more detail.

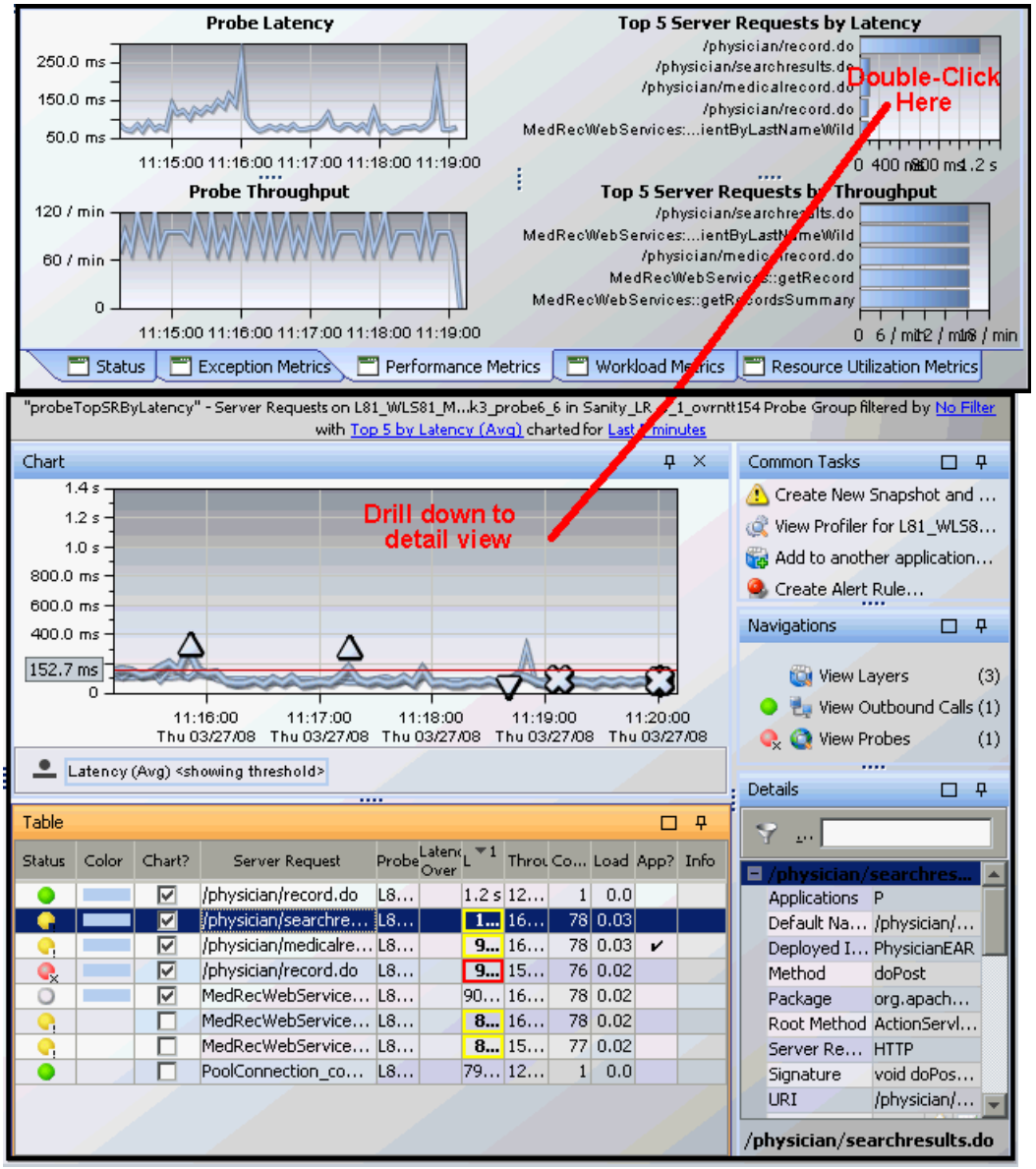
Metrics Details

When you double-click on any of the graphs in the Performance Metrics, Workload Metrics or Resource Utilization tabs, a detail view is displayed filtered on the entity you selected in the topology.

So, for example, if you double-click on the Top 5 Server Requests by Latency for a probe then a screen is displayed showing server request details for the selected probe, with the top 5 server requests charted.

For a drill down from probe to server request details, you will see all server requests active in the probe. They may or may not be assigned to the application you had selected. The App? column in the drilled to table will be checked to indicate if the server request is part of the application.

See the example below.


Probe Latency
250.0 ms
150.0 ms
50.0 ms
11:15:00 11:16:00 11:17:00 11:18:00 11:19:00
Probe Throughput
120 / min
60 / min
0
11:15:00 11:16:00 11:17:00 11:18:00 11:19:00
Top 5 Server Requests by Latency
/physician/record.do
/physician/searchresults.do
/physician/medicalrecord.do
/physician/record.do
MedRecWebServices:...ientByLastNameWild
Double-Click Here
0 400 ms 800 ms 1.2 s
Top 5 Server Requests by Throughput
/physician/searchresults.do
MedRecWebServices:...ientByLastNameWild
/physician/medicalrecord.do
MedRecWebServices::getRecord
MedRecWebServices::getRecordsSummary
0 6 / min 12 / min 18 / min
Status
Exception Metrics
Performance Metrics
Workload Metrics
Resource Utilization Metrics
"probeTopSRByLatency" - Server Requests on L81_WLS81_M...k3_probe6_6 in Sanity_LR
with Top 5 by Latency (Avg) charted for Last 5 minutes
Chart
Drill down to detail view
1.4 s
1.2 s
1.0 s
800.0 ms
600.0 ms
400.0 ms
152.7 ms
0
11:16:00 Thu 03/27/08
11:17:00 Thu 03/27/08
11:18:00 Thu 03/27/08
11:19:00 Thu 03/27/08
11:20:00 Thu 03/27/08
Latency (Avg) <showing threshold>
Common Tasks
Create New Snapshot and ...
View Profiler for L81_WLS8...
Add to another application...
Create Alert Rule...
Navigations
View Layers (3)
View Outbound Calls (1)
View Probes (1)
Details
Table
Status
Color
Chart?
Server Request
Probe
Latency Over
Throu
Co...
Load
App?
Info
/physician/record.do
/physician/searchre...
/physician/medicalre...
/physician/record.do
MedRecWebService...
MedRecWebService...
MedRecWebService...
PoolConnection_co...
/physician/searchres...
Applications
Default Na...
Deployed I...
Method
Package
Root Method
Server Re...
Signature
URI
/physician/searchresults.do

The following table shows the metrics available for each entity type in the Performance Metrics tab, Workload Metrics tab and Resource Utilization tab.

Entity Type	Performance Metrics	Workload Metrics	Resource Utilization Metrics
Application entity:	Throughput Top N Server Requests by Throughput Latency by layers Top N Server Requests by Latency	Count Top N Server Requests by Count Load Top N Server Requests by Load	CPU (Avg) Heap Total/Used Disk bytes/sec Network bytes/sec
Probe (Java and .NET) entity:	Throughput Top N Server Requests by Throughput Latency Top N Server Requests by Latency	Count Top N Server Requests by Count Load Top N Server Requests by Load	CPU (Avg) Heap Total/Used
Probe (ABAP) entity:	Throughput Top N Server Requests by Throughput Latency Top N Server Requests by Latency	Count Top N Server Requests by Count Load Top N Server Requests by Load	CPU (Avg) Extended Memory Private Memory
Host entity:	N/A	N/A	CPU (Avg) Memory, Virtual Memory Disk bytes/sec Network bytes/sec

Entity Type	Performance Metrics	Workload Metrics	Resource Utilization Metrics
Probe Group entity:	Throughput Top N Aggregated Server Requests by Throughput Latency Top N Aggregated Server Requests by Latency	Count Top N Aggregated Server Requests by Count Load Top N Aggregated Server Requests by Load	N/A
Probe Connection entity (when connection has an outbound call):	Throughput Top N Outbound Calls by Throughput Latency Top N Outbound Call by Latency	Count Top N Outbound Calls by Count Load Top N Outbound Calls by Load	N/A
Probe connection entity (when connection is only an inbound probe call):	Throughput Top N Server Requests by Throughput Latency Top N Server Requests by Latency	Count Top N Server Requests by Count Load Top N Server Requests by Load	N/A
Oracle collector entity:	User Transaction Per Sec Executions Per Sec	Load by layers Load by wait time	DB wait time ratio DB CPU time ratio CPU usage per sec CPU usage per txn

Entity Type	Performance Metrics	Workload Metrics	Resource Utilization Metrics
SQL Server collector entity:	Database:Transactions/sec	Load by wait time	CPU Busy Time (Avg) IO Busy Time (Avg) Idle Time (Avg)
MQ collector entity:	N/A	Messages Transferred/sec (Avg) Top N Channels by Messages Transferred/sec Depth (Avg) Top N Channels by Depth	N/A

13

Application Metrics View

The **Application Metrics** view contains performance metrics for the applications you are monitoring. By default, the Application Metrics graph presents trend lines that represent the CPU utilization for each application.

This chapter includes:

Accessing the Application Metrics View

You can access the Application Metrics view from the View bar in the Diagnostics views.

To access the Application Metrics view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

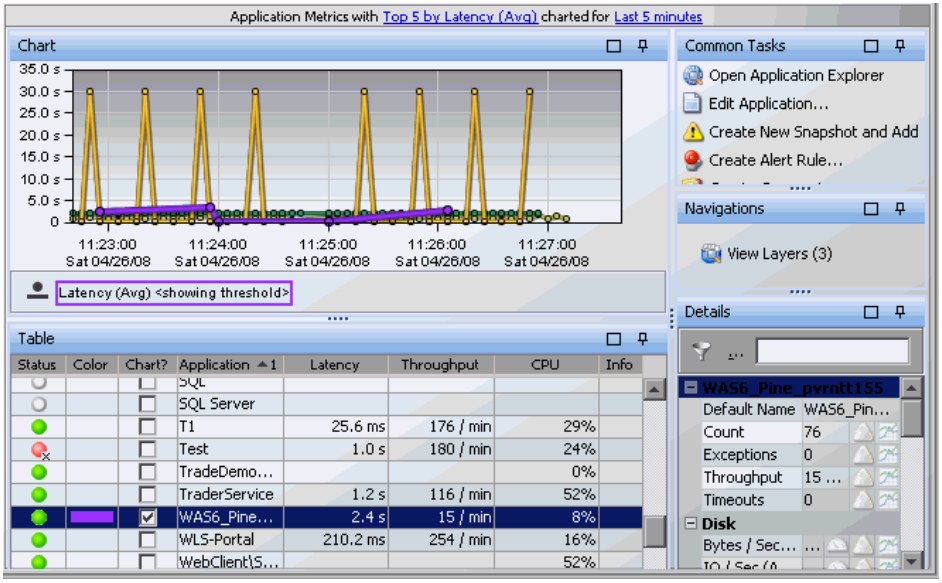
3. In the Standard Views view group, click **Application Metrics**.

   Diagnostics displays the Application Metrics view with the default settings where the five applications that have the highest CPU utilization. The time period depends on the mode of integration, as described on the previous page.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **application metrics** link.

Description of the Application Metrics View

The following image is an example of the Application Metrics view:



The Application Metrics view has the format of a detail layout view. For information about the layout and controls in the detail layout views, see Chapter 4, “Working with Diagnostics Data Displays.”

Graph

The Application Metrics View graph charts the CPU utilization metrics for the five applications that have the highest utilization during the previous hour.

Diagnostics displays the CPU utilization for each application, 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 applications that pertain to the context shown in the breadcrumbs in the graph entity table of the Application Metrics view. The metrics for each application that are reported in the table are aggregated and reported based on the time period specified in the **Time Range** view filter.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The Details pane in the Application Metrics view lists the system metrics for the application in the selected row of the graph entity table. For information on configuring system metrics, see the *HP Diagnostics Installation and Configuration Guide*.

Customizing the Application Metrics View

You can use the standard Diagnostics view controls to adjust the amount and type of data that Diagnostics displays in the Application Metrics view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 4, "Working with Diagnostics Data Displays."

Interpreting the Application Metrics View

When you are in the Entire Enterprise, the Application Metrics view gives you the ability to look at the resources (such as memory/heap, CPU, threads, I/O, network and VM metrics) used by the entire application in a rolled up fashion. If you find problems in a particular application, you can open that application and drill down deeper without having to deal with other applications' components and resources.

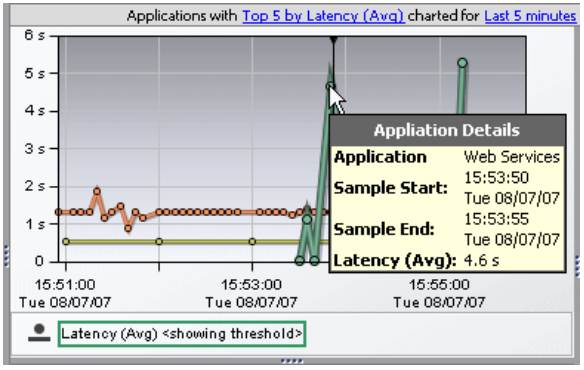
Displaying Application Details from the Charted Metrics

You can get additional details about an application 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 an **Application Details** tooltip.

  The **Application Details** tooltip contains the name and path of the application 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 **Application 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 **Application Details** tooltip displays the following information:

- **Application.** The name of the application whose performance metric is represented by the selected trend line.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** The end time for the aggregation period represented by the selected data point.
- **Latency (Avg).** By default, the average latency 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.

Editing an Application from the Application Metrics View

You can edit an application in the Application Metrics View by:

- right-clicking the application's row in the graph entity table, and selecting **Edit Application** from the menu.
- clicking **Edit Application** in the Common Tasks Pane.

Diagnostics displays the Edit application dialog box. See "Creating Application Entities in Diagnostics" on page 47 for details.

Important: You will only be able to edit an application if the user you are logged in as, has permissions for editing.

Viewing Application Layers

You can go to the application Layers view by:

- double-clicking the row for the application in the graph entity table.
- double-clicking a trend line for the application.
- clicking **View Layers** in the Navigations Pane.

In the Layers view you can see the latency contribution from the different layers in the application.

14

Dependent Services View

The **Dependent Services** view contains performance metrics for dependent services (locations and types). By default, the Dependent Services view graph presents trend lines that represent the Average Latency for each dependent service.

This chapter includes:

Accessing the Dependent Services View

You can access the Dependent Services view from the View bar in the Diagnostics views.

To access the Dependent Services view:

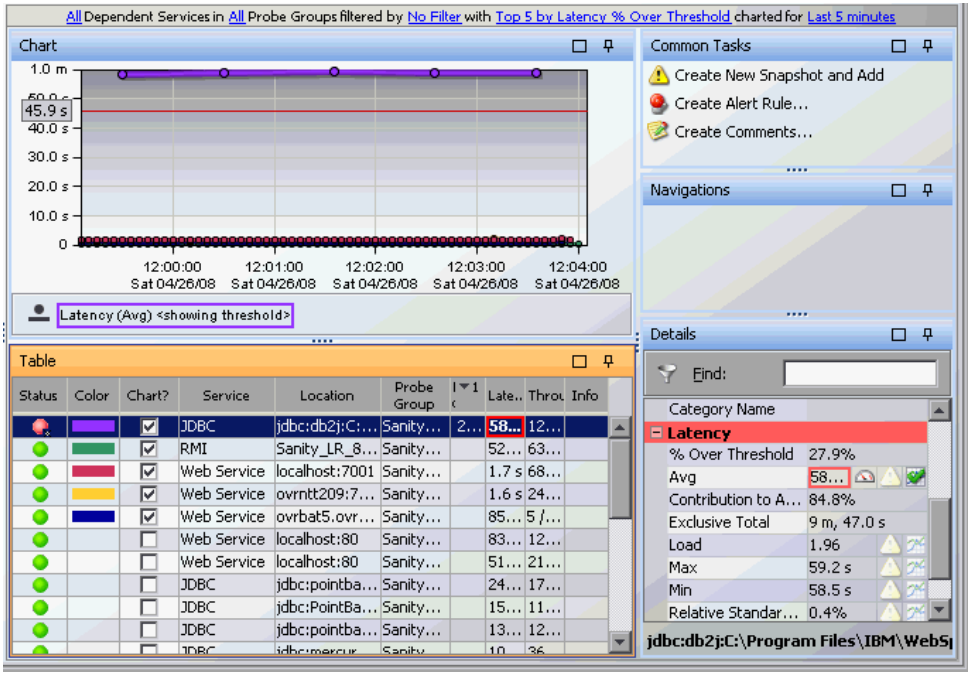
1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

3. In the Standard Views view group, click **Dependent Services**.

   Diagnostics displays the Dependent Services view with the default settings, showing the top five services that have the highest latency. The time period depends on the mode of integration, as described in "Description of the Dependent Services View" on page 267.

Description of the Dependent Services View

The following figure shows an example of the Dependent Services view.



The Dependent Services view has the format of a detail layout view. For information about the layout and controls in the detail layout views, see Chapter 4, “Working with Diagnostics Data Displays.”

Graph

The graph in the Dependent Services view shows dependent services at the probe group level.

By default, the graph displays the top five dependent services with the highest latency percent over threshold during the previous five minutes. By default, the data is based on all dependent services for all probe groups. You can filter the data for an individual dependent service or an individual probe group.

Diagnostics displays the Average Latency, using a trend line. The x-axis of the graph shows the actual chronological time. The y-axis of the graph shows the latency.

Graph Entity Table

Diagnostics lists the services (service type and location) related to the context shown in the bread crumbs in the graph of the Dependent Services view. The metrics for each service that is reported in the table are aggregated and reported, based on the time period specified in the **Time Range** view filter.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The Details pane in the Dependent Services view lists the metrics for the service in the selected row of the graph entity table.

Customizing the Dependent Services View

You can use the standard Diagnostics view controls to adjust the amount and type of data that Diagnostics displays in the Dependent Services view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 4, "Working with Diagnostics Data Displays."

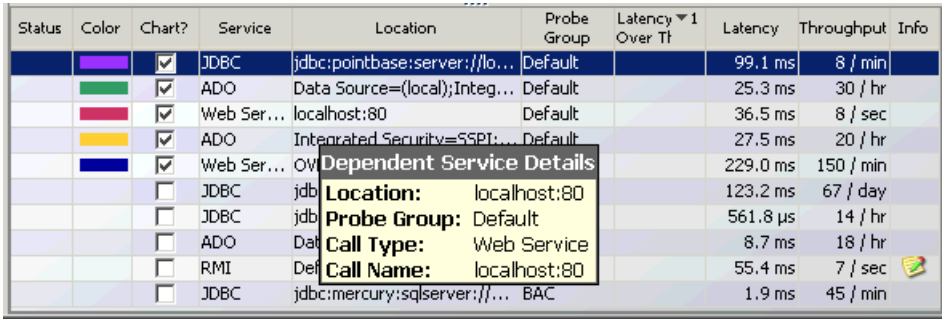
Interpreting the Dependent Services View

The Dependent Services view displays latency broken down by services that are utilized by a service request. For example, if a number of server requests use 'JDBC' and 'WebServices', by clicking on one of these services you can see how the time was used in each of the server requests that used this service. This is a reverse breakdown of time, normally you break a server request down to see where the time is spent. In this case you know some time was spent in certain services and you are interested in knowing which server requests used this service and are dependent on it.

If the metrics displayed for a service type or location raise any concerns, you can drill down to find more information from a view that depicts performance metrics at a lower level. For example, from the Dependent Services view, you can drill down to the Service Calls view showing individual probe group outbound calls.

Displaying Dependent Services Details

You can view details for a service or location listed in the graph entity table by holding your mouse pointer over that entry until the **Dependent Service Details** tooltip is displayed, as shown in the following figure.



The **Dependent Service Details** tooltip displays the following information:

- **Location.** The dependent service location.
- **Probe Group.** The name of the probe group to which the probe was assigned when it was installed.
- **Call Type.** The type of outbound call.
- **Call Name.** The method call into the dependent service.

You can also get details about dependent services whose metrics are charted in the graph by viewing the **Dependent Service Details** tooltips that Diagnostics displays for each charted trend line.

Drilling Down from the Dependent Services View

You can drill down from a service in the Dependent Services view by doing the following:

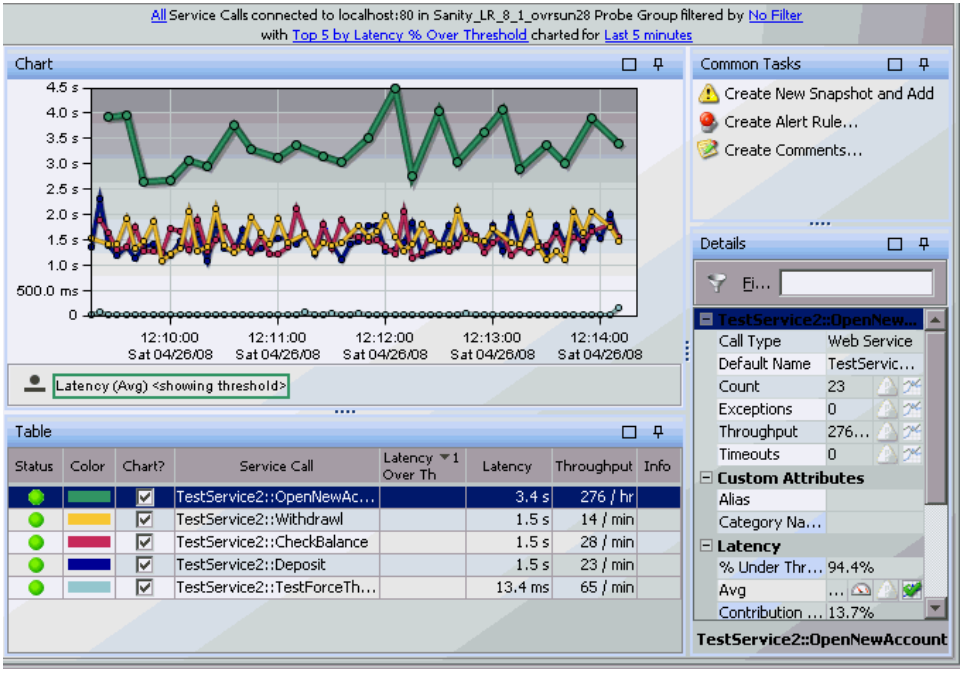
- Double-clicking the row for the service or location in the graph entity table.
- Right-clicking the row for the service or location in the graph entity table, and then selecting **View Service Calls** from the menu.
- Double-clicking a trend line for the service.

- Right-clicking a trend line for the host, and then selecting **View Service Calls** from the menu.
- Clicking **View Service Calls** in the Navigations Pane.

When you drill down from a service or location, Diagnostics displays the Service Calls view.

Service Calls View

The Service Calls view shows the calls that were executed for a selected dependent service during the specified time range.



Graph

The graph in the Service Calls view shows trend lines for service calls made to the dependent service you selected.

By default, the graph displays the top five service calls for this dependent service with the highest latency percent over the threshold during the previous five minutes.

Diagnostics displays the Average Latency, using a trend line. The x-axis of the graph shows actual chronological time. The y-axis of the graph shows the latency.

Graph Entity Table

Diagnostics lists the service calls that are related to the dependent service you selected. (This context is shown in the breadcrumb at the top of the graph.) The metrics for each service call that is 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 Service Calls view lists the system metrics for the service call in the selected row of the graph entity table. For information about configuring system metrics, refer to the *HP Diagnostics Installation and Configuration Guide*.

15

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.

This chapter includes:

Accessing the Hosts View

You can access the Hosts view from the View bar in the Diagnostics views.

To access the Hosts view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.
3.  In the Standard Views view group, click **Hosts**.

   Diagnostics displays the Hosts view with the default settings showing the five hosts that have the highest CPU utilization. The time period depends on the mode of integration, as described in "Description of the Hosts View" on page 275.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **Hosts** link.

You can drill down to this view from the Host table in the Status view. Right-click a probe group in the Host table and select **View Hosts**.

Description of the Hosts View

The following image is an example of the Hosts view:

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 4, “Working with Diagnostics Data Displays.”

Graph

By default in 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 breadcrumbs 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.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

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 *HP Diagnostics Installation and Configuration Guide*.

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 4, "Working with Diagnostics Data Displays."

Interpreting the Hosts View

The Hosts view can be used in determining the health of a physical system or operating system. You can look at host system availability and operating system metrics for an indication of infrastructure problems and isolate problems to a particular host system. All the metrics that a host or operating system provides are trended over time.

If the metrics displayed in the Hosts view raises any concerns, you can navigate to other views to find out more information. For example, from the Hosts view, you can navigate to the probes on a given host, and then into the server requests captured by one of the probes.

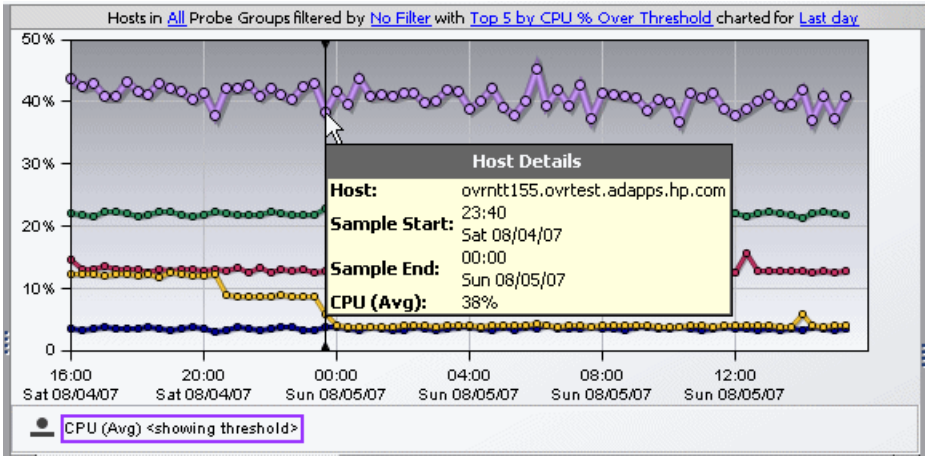
Displaying Host Details from the Charted Metrics

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 fully qualified 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.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** 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.

Navigating to Other Views from the Hosts View

You can navigate to other views from a host 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 Navigations 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 18, "Probes View."

16

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. For information about how to define layers in Diagnostics, see the *HP Diagnostics Installation and Configuration Guide*.

This chapter includes:

Diagnostics Load Explained

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.

The following formula provides a more specific explanation of how load is calculated in Diagnostics:

Load = Average Latency / Point Duration

Where:

- Average Latency = Total Time / Count
- Total Time is the sum of the latency of each of the threads that are executing in a layer.
- Count is the number of threads that executed in the layer during the time of interest.
- Point Duration is the length of time over which data has been aggregated to display a level of granularity in the Diagnostics views.

Accessing the Load View

You can access the Load view from the View bar in the Diagnostics views.

To access the Load view:

1. In the navigation pane of the Diagnostics Applications window, double-click the **Entire Enterprise** application. The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

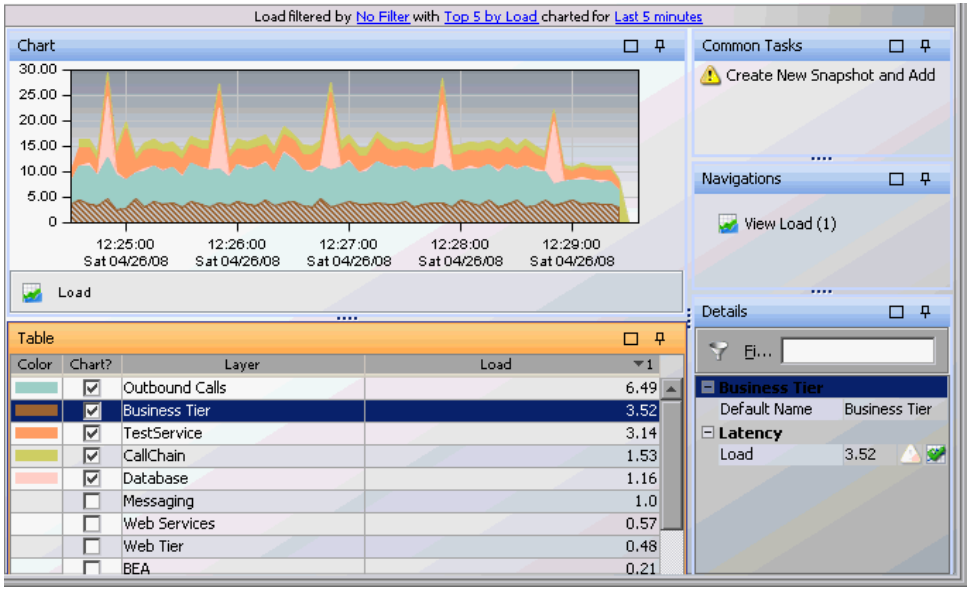
3 In the Standard Views view group, click **Load**.

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.

Note: From the Status view, you can drill down to the Load view for a particular probe or probe group. Right click a probe or probe group and select **View Load**. You can also drill down to the Trended Methods view for a particular probe from the Probes view.

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:



The Load view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, "Working with Diagnostics Data Displays."

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 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.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The **Details pane** in the Load view lists the metrics for the selected row of the graph entity table.

Customizing the Load View

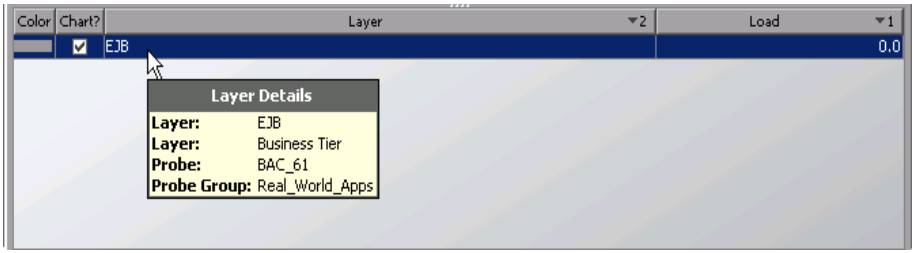
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Load view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 4, “Working with Diagnostics Data Displays.”

Interpreting the Load View

The Load view can be used to isolate a performance problem to a particular tier in an application. 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.

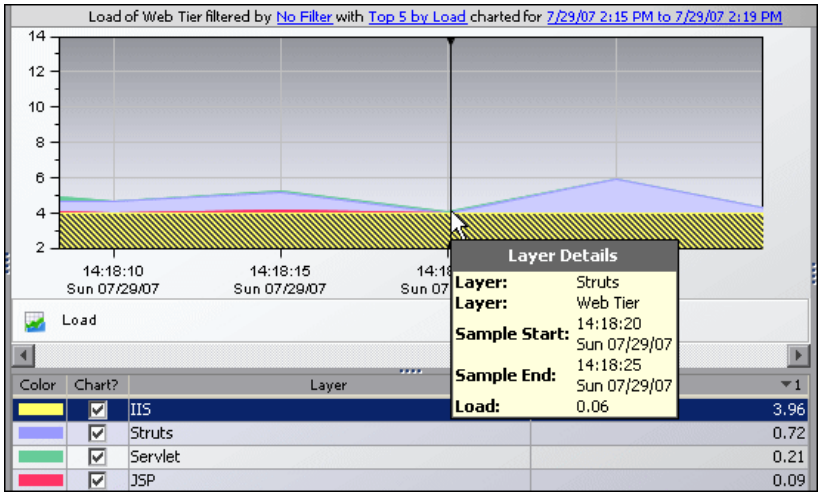


The information included in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Load view. In the example above, the Load view was accessed by drilling down from a probe in the Probes view through a layer in a higher-level Load view. The tooltip displays the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

- **Layer.** The name of the layer.
- **Layer.** The name of the parent layer from which you drilled down.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** 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 or selecting View Load in the Navigations pane.

Diagnostics displays the breakdown of the load for each sub-layer for the selected layer. If there are no sublayers defined for the selected layer, no navigation links to View Layers is provided for the entity.

17

Outbound Calls View

The **Outbound Calls** view shows server activity that is distributed between multiple servers (calls made from one VM to another).

This chapter includes:

Accessing the Outbound Calls View

You can access the Outbound Calls view from the View bar in the Diagnostics views.

To access the Outbound Calls view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.
3.  In the Standard Views view group, click **Outbound Calls**.

   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.

Note: You can also access this view by drilling down from the originating server request in the Server Requests view. Right-click the server request in the graph entity table and select **View Outbound Calls**.

Description of the Outbound Calls View

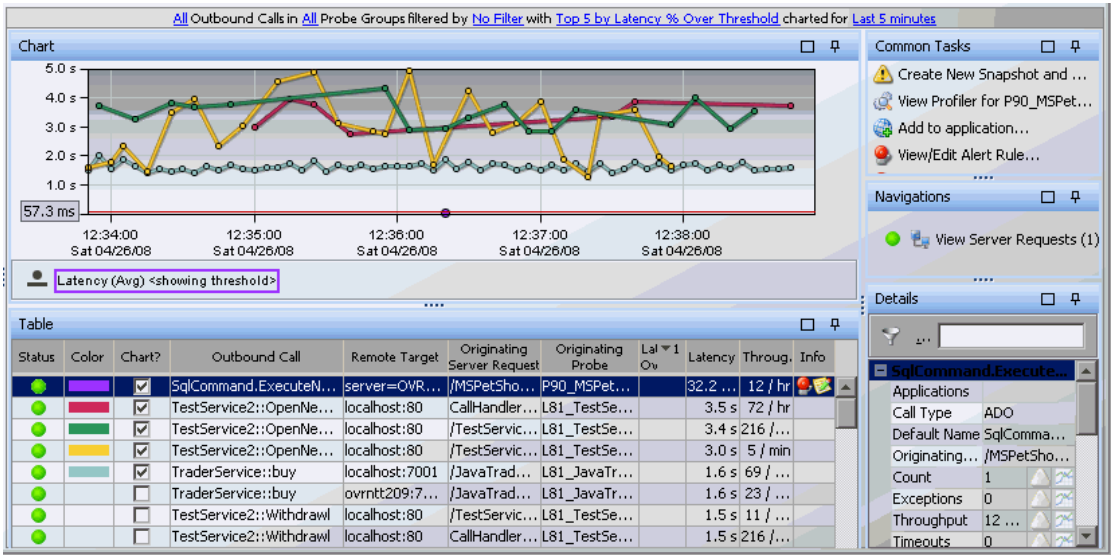
In Diagnostics, outbound Web service calls are displayed as remote calls within a server request.

Diagnostics monitors the following types of outbound calls:

- **Web Service.** Web services are also monitored in the SOA Service views. For more information, see "SOA Services Views" on page 419.
- **RMI**. Calls between Java servers.
- **RFC (SAP).** Calls between SAP servers.
- **CICS.** Calls within an IBM environment.

➤ **JMS.** Calls between Java servers and message servers.

The following image is an example of the Outbound Calls view:



The Outbound Calls view has the layout of a typical Diagnostics view. For information about the layout and controls of a typical Diagnostics view, see Chapter 4, “Working with Diagnostics Data Displays.”

Graph

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.

Graph Entity Table

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**).

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The Details pane in the Outbound Calls view lists the metrics for the outbound call in the selected row of the graph entity table.

Drilling Down from the Outbound Calls View

You can drill down from an outbound call by:

- double-clicking the row for the outbound call in the graph entity table
- right-clicking the out bound call's row in the graph entity table, and selecting **View Server Requests** from the menu
- double-clicking a trend line for the outbound call
- right-clicking a trend line for the outbound, and selecting **View Server Requests** from the menu
- clicking **View Server Requests** in the Navigations pane

When you drill down from an outbound call, Diagnostics displays the Server Requests view with details for the originating server request. From the Server Requests view you can view the call profile for a request. For more information on the Probes view, see Chapter 20, "Aggregate Requests and Server Requests Views."

18

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.

This chapter includes:

Accessing the Probes View

You can access the Probes view from the View bar in the Diagnostics views.

To access the Probes view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.
3.  In the Standard Views view group, click **Probes**.

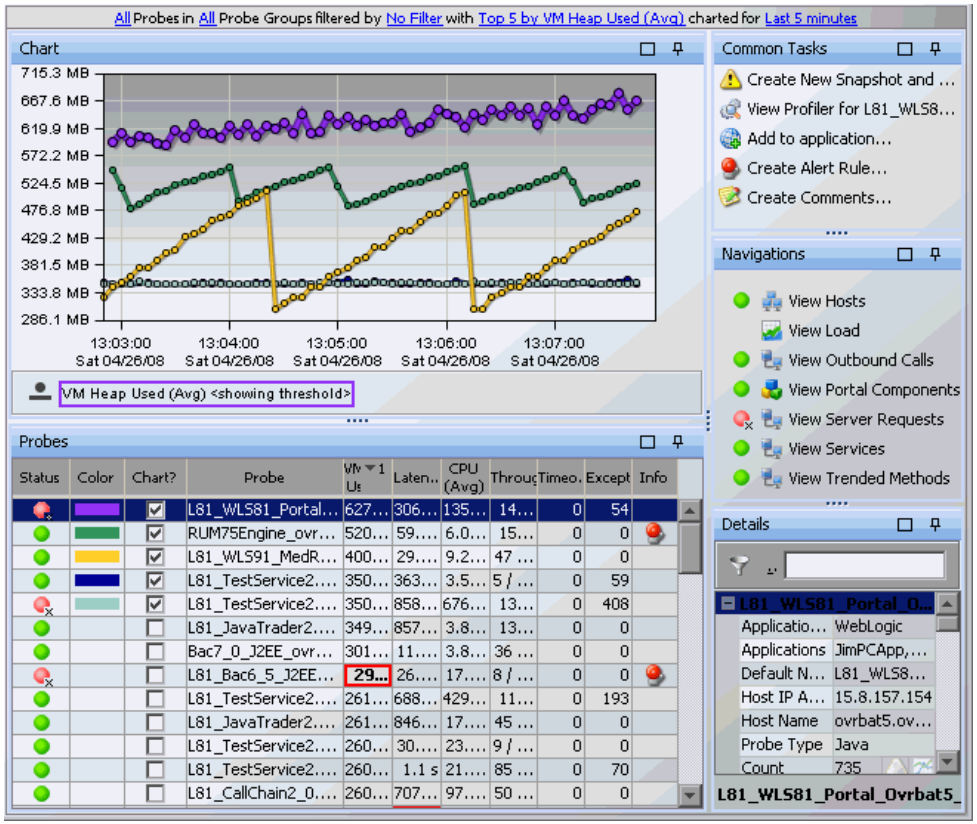
   The Probes view is displayed with the default settings so that the five probes that have the highest heap usage during the previous five minutes are displayed on the graph.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **Probes** link.

You can drill down to this view from the Host table in the Status view. Right-click a probe group in the Host table and select **View Probes**.

Description of the Probes View

The following image is an example of the Probes view:



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.

The Probes view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, "Working with Diagnostics Data Displays."

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.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

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.

Customizing the Probes View

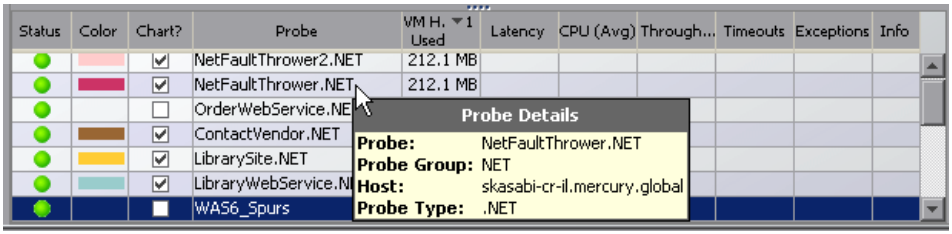
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Probes view. For more information about the ways you can control how performance metrics are presented in this view, see Chapter 4, "Working with Diagnostics Data Displays."

Interpreting the Probes View

Using the information displayed in the Probes view, you can isolate performance problems to applications monitored by a particular probe. 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:

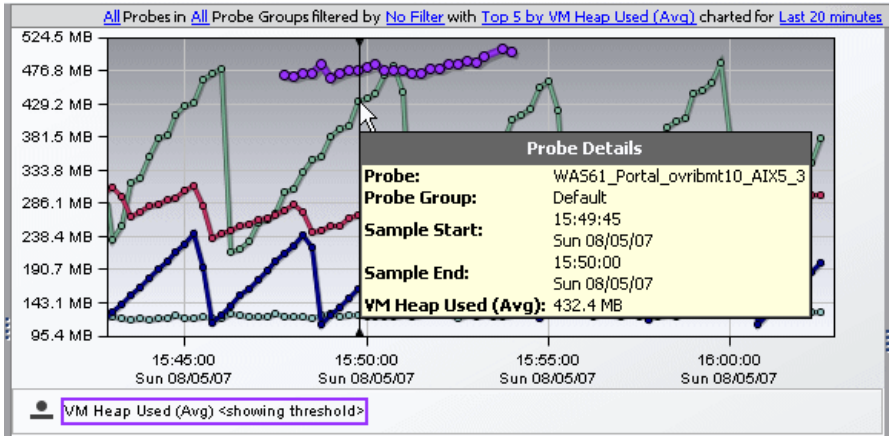


The **Probe Details** tooltip displays the following information:

- **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.** Java, .NET, Oracle, SAP, MQ, SQL Server.

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.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** 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 by:

- double-clicking the row for the probe in the graph entity table
- right-clicking the probe's row in the graph entity table, and selecting **View Server Requests** from the menu
- double-clicking a trend line for the probe
- right-clicking a trend line for the probe, and selecting **View Server Requests** from the menu
- clicking **View Server Requests** in the Navigations pane

When you drill down from a probe, Diagnostics displays the Server Requests view with more details for the server requests that are being run on the selected probe.

Other navigation links and common tasks are available for a probe depending on the type of data being monitored. Some possible navigation links include: View Hosts, View Load, View Portal Components, View Outbound Calls, View Services, View Trended Methods. Some common tasks you may want to perform in analyzing performance problems include: View Profiler for the probe and Create New Snapshot.

19

SQL Statements View

The **SQL Statements** view displays performance metrics for SQL statements executed by certain methods within your monitored environment.

This chapter includes:

Aggregating and Trending of SQL Statements

Important: The SQL Statements view should only be used when you are working with a Diagnostics probe from version 6.6 or later.

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 Java Probe in a deployment where Diagnostics is integrated with LoadRunner or Performance Center

If the Java Probe is running in a deployment where Diagnostics is integrated with Business Availability Center or where Diagnostics is running in standalone mode (no integration), SQL trending continues until the application monitored by the Java 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 **<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	Java Probe
Diagnostics Standalone	server.properties file	dispatcher.properties file
Integrated with 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 **<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 View bar in the Diagnostics views.

To access the SQL Statements view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

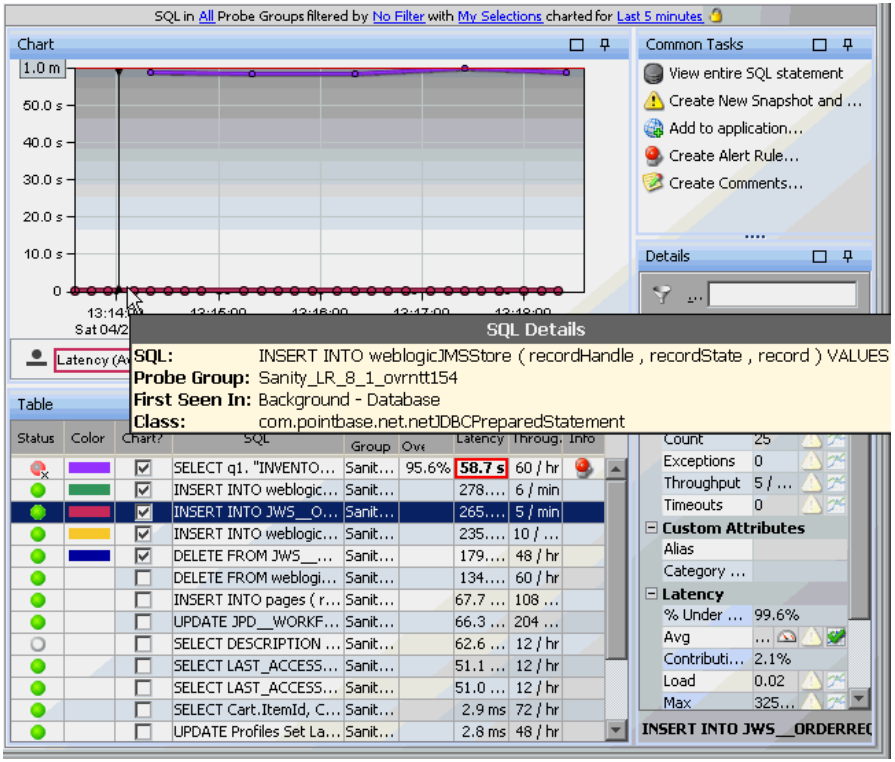
3. In the Standard Views view group, click **SQL Statements**.

   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.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **SQL Statements** link.

Description of the SQL Statements View

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



This view is presented in detail layout. For more information about views presented in detail layout, see Chapter 4, “Working with Diagnostics Data Displays.”

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.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

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 126.

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 select **View entire SQL statement** from the menu, or you can select the **View entire SQL statement** option from the Common Tasks menu.

20

Aggregate Requests and Server Requests Views

The **Server Requests** view provides the top entry points to the application server and is useful in identifying individual server requests with poor performance.

When multiple probes monitor the same application, the **Aggregate Requests** view shows server requests at the probe group level.

This chapter includes:

Aggregate Requests View

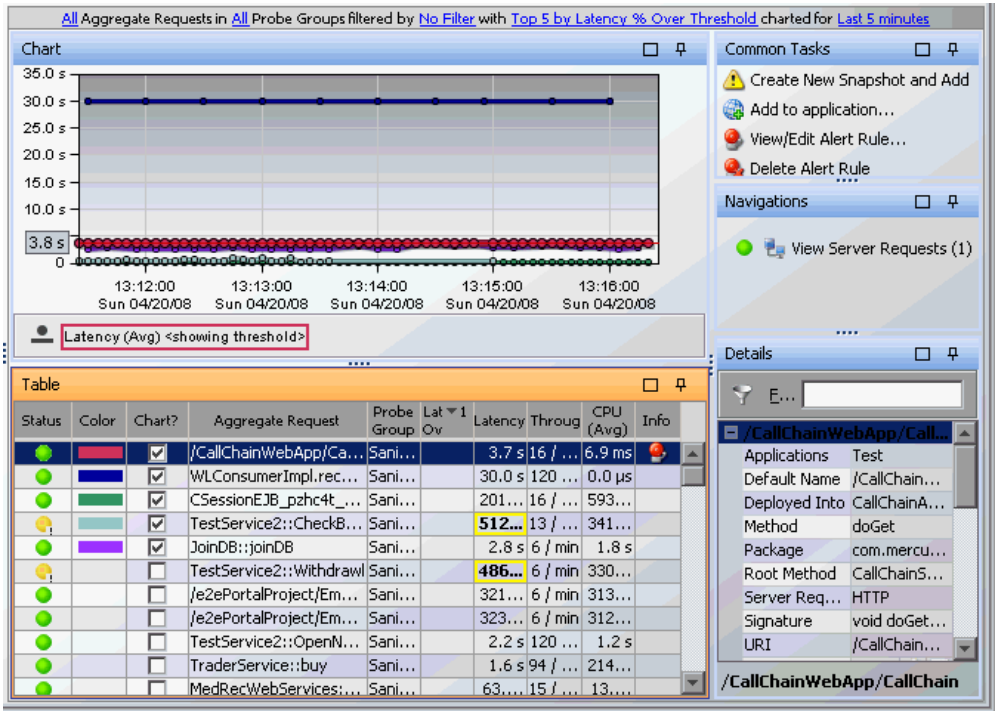
Server requests contain data relevant to a particular probe. When multiple probes monitor the same application, the server requests can be aggregated to the probe group level.

You can assign probes that monitor the same application to a single probe group. This group enables you to monitor a single aggregated data item, and to see the status and key metrics for a single item. If a problem is found, you can then drill down to the individual server requests within the aggregated data item.

For example, you could use the Aggregate Requests view to monitor the virtual machines that make up a distributed application that is deployed in a cluster. If you assign these probes to a "cluster" probe group, you can monitor this probe group's server requests and dependent services as a group with the aggregation of status, latency, and other key metrics. When a problem at the probe group level is detected, you can drill down into the server requests and dependent service calls.

You can access the Aggregate Requests view from the Standard views in the View Bar.

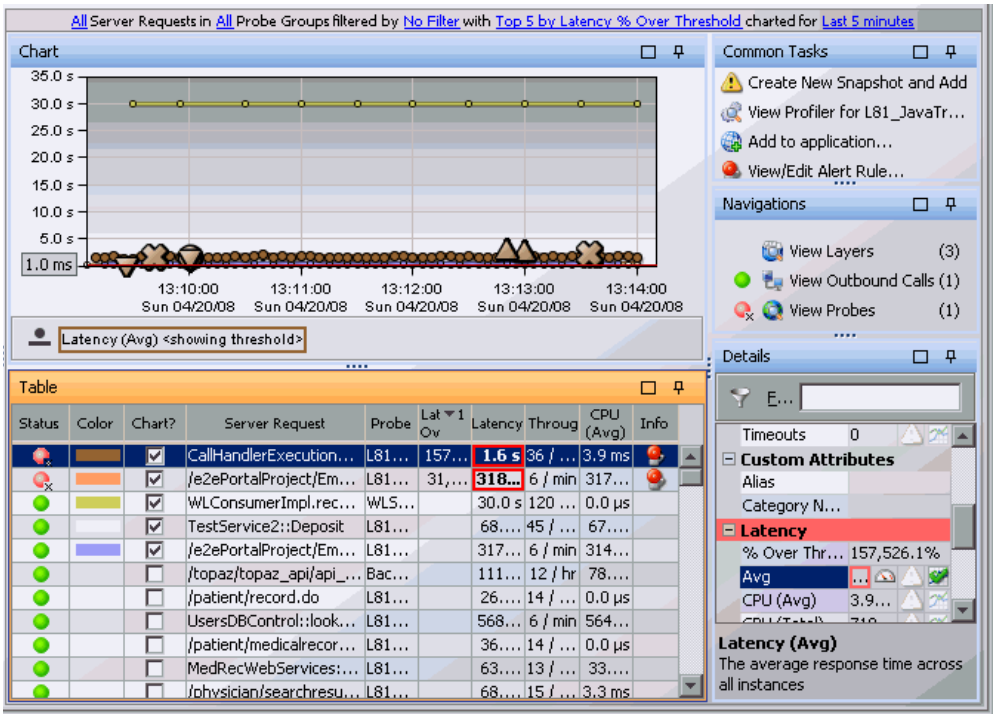
The Aggregate Requests view appears, as shown below.



From the Aggregate Requests view, you can drill down into the individual server requests.

Server Requests View

The following figure shows an example of the Server Requests view:



Graph

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

Graph Entity Table

The graph entity table in the Server Request view lists all of the requests that pertain to the context shown in the breadcrumbs displayed at the top of the view. If you navigated to the Server Requests view from the View bar, the Server Requests view shows all of the Server Requests for all the probes. If you navigated to the Server Requests view from a drill down (for example from the Probes view), then only the requests for the selected probe are displayed. The view filter specifies the **Server Request type**, **Time Range** and **Probe Group** for the metrics displayed in this table. Server request types can include the following: HTTP, Web Services, ADO, JDBC, JMS, RMI, SAP ABAP types, .NET Remoting, CICS.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The Details pane in the Server Requests view lists the metrics for the selected row of the graph entity table.

Accessing the Server Requests View

You can access the Server Requests view from the View bar in the Diagnostics views.

To access the Server Requests view:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **Standard Views** to open the Standard Views view group.

3 In the Standard Views view group, click **Server Requests**.

The Server Requests view appears with the default settings, so the graph displays the average latency for the five server requests that have the highest latency percent over threshold values.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **Server Requests** link.

You can drill down to Server Requests from the metrics reported in the Probes view, the Transactions view, and the Status view. You can also drill down to the originating server request for an Outbound Call, and drill down to the server requests for a probe from Topology.

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 "Working with Diagnostics Data Displays" on page 81.

Displaying Latency for Server Requests

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

You configure the display of latency 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 latency for each server request in the Server Requests view. Setting the value of these properties to false causes Diagnostics to stop reporting the latency for each server request in the Server Requests view.

The value of the latency for each server request is displayed in the Details pane. To configure the display of the latency column in the graph entity table, modify the graph entity table as described in "Customizing the Graph Entity Table" on page 109.

Interpreting the Server Requests View

You can use the Server Requests view to find the top entry points to the application server and isolate performance problems to a particular server request. Then you can drill down to a breakdown of methods and a call profile. The running trend line in the graph lets you see the average of all the instances of a single unique 'named' request. The instance tree icons let you view the minimum/maximum and average user request and compare them to the running average in the trend line. See "Drilling Down to an Instance Tree for a Server Request" on page 322.

When you drill down from a probe to server requests, all server requests for the probe are displayed, not just those filtered for the application you selected in the Applications window. When you are viewing data for a particular application there is a column in the graph entity table (**App?**) that indicates which server requests are part of the application context you selected.

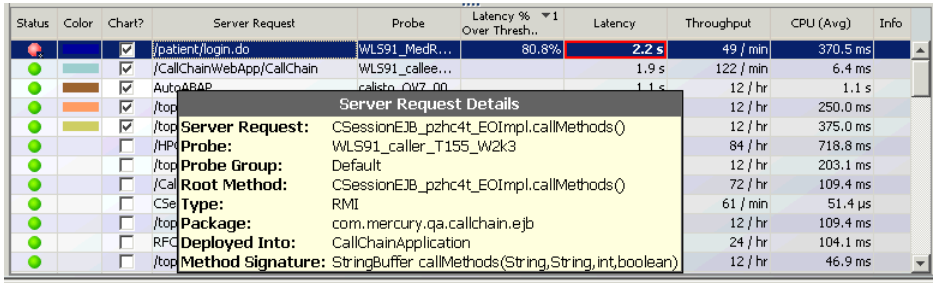
Depending on the application and workload, you may see an average of zero latency if you look at average response time on the probe. This result is caused by the low-resolution java timer. (The java timer cannot measure things that take less than 10ms on many platforms, including Windows.) The Server Requests that typically fall into this category include static content (for example, GIFs and JPEGs) as well as Database and housekeeping threads in an application. If you want non-zero response times, you can turn on the native (JNI) timestamps for high-resolution timings.

If a server request involves EJB business method invocations you can see these methods in the call profile and see a business tier in the Load or Layers views.

Important: Diagnostics does not track forwarded requests (HTTP redirection with HTTP 30x response code).

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. An example as shown in the following figure:



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 name 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 method may be a portion of a URL or, in the case of an RMI call, a class and method representing the name of the server request itself.
- **Type.** The Server Request type. Examples of server request types include: HTTP, Web Services, ADO, JDBC, JMS, RMI, SAP ABAP types, .NET Remoting, CICS.

  In addition, you may see server request types named Pseudo and RootRename. Pseudo means the server request was not really a server request defined as something that 'hits' your server, like http. If you instrumented just a regular method, then the server request will show up as type pseudo.

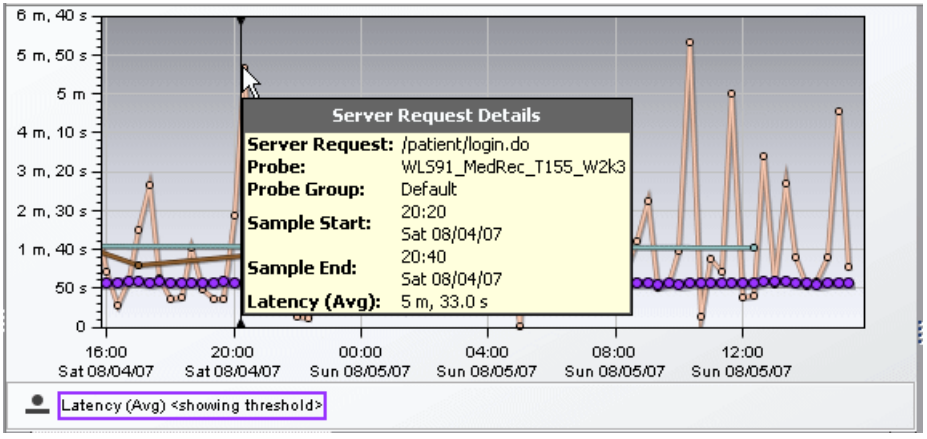
RootRename is also used for things that are not really server requests. Instead of pseudo the RootRename is based on a points file specification. These server requests will have a name that start with Background -. The suffix is the value of the rootRenameTo property specified in the point or the name of the first sub-layer if this property is not specified. To mark a point as a root rename, the detail 'when-root-rename' must be specified in the point. Refer to the *HP Diagnostics Installation and Configuration Guide* Custom Instrumentation sections for details.

- **Package.** The name of the package that contains the class from which the method was called.
- **Deployed Into.** Tells where the web service was deployed into (usually an EAR file).
- **Method Signature.** The signature of the root method.

Other types of details tooltips may be displayed depending on the type of server request (for example, SAP ABAP Server Request Details or Web Service Details).

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 figure.



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.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** 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

You can drill down from a server requests by:

- double-clicking the row for the server request in the graph entity table
- right-clicking the server request's row in the graph entity table, and selecting **View Layers** from the menu
- double-clicking a trend line for the server request
- right-clicking a trend line for the server request, and selecting **View Layers** from the menu
- clicking **View Layers** in the Navigations pane

When you drill down from a server request, Diagnostics displays the Layers view with more details for the selected server request. The Layers view displays a breakdown of the latency contribution for each layer in the server request.

Other navigation links and common tasks are available for a server request depending on the type of data being monitored. Some possible navigation links include: View Layers, View Probes, View Breakdown by Portal Components, View Outbound Calls. Some common tasks you may want to perform in analyzing performance problems include: View Profiler for the probe and Create New Snapshot.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.

Drilling Down to an Instance Tree for a Server Request

The Server Requests view displays a trend line showing the average of all the instances of a request. You select a server request in the Server Requests view, either by 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.

Special icons are displayed to mark the points at which instance trees (for example, minimum, maximum and average) have been created for significant method calls that were made during the execution of the server request. The instance examples let you view, for example, the minimum, maximum and average user and compare this to the running average in the trend line.

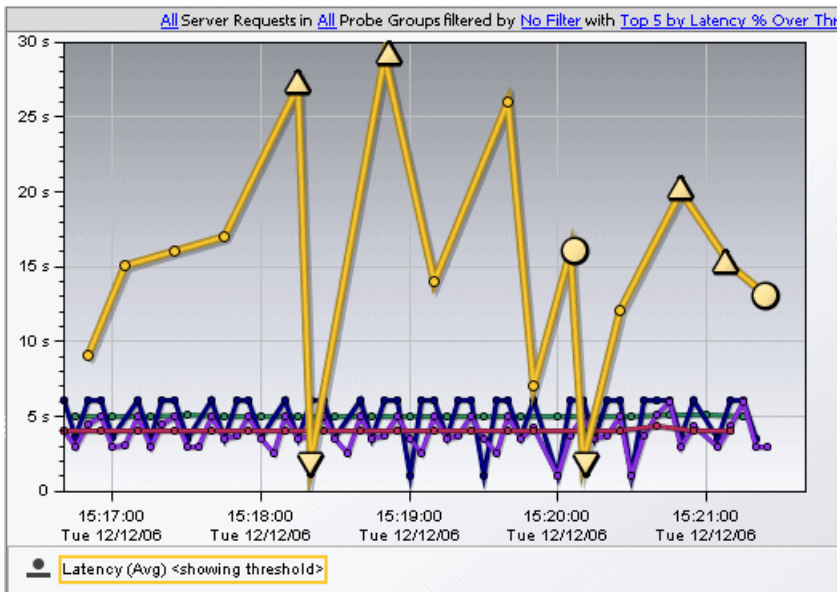
Visually analyzing the convergence of the min/max with the running average line lets you determine the real behavior of the application to determine if the application is seeing consistently bad performance or overall good performance with some outliers.

Once you've determined which instance to analyze, you can drill down from the icon to see a call profile. You can then compare a 'good' performing call profile with a 'bad' performing call profile. In addition, if you see a pattern in the convergence of the min/max instances with the running trend line you can create a snapshot and look for other events that follow a similar pattern.

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 about instance trees, see Chapter 9, “Instance Trees.” For more information about the Call Profile view, see Chapter 26, “Call Profile View.”

Note: The instance trees are available on the Min, Max, and Average Latency metrics and for cross-VM calls, exceptions and SOAP faults.

In the following figure, 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 minimum, maximum and average instance trees for the server request have been saved to help you understand the performance of your applications.

You may also see markers to indicate when there is an exception or a SOAP fault is thrown.

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

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 tree 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 tree depicts one of the best-performing invocations of the selected server request and resulting root method. It 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 tree depicts a typical invocation of the selected server request and resulting root method call. It can be useful for comparison to instances that perform poorly.

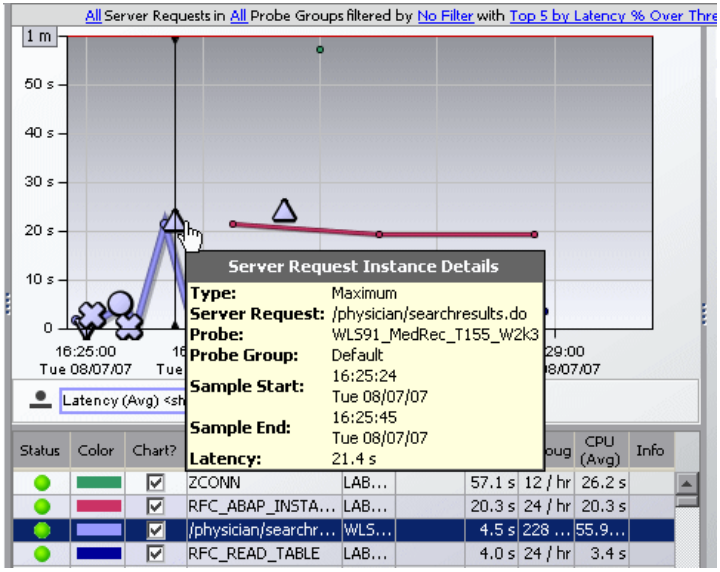
- **Cross VM Instance Tree**. The cross VM instance tree for a server request represents an instance tree that includes a call to another server request by using a technology such as RMI or Web services.

The following icons are used to represent exceptions and faults. Additional information is collected by the probe when exceptions or SOAP faults occur. You can drill down to the call profile graph and view exception or fault details for a method, including the stack trace.

- **Exception.** The exclamation mark indicates where an exception occurred. When you click the icon, a call profile opens. This call profile has the method that threw the exception marked with a yellow dashed outline and an Exceptions tab providing details on the exception including the stack trace.

- **SOAP Faults.** The icon indicates where an soap fault occurred. See “Types of SOAP Faults” on page 447 for details on the icons used to represent different types of faults. When you click the icon, a call profile opens. This call profile has a SOAP Faults tab providing details on the fault, including the stack trace.

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 figure.



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.
- **Sample Start.** The start time for the server request invocation represented by the selected instance tree.
- **Sample End.** 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 Requests 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.

Note: It is possible that the instance tree that is displayed for an instance tree marker is different than the one you anticipated. This can happen if the instance tree on the Diagnostics Server was replaced since the last update to the UI. As a result, even though a specific tree was selected, another tree that matches the type of the selected instance tree is displayed.

For more information about the Call Profile view, see Chapter 26, "Call Profile View."

21

Status View

The **Status** view displays a table that gives you a quick view of the status and performance of all the probes and hosts in your monitored environment. Understanding how to use the controls and features of this view will make it easier for you to analyze the performance characteristics of your applications.

This chapter includes:

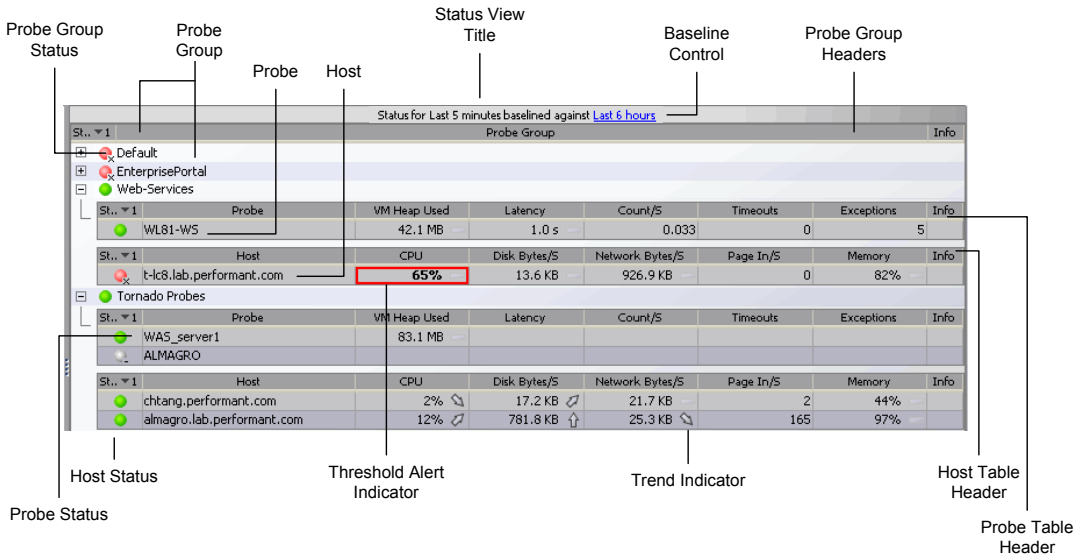
About the Status View

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 several sub-tables.

- **Probe Table.** Lists all of the probes in the probe group.
- **Host Table.** Lists the hosts for each of the probes in the probe group.
- **Monitors Table.** If you have configured an integration with SiteScope to forward data to Diagnostics you will see the Monitors table, which lists the SiteScope monitors in the probe group.

  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, for each host in the host table and for each SiteScope monitor in the monitors table.

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



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. The monitors table is also shown if you have configured integration with SiteScope.

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.

Monitors Table

The third table is the monitor table. You will only see this table if you have configured an integration with SiteScope to forward data to Diagnostics. See Chapter 37, "External Monitors Views" for more information.

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

To access the Status view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.
3.  In the Standard Views view group, click **Status**.

   Diagnostics displays the 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 4, “Working with Diagnostics Data Displays.”

Note: Changing the global time range has no affect 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 104.

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	**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 recent 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.

The following describes, in general, how a status of RED or YELLOW is determined:

- Diagnostics compares the average value for the metric in the last five minutes to the threshold. If the threshold is crossed, then the status is RED.
- If not, Diagnostics counts the total number of points (five second averages from the probe) that violate the threshold. If the count is greater than three but the average is still under the threshold (current status is not RED), then the status is YELLOW; otherwise the status stays at GREEN.
- If none of the above, the status is GREEN.

Status indicators can progress in severity from yellow, and then to red as the metric's performance deteriorates. However, the indicators do not regress in the same way. Once a red indicator has been issued, it will continue to be displayed as red until the metric value has returned to normal levels.

For information on setting thresholds, see "Setting Metric Thresholds" on page 131.

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:

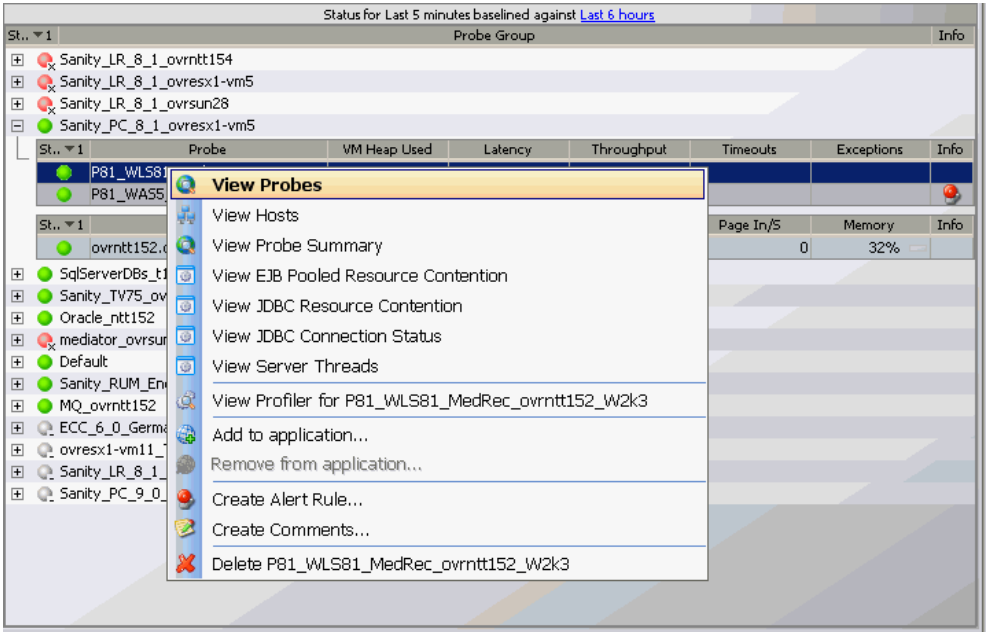
- **Probe.** The name of the selected probe.
- **Probe Group.** The name of the probe group to which the selected probe was assigned when it was installed.
- **Host.** The host name or IP address of the host for the probe. This host is also listed in the Host table in the same probe group.
- **Probe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

Drilling Down from the Status View

From the Status table, you can drill down into the details for the probe groups, probes and hosts that are behind the status that you see in the view. And you can take actions that are relevant to the entity selected.

Right-clicking a row in the Status table displays a pop-up menu of suggested actions you can take to further analyze a performance problem and diagnose the cause.

When you select a row in the table, Diagnostics updates the right-click menu items so that they contain only **relevant** navigations and actions that are appropriate for the selected entity. An example of the relevant navigations and actions for a probe is shown below:



In general, the menu items can be described as follows:

- Some navigation links (such as View Probes, View Server Requests, View Load) provide a drill down to more detailed information in a standard view such as the Probes view.
- Some navigation links (such as View JDBC Connection Status) are specific to the type of probe selected and display more detailed information in a specialized view group such as the IBM WebSphere view group.
- The View Profiler... navigation link opens the Java profiler or the .NET profiler which you can use to analyze method latency and monitor memory usage in your application.
- Common actions are also available in the pop-up menu. Some typical actions you might want to take when analyzing a performance problem include: Create Alert Rule, Create Comments, Add to Application. See “Common Tasks” on page 116 for more information on these actions.

Navigation links for a probe group, probe or host are filtered by the selected probe group, probe or host and display data for the previous five minutes, regardless of the global time range.

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

When you double-click a row in the host table, Diagnostics displays the Hosts view for the selected host.

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.

22

Topology View

The **Topology** view provides a graphical representation of the way your applications or business processes are working. This view contains a topology diagram that shows interactions and relationships between key application components, such as server requests, application servers, and databases.

This chapter includes:

Accessing the Topology View

You can access the Topology view from the View bar in the Diagnostics views.

To access the Topology view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.
3.  In the Standard Views view group, click **Topology**. Select an entity in the table to generate a topology diagram.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **viewing the application topology** link.

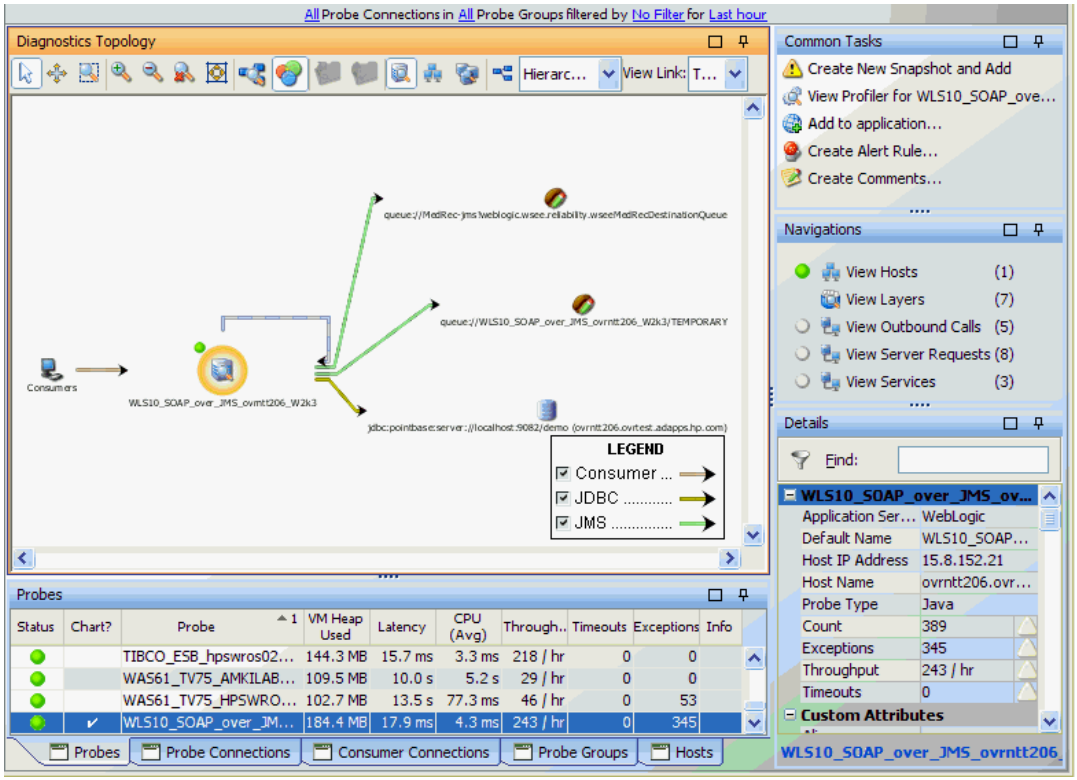
You can drill down to this view from transactions in the Transactions view. Right-click the transaction in the graph entity table and select **View Topology**.

When accessed through the Transactions view, topology data is limited to the probes and connections that are part of the transaction selected.

Note: The probe connections that are available, and subsequently shown in this view, are based on probe instrumentation.

Description of the Topology View

The following is an example of the Topology view.



Diagnostics displays the topology diagram where a graph is normally displayed. The topology diagram is a mapping of entities in an application and the connections between entities. It provides you with a visualization of your application or business process. See Chapter 6, “Working with the Topology Views” for details on using the topology toolbar and other controls in the topology views.

In the example above, the probe ID is shown under the probe icon. Calls being made to various targets are shown as connection arrows. The remote target ID indicates the type of call being made. For example jdbc: indicates a database call, queue: or topic: indicates a JMS call. The legend describes the types of calls.

Note: With JMS calls the temporary queues are grouped into a single node. Diagnostics matches the queue/topic name to a list of regular expressions in the **capture.properties** file to find the temporary queue/topic names. The ones that match are replaced with either queue:<probe-id>\TEMPORARY or topic:<probe-id>\TEMPORARY depending on the type.

When you first access the Topology view, you see a message in the place of the topology diagram instructing you to select a entity in the table to generate a topology.

The table in the Topology view contains the following tabs:

- Probes
- Probe Connections
- Consumer Connections
- Probe Groups
- Hosts

All the possible entities and connections in your enterprise are shown in the various tabs of the table. Once you generate a topology, each tab in the table will have check marks in the Chart column for those items that are represented in the currently displayed topology.

If you want to generate a different topology, you can click through the tabs in the table and select the entity you want to view from the appropriate tab. A new topology is then drawn representing the entity you just selected. Notice that the table will have different check marks in the Chart column of each tab for those items represented in the newly drawn topology.

You can also navigate to rows and tabs in the table by selecting entities in the currently displayed topology diagram (selectable entities have a blue 'halo'). For example, if the topology displayed has five probes, you will see check marks next to the five probes in the Chart column of the Probes tab. If you select a probe in the topology, the row for that probe in the table is highlighted.

If you select a connection in the topology diagram, the Probe Connections tab or Consumer Connections tab in the table will open with a check mark in the Chart column next to the consumers or probes represented by the connection arrow selected.

You can select the Group Probes by Host and Group Probes by Probe Group toolbar buttons in the topology diagram to see which hosts or probe groups are represented in the currently displayed diagram.

If you generate a topology diagram by selecting a host in the Hosts tab of the table, then the Group Probes by Host toolbar button in the topology is automatically selected. If you generate a topology diagram by selecting a probe group in the Probe Groups tab of the table, the Group Probes by Probe Group toolbar button in the topology is automatically selected.

The topology display includes a toolbar with controls that allow you to pan, zoom, and alter the way the diagram is drawn by grouping entities, changing diagram type, or changing connection (link) information. To find out how to use the diagram tool bar, see Chapter 6, “Working with the Topology Views”.

Note: You can right-click items on the diagram to see a tooltip containing Name, Type, Latency, and Throughput.

Filters

You can filter the data by **Probe Connections**, **Probe Group** and **Time Range**. And you can select to **Filter by** a particular string contained in a selected field like Target Name. The filter selections are located above the Topology toolbar.

All the filters apply to the topology diagram. The Filter by and Probe Connections filters are applied to the two Connections tab tables. The Probe Groups filter is applied to tables in all tabs.

So, for example, you have a topology diagram that includes both Web Service calls and JMS calls. Then you select to filter Probe Connections by Web Service. A new query is made for just web services and the topology is redrawn. The new topology diagram will only show the web service calls and not the JMS calls.

If you select a filter that results in no matching data the diagram will be replaced by the initial topology message.

Probe Icons

The probe icons generally represent the applications that are being monitored. Probe icons (and other entities) with a blue 'halo' are selectable in the topology. When you select a probe in the topology diagram, the probe is displayed with a colored halo to indicate that it is the selected entity. The metrics for the selected probe are displayed in the details table, and the row for the probe in the table is shown as selected.

The colored dot next to each probe in the diagram indicates the status of probe, just as it does in the Status view and in the table.

Consumer Icon

The consumer icon represents calls to a probe (or other entity) coming from a non-instrumented source (not monitored by a probe). These calls are grouped together in the topology and are represented by the consumer icon.

You can't select the consumer icon but if you want to see which consumers are making the calls represented in the topology you can select the consumer connections arrow coming from the consumer icon. The Consumer Connections tab then opens to display details about all the different consumers making calls to a particular entity.

The default consumer ID is the IP address (for SOAP over HTTP web services) and queue name (for SOAP over JMS web services) but you can configure consumers in a number of ways. See "About Consumer IDs" on page 433 for details.

Connections (Links)

The connection arrows in the diagram represent aggregated calls to and from application components within your environment. Connections can be shown between various entities such as probes, probe groups, Web services and consumers, or targets. The arrow on the connection indicates the direction of the communication.

Note: Connections can be shown between Java and non-Java systems such as SAP and Oracle or SQL Server database systems.

The color of the probe connection can indicate the type of call or performance status and load, depending on the selection from the View Link drop-down menu.

The Legend shows the meaning of each connection line in the current display, according to whether you have selected Type or Load from the View Link drop-down menu.

When you click a probe connection in the diagram, the line is emphasized with a black stripe (indicating that you have selected it). You can then view information about that specific connection in the Consumer Connections or Probe Connections tabs in the table. The data displayed in the details pane is specific to the connection.

Consumer connections are indicated by an arrow pointing from the consumer icon to a probe (or other entity). The consumer icon shows the origin of an aggregation of calls coming in to an entity from a non-instrumented source (not monitored by a probe).

Probe connections are indicated by an arrow pointing to or from the probe to a target or other entity. These targets show the destination of an aggregation of calls from the caller's point of view.

When you select a probe connection in the topology diagram you may actually see several rows highlighted in the Probe Connections tab of the table. This is when the application monitored by one probe is making an inbound call to a target (not monitored by a probe) and then an outbound call is made from the target to the application monitored by another probe. In this case, since multiple connections are selected, the details pane will be empty. To see details select the connection of interest.

Probe Groups

The probe groups nodes represent groupings of probes. When you select a probe group from the probe groups tab in the detail table, the probe group is displayed with a colored halo to indicate that it is the selected entity. The metrics for the selected probe group are displayed in the details table, and the row for the probe group in the table is shown as selected.

The colored dot next to each probe group in the diagram indicates the status of probe group, just as it does in the Status view and in the table.

Hosts

The host nodes represent hosts being monitored. When you select a host from the hosts tab in the detail table, the host is displayed with a colored halo to indicate that it is the selected entity. The metrics for the selected host are displayed in the details table, and the row for the host in the table is shown as selected.

The colored dot next to each host in the diagram indicates the status of host, just as it does in the Status view and in the table.

Status Bulb

The status bulb indicates probe status, which is based on a variety of factors, including metric thresholds. Status is only available when you view time ranges that are an hour or less. Status is shown as gray when no status is available, or when the time range exceeds an hour. See “About the Graph Entity Table” on page 101 for more information on how status is calculated.

Targets

The target icon represents calls to or from a non-instrumented source (not monitored by a probe). 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. You cannot select targets in the diagram. 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.

JDBC Targets

JDBC calls can be made to the following supported JDBC drivers: Oracle and SQL Server. For these supported JDBC drivers the target in the topology shows information such as host:port:sid or host:port:dbname.

For unsupported JDBC drivers (not Oracle or SQL Server) Diagnostics collects the JDBC connection URL. To avoid showing the raw URL (which may contain passwords), the **jdbc.url.length** property in **server.properties** is set to 0 by default which results in the unsupported JDBC connection URLs not being shown in the topology. Instead they are grouped into a target node called "Unsupported Database_<url's hash code>". The purpose of the hash code is so the entry is unique. This property does NOT apply to the supported JDBC drivers (Oracle, SQL Server).

If instead, for unsupported JDBC drivers, you want to see a portion of the JDBC URL you can specify a **jdbc.url.length** greater than 0 (depending on how much of the URL you want to see). This may allow you to avoid showing the full raw URL containing password. For example if you specify jdbc.url.length=5 you will see "Unsupported Database" and the first 5 characters of the raw JDBC URL instead of the url's hash code.

You may also see a target called "Unknown Database" which means that Diagnostics could not collect the JDBC connection URL.

Details Pane

The Details pane in the Topology view lists the metrics for the selected row and selected tab of the table. For information about the details pane see Chapter 5, "Working with Metrics, Thresholds and the Details Pane."

Using the Topology Diagram

When Diagnostics first displays the topology diagram for a probe or probe connection that you selected from the table, the diagram is zoomed to fit the view. The default format for the diagram is hierarchical. 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 you can see the topology the way you want to see it. See Chapter 6, "Working with the Topology Views" for details on using the topology toolbar and other controls in the topology views.

When the topology diagram is first displayed, the probes are not grouped. Each probe is shown individually in the diagram. You can change the way the diagram is drawn by selecting to group probes by hosts or by probe groups. Grouping probes can simplify the appearance of the diagram.

Grouping Probes by Hosts

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

Grouping Probes by Probe Group

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

Displaying Ungrouped Probes

To cause the probes to again be displayed as individual probes in the diagram, click **Don't Group Connected Probes** in the toolbar.

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 this icon, or right-click the group icon, the group is opened so 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.

Drilling Down from the Topology View

You can drill down on the entities listed in the table by using the right-click menu for the listed entity, or by using the links in the Navigations menu.

Drilling Down on Probe Groups

From the Probe Groups tab in the table, you can drill down to detailed views specific to the type of application or data being collected by the probes in the probe group. For example, a probe group for monitoring a SQL Server database may have drill down selections for the SQL Server Probes and Wait Time specialized views, for a probe group with Java Probes you can drill down to the Probes view.

Drilling Down on Hosts

From the Hosts tab in the table, you can drill down to the Probes view for those hosts that have a probe installed.

Drilling Down on Probe Connections

From the Probe Connections tab in the table, you can drill down to the Probes view and Outbound Calls view for those entities that have a probe system as a destination.

Drilling Down on Probes

From the Probes tab in the table, you can drill down to the Hosts view.

In addition, you can drill down to detailed views specific to the type of application or data being collected by the probe. For example, a probe monitoring a WebLogic application may have drill down selections to the Layers view, Outbound Calls view, Server Requests view and the EJB Pooled Resource Contention, JDBC Resource Contention, JDBC Connection Status and Server Threads specialized views.

23

Transactions View

The **Transactions** view displays performance metrics for the transactions that are being executed by your applications.

This chapter includes:

Accessing the Transaction View

You can access the Transactions view from the View bar in the Diagnostics views.

To access the Transactions view:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **Standard Views** to open the Standard Views view group.

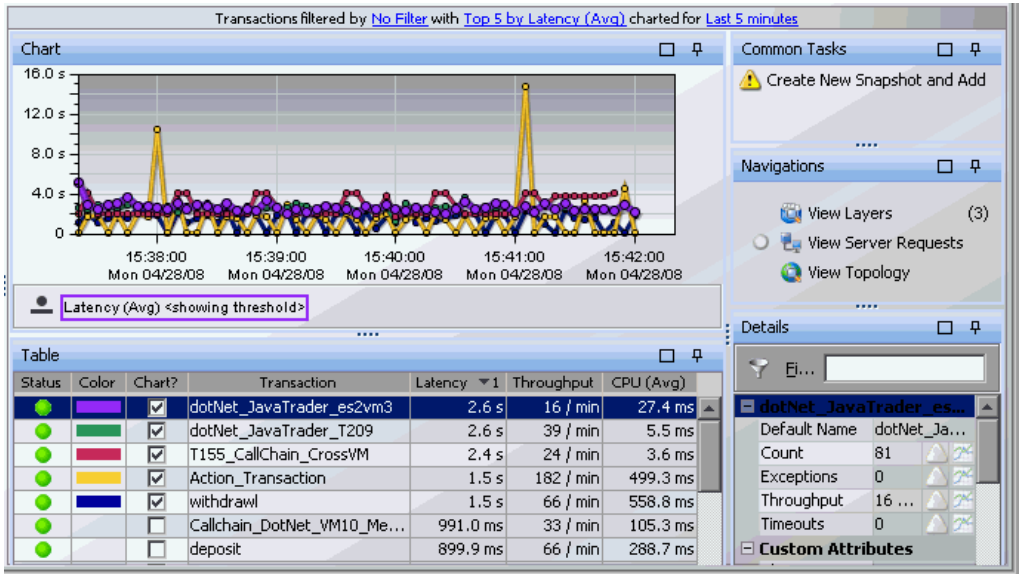
3 In the Standard Views view group, click **Transactions**.

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.

Note: You can also access this view directly from the Diagnostics Applications window. To do this, select an application in the navigation pane. In the details pane, click the **Transactions** link.

Description of the Transactions View

The following image is an example of the Transactions view:



This view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, “Working with Diagnostics Data Displays.”

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.

If a threshold has been set for the metric displayed (average latency) you will see a red horizontal line indicating the threshold value.

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.

Transaction data can come from BPM monitors or from LoadRunner or Performance Center if integrated with those products.

Graph Entity Table

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

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

Note: The View Server Requests status indicator and counts will always be unknown. You need to navigate to the view to determine the server request counts and status.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

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 Metrics, Thresholds and the Details Pane" on page 123.

Customizing the Transactions View

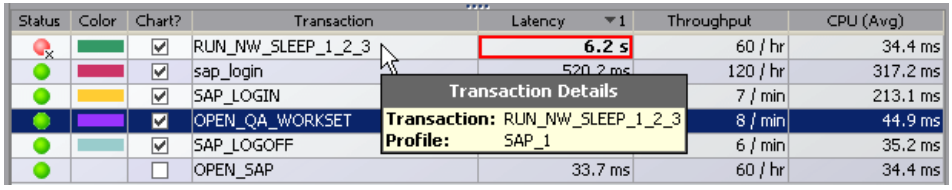
You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Transactions view. For more information, see Chapter 4, “Working with Diagnostics Data Displays.”

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:

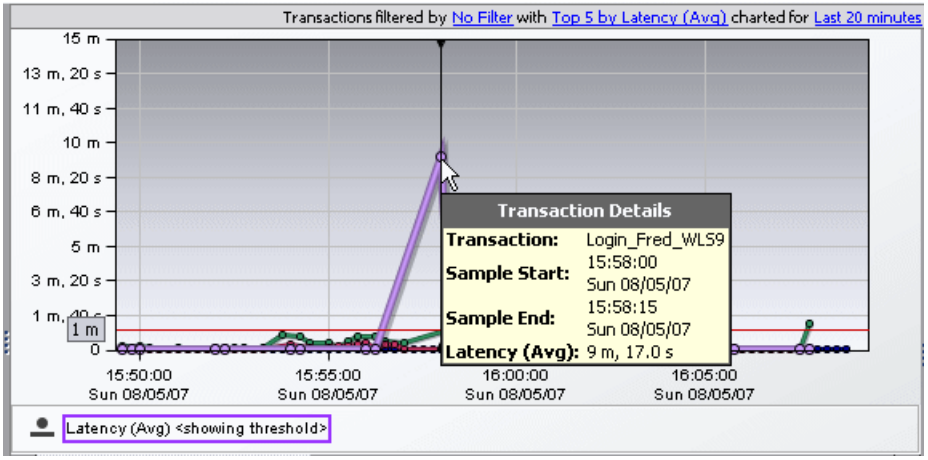


The **Transaction Details** tooltip displays the following information:

- **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.
- **Sample Start.** The start time for the aggregation period represented by the selected data point.
- **Sample End.** 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 options to: View Server Requests, View Layers, View Topology and Create New Snapshot and Add.

24

Trended Methods View

The **Trended Methods** view displays latency for methods you've configured for trending.

This chapter includes:

Configuring Instrumentation Points for Method Trending

In order to see data in the Trended Methods view you need to configure the instrumentation point for the method so that the latency for this method gets captured. This data can then be displayed as a trend line in the Trended Methods view.

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 Java Probe capture points file is located at **<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 View bar in the Diagnostics views or you can drill down to the Trended Methods view for a particular probe or probe group.

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 360.

To access the Trended Methods view:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **Standard Views** to open the Standard Views view group.

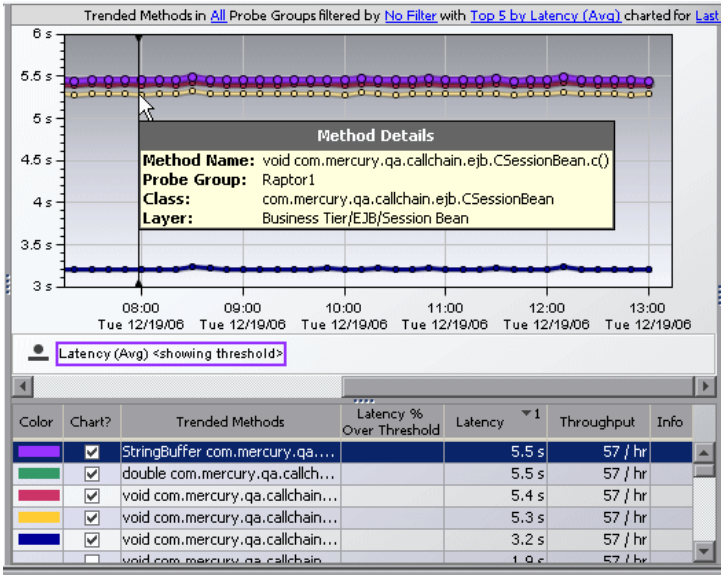
3 In the Standard Views view group, click **Trended Methods**.

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.

Note: From the Status view, you can drill down to the Trended Methods view for a particular probe or probe group. Right click a probe or probe group and select **View Trended Methods**. You can also drill down to the Trended Methods view for a particular probe from the Probes view.

Description of the Trended Methods View

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



This view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, “Working with Diagnostics Data Displays.”

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 359), 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 breadcrumbs 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 5, "Working with Metrics, Thresholds and 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 359), 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.

25

Layers View

The **Layers** view presents a breakdown of the processing across the layers in your application.

This chapter includes:

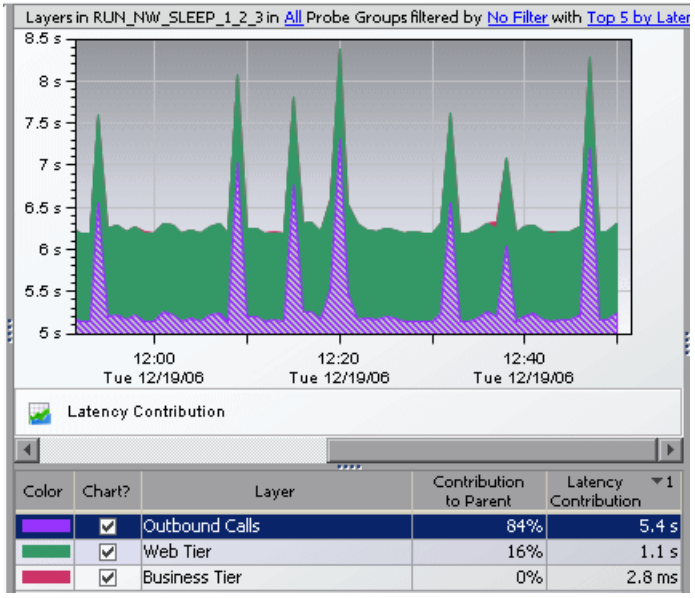
Accessing the Layers View

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, right-click a row in the Transactions view graph entity table and select **View Layers**.
- To drill down to the Layers view for a particular server request, right-click a row in the Server Requests view graph entity table and select **View Layers**.

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:



The Layers view has the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, "Working with Diagnostics Data Displays."

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.

Common Tasks and Navigations

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Details Pane

The Details pane in the Layers view lists the metrics for the selected row of the graph entity table.

Customizing the Layers View

You may have a class in your application that does not directly implement a Java component, but contains Java functionality that you would like to monitor. Or 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.

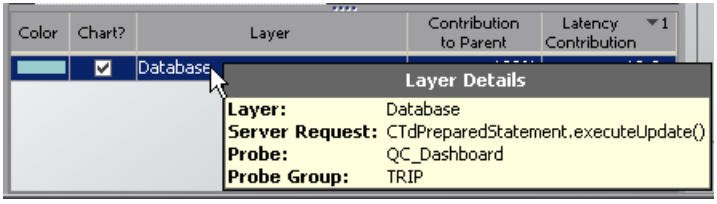
For instructions on configuring the instrumentation of the probes for custom layers, see the *HP Diagnostics Installation and Configuration Guide*.

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:

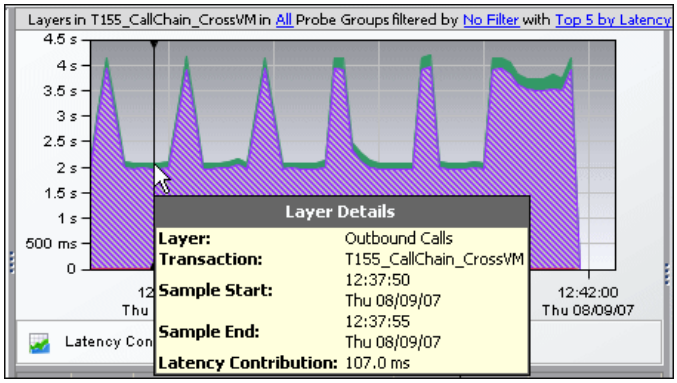


The information displayed in the **Layer Details** tooltip depends on which views were accessed before drilling down to the Layers view. In the example above, the Layers view was accessed by drilling down in the Server Requests view. The **Layer Details** tooltip displays the following information:

- **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 Requests 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.

- **Sample Start.** The start time for the sample aggregation period represented by the selected data point.
- **Sample End.** The end time for the sample 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.

26

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 call tree table. For an explanation of instance trees, see Chapter 9, "Instance Trees."

This chapter includes:

Accessing the Call Profile View

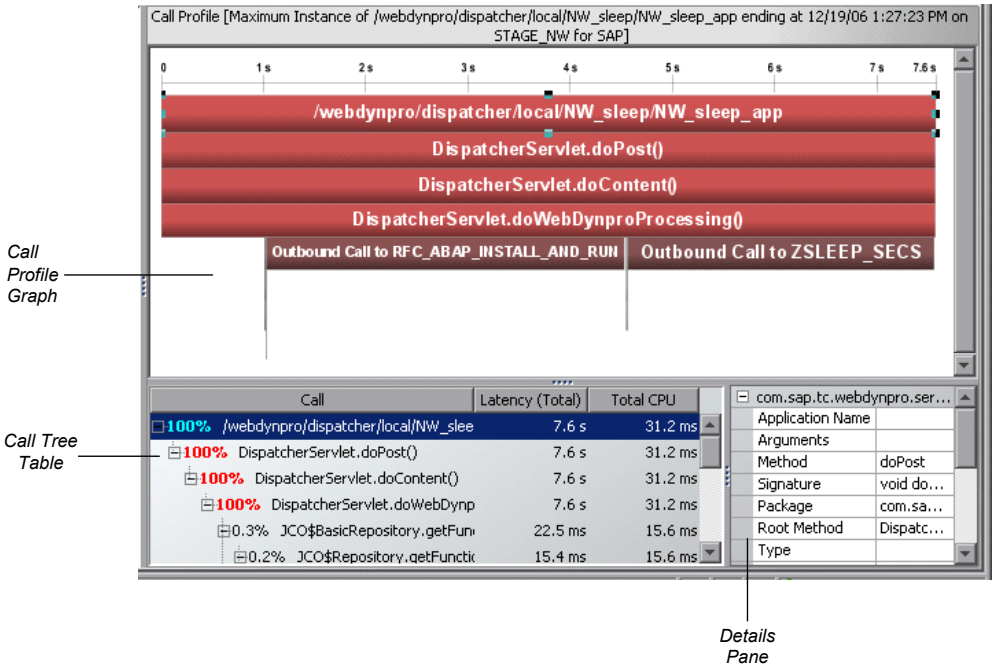
You can access the Call Profile view by drilling down from server requests in the Server Requests view.

To access the Call Profile view from the Server Requests View, see "Drilling Down to an Instance Tree for a Server Request" on page 322.

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.

Description of the Call Profile View

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



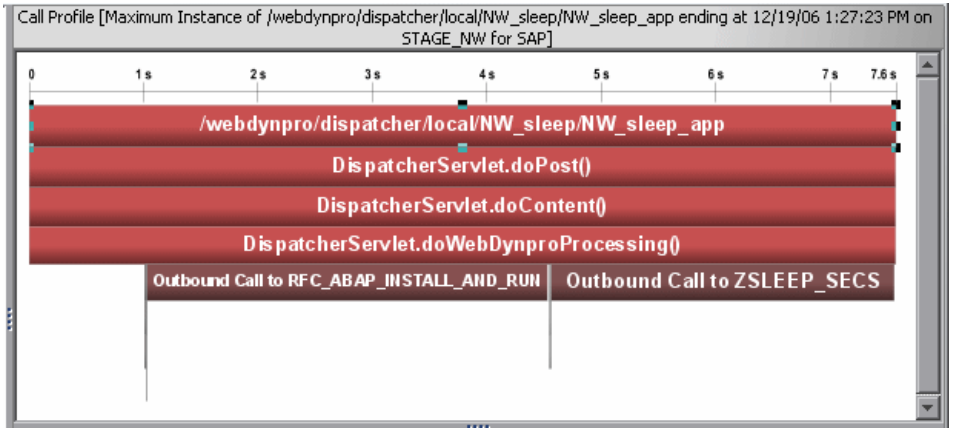
The Call Profile view is made up of three areas in addition to the View bar that is part of all Diagnostics views:

- Call Profile Graph
- Call Tree Table
- Details pane

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 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.

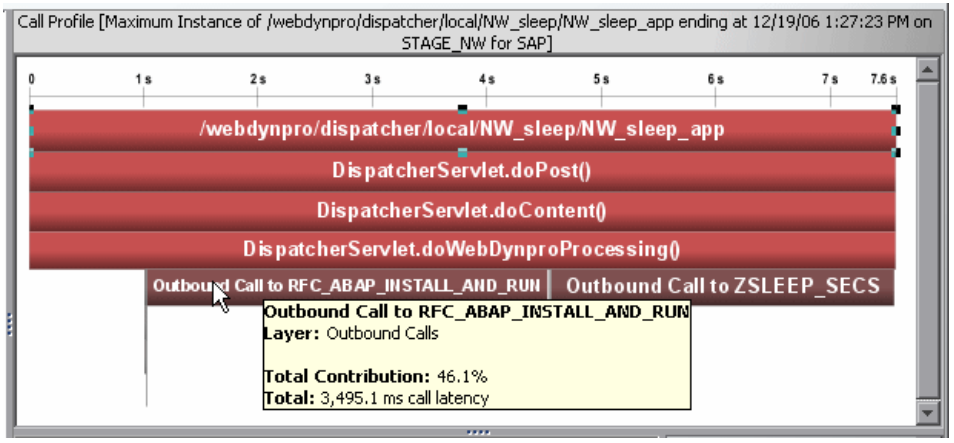
The gaps between the call boxes on a layer of the instance profile indicate one of the following processing conditions:

- 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.

The call profile graph may have tabs across the top if data for exception instances and SOAP faults was captured. See “Exceptions” on page 377 and “SOAP Fault Call Profiles” on page 379 for details.

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.

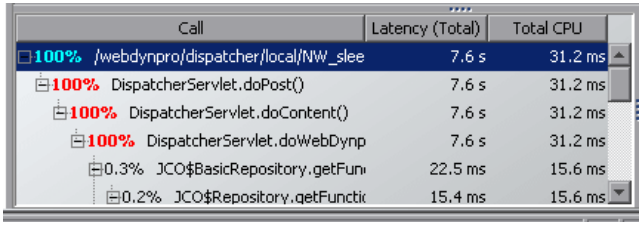


The tooltip displays the following call details:

- 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.



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.

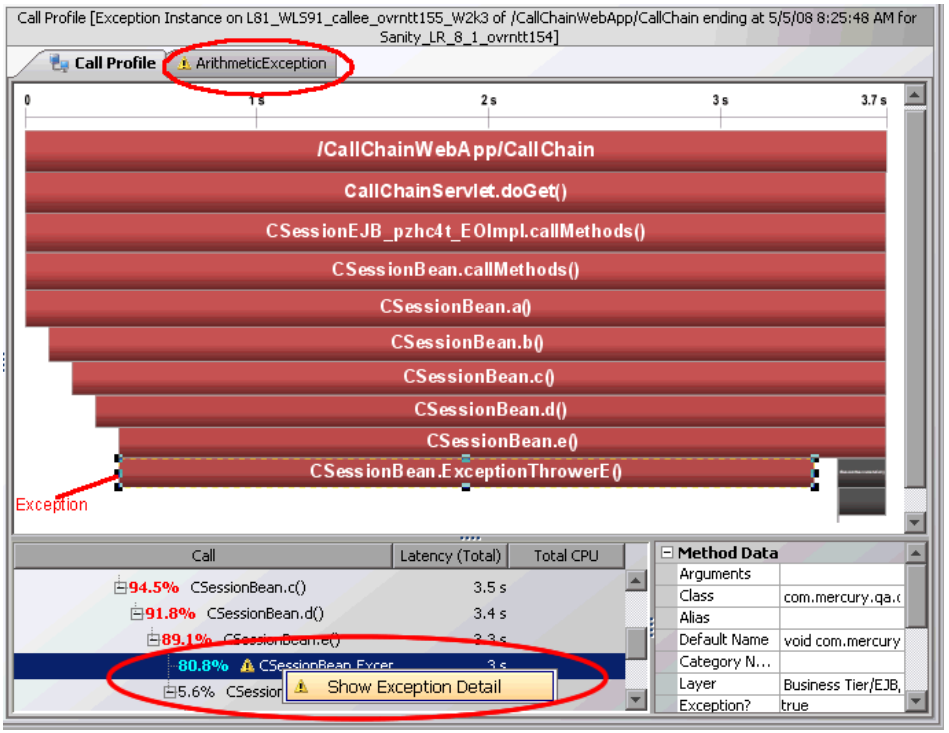
Description of the Details Pane

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 5, "Working with Metrics, Thresholds and the Details Pane."

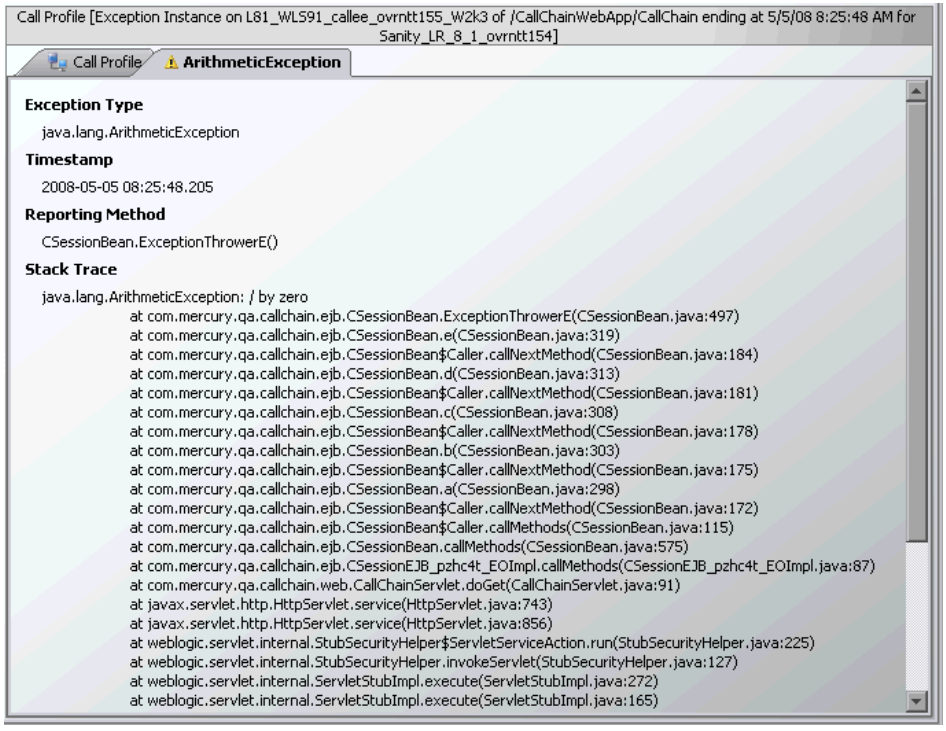
Note: The Details pane in the Call Profile view does not enable setting thresholds, adding to a snapshot, or charting other metrics.

Exceptions

If the call profile has any methods that threw exceptions, they are marked with a yellow dashed outline. In the call tree the method throwing the exception is marked by a small exclamation mark icon.



You can select the Exceptions tab to get details on the exception including the stack trace. Or you can right-click the exception in the call tree and select Show Exception Detail. This is useful when there are several tabs with detail data. The Exception tab is labeled with the exception name (not including the package and class information).



You can copy and paste the stack trace listing in this view to send to others for use in diagnosing the problem.

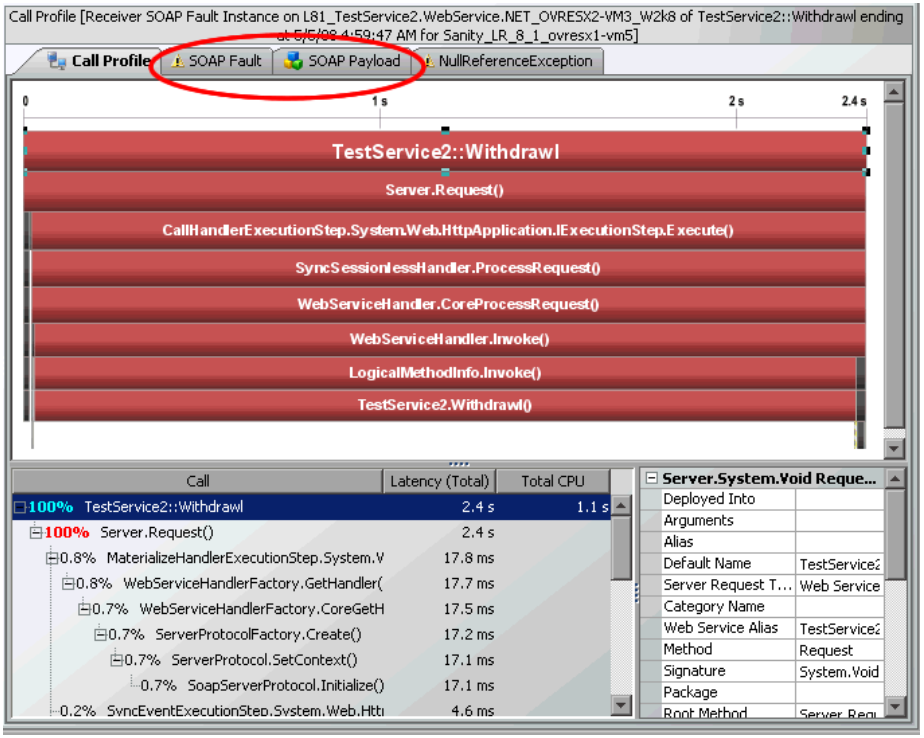
SOAP Fault Call Profiles

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 SOA Services - Operations 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 a **SOAP Fault tab** that displays detailed information about the SOAP fault.

You may also see a **SOAP Payload tab** if the payload was captured on the SOAP fault. The Diagnostics SOAP message handler is required for probes to support the capture of SOAP Payload. See "Payload on SOAP Faults" on page 449 for more information. See the *HP Diagnostics Installation and Configuration Guide* for details on configuring the SOAP message handler as well as configuring the probes to capture SOAP fault data.

For more information about SOAP fault call profiles, see "SOAP Fault Instances" on page 444.


Call Profile [Receiver SOAP Fault Instance on L81_TestService2.WebService.NET_OVRESX2-VM3_W2k8 of TestService2::Withdrawl ending
at 5/5/08 4:59:47 AM for Sanity_LR_8_1_ovresx1-vm5]
Call Profile
SOAP Fault
SOAP Payload
NullReferenceException
0
1 s
2 s
2.4 s
TestService2::Withdrawl
Server.Request()
CallHandlerExecutionStep.System.Web.HttpApplication.IExecutionStep.Execute()
SyncSessionlessHandler.ProcessRequest()
WebServiceHandler.CoreProcessRequest()
WebServiceHandler.Invoke()
LogicalMethodInfo.Invoke()
TestService2.Withdrawl()
Call
Latency (Total)
Total CPU
100% TestService2::Withdrawl 2.4 s 1.1 s
100% Server.Request() 2.4 s
0.8% MaterializeHandlerExecutionStep.System.V 17.8 ms
0.8% WebServiceHandlerFactory.GetHandler(17.7 ms
0.7% WebServiceHandlerFactory.CoreGetH 17.5 ms
0.7% ServerProtocolFactory.Create() 17.2 ms
0.7% ServerProtocol.SetContext() 17.1 ms
0.7% SoapServerProtocol.Initialize() 17.1 ms
0.2% SyncEventExecutionStep.System.Web.Htt 4.6 ms
Server.System.Void Reque...
Deployed Into
Arguments
Alias
Default Name TestService2
Server Request T... Web Service
Category Name
Web Service Alias TestService2
Method Request
Signature System.Void
Package
Root Method Server.Requ

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.
- See if any unexpected exceptions affected the performance of the operation.

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.

Asynchronous Thread Call Stack Sampling

Asynchronous thread call stack sampling for Java applications can be used to identify what the application is doing when exceptionally long executing times are observed. It is a collection technique in which some threads are periodically looked at and their state is recorded. The state includes the thread stack trace.

The main advantage of thread stack trace sampling is its ease of use, because no application re-start is required, sampling can be turned on and off dynamically and there's no additional overhead while sampling is turned off.

Important:

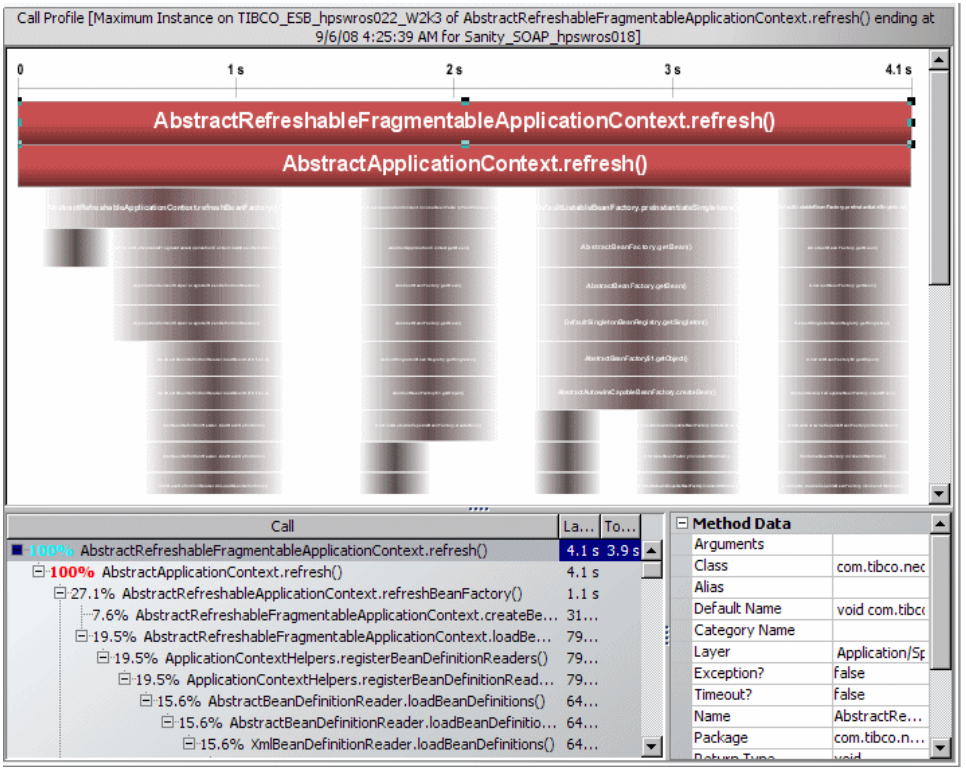
- To use thread sampling you need JVM 1.5 or higher.
- Make sure that all recommended operating system patches are installed.

Collecting stack trace samples is a costly and heavyweight operation for all contemporary Java Virtual Machines. At the same time, there's a natural trade-off between the quality of data reported by sampling, which calls for frequent sampling, and performance requirements for production environments.

Several properties are available to enable and control the sampling process. In **dynamic.properties**: enable.stack.trace.sampling, tardy.method.latency.threshold, stack.trace.sampling.rate, stack.trace.depth.max. In **dispatcher.properties**: enable.stack.trace.aggregation, aggregated.stack.trace.validity.threshold.

See the *HP Diagnostics Installation and Configuration Guide* chapter on Instrumenting Java Applications and Configuring the Probe from the Profiler for more information on how to configure thread sampling.

When asynchronous thread sampling is enabled you can see additional nodes added into the call profile view by sampling. These nodes are distinguished by their different (fuzzy) shading to emphasize lack of data about the represented method start and end times as shown in the picture below.



It is important to remember that the method invocations presented by the solid nodes - the part of the call profile corresponding to the instrumented methods - do not show direct call relationship, but merely the indirect call hierarchy and order.

The role of sampling is to show what the topmost (on the call stack) instrumented methods were doing, so the additional nodes are added only at the bottom (on the graph) of the Call Profile as obtained from execution of the instrumented methods.

An instrumented method will never be presented by a sampling (fuzzy) node, and vice versa, a non-instrumented method will never be presented by a solid node. Therefore, the only non-instrumented methods presented are those that were called directly or indirectly from the currently topmost instrumented method.

To avoid confusion, you should keep in mind that sub-trees consisting of sampling nodes are consistent only as long as the topmost instrumented method remains the same. For example, if a solid node A has two children: a sampling node B and a solid node C, we cannot conclude that C was called directly from A. It is possible that C was called from B, or any other non-instrumented method called directly or indirectly from A.

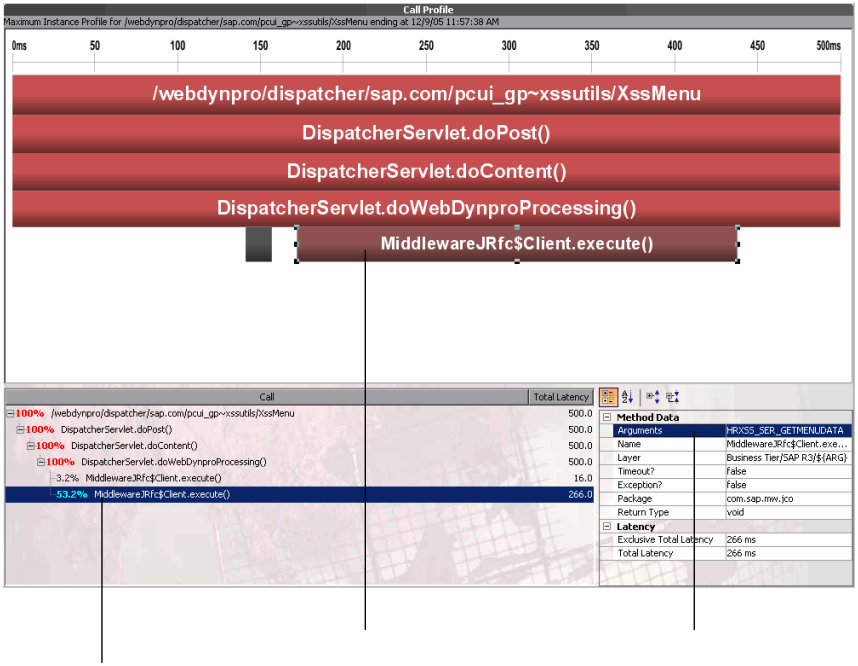
Each sampling node may represent a single, or multiple invocations of the method. No call arguments are available. Exceptions thrown by these methods are shown as thrown by the topmost (on the call stack) instrumented method. The only attributes shown by the sampling nodes are the call hierarchy and order, and an estimate of summary latency. Due to the nature of sampling the sub-graphs of sampling nodes are almost always incomplete.

See the *HP Diagnostics Installation and Configuration Guide* chapter on "Instrumenting Java Applications and Configuring the Probe from the Profiler" for troubleshooting information if you don't see any sampling nodes after enabling stack trace sampling.

Important: Dynamic instrumentation (the **Instrument** menu item which allows you to instrument sampled methods) is ONLY available for sampled data and you need to access the Call Profile from the Diagnostics Profiler for Java. It is NOT available when accessing the Call Profile from an instance tree icon in the main Diagnostics UI. See "Instrumenting a Sampled Method Dynamically" on page 558 of the Java Profiler section for details.

Analyzing Java to SAP ABAP Remote Function Calls

The Remote Function Call (RFC) protocol in SAP allows communication to take place between SAP Java and SAP ABAP environments. The remote calls that are made between SAP Java and SAP ABAP 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 SAP 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.

27

Collections and Resources Views

The **Collections** view and the **Resources** view show the collections in your system and the resource allocating the collections.

This chapter includes:

Using the Collections and Resources Views

The Collections view displays the collections in your system as well as the allocation point for each collection. The metrics available are minimum, maximum and average size of the collection. By default, the average size is charted.

The Resources view displays the resources that are allocating the collections (for example, a resource might be a .jar file). For each resource, the following metrics are available:

- Largest collection size
- Number of collection instantiations
- Total elements

A first look at the Resources view gives an understanding of which resource is the most expensive in terms of collection allocations. You can then move to the collections view (or the Diagnostics Profiler) in order to find the 'responsible' collections within that resource.

The Collections view and the Resources view have the format of a detail layout view. For information about the layout and controls in the detail layout, see Chapter 4, "Working with Diagnostics Data Displays."

Note: The Collections view and the Resources view are only available for the Java probe.

Enabling LWMD for a Probe

The option to select the Collections view or Resources view in the Navigations pane will not be visible unless you enable Lightweight Memory Diagnostics (LWMD) for the Java probe as follows:

1. Set the property **lwm.diagnostics.capture=true** in the **<probe-install-dir>/etc/dynamic.properties** file.
2. Set **active=true** (for lwmd) in the **<probe-install-dir>/etc/auto_detect.points** file.

Additional configuration is as follows:

- It may be desirable not to show all the collections in the Collections view. This can be configured using property **lwm.diagnostics.top.n** (in **<probe-install-dir>/etc/dynamic.properties**). For example, if the value of this property is set to 10, only the top 10 collections will be reported. However you may see more than 10 collections since both the top N by size as well as the top N by growth are being reported.
- The resources reported can be controlled via regular expressions using the **property lwmd.resource.replace** (in **<probe-install-dir>/etc/dynamic.properties**). This property allows for pattern replacement of resources. The default regular expressions ensure two transformations.
  - Transformation 1 example:

    From: jar:file:/C:/bea/weblogic81/common/eval/pointbase/lib/pbclient44.jar!/com/pointbase/net/netJDBCConnection.class

    To: jar:file:/C:/bea/weblogic81/common/eval/pointbase/lib/pbclient44.jar
  - Transformation 2 example:

    From: file:/C:/bea/weblogic81/classes/com/pointbase/net/netJDBCConnection.class

    To: Exploded Classes
  - More transformations can be specified as desired.

For instructions on setting properties for the probes, see the *HP Diagnostics Installation and Configuration Guide*.

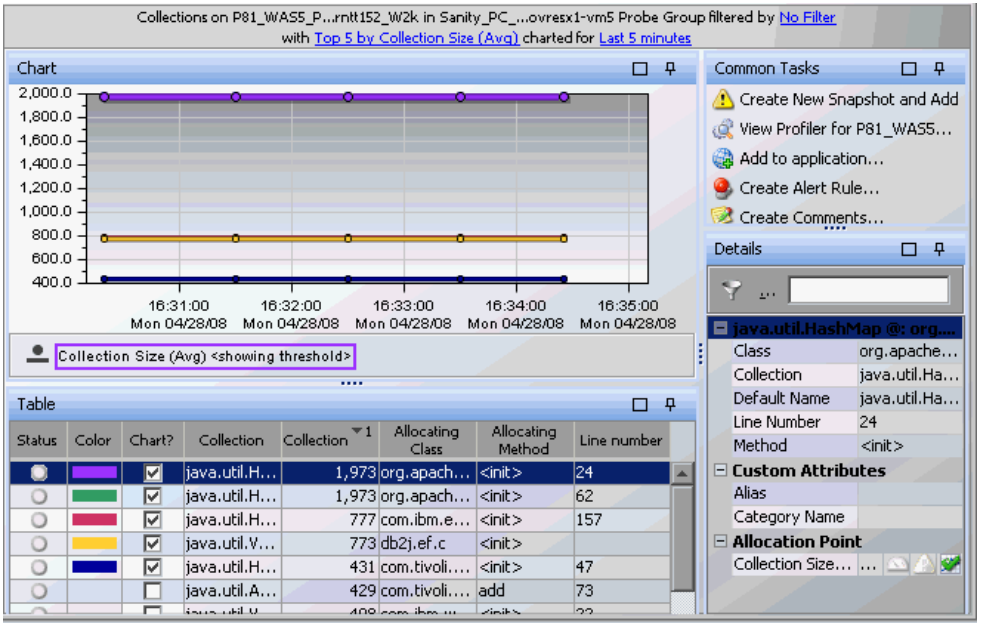
You can also use the Java profiler for more detailed memory use analysis. See Chapter 38, “Using the Java Diagnostics Profiler,” for more information.

Accessing the Collections View or Resources View

The Collections view and Resources view cannot be accessed directly from a view group on the View bar or from a standard dashboard view because collections and resources are always displayed in the context of a probe. You can access the Collections view and Resources view by drilling down from a probe in the Status view or in the Probes view. Right click on the probe and select **View Collections** or **View Resources**.

Description of the Collections View

By default, Diagnostics displays the top N collections by size as well as the top N collections by growth during the currently selected time frame. Diagnostics displays the size for each collection, as shown in the following example:



Collections View Graph

By default the collections graph shows the trend for the average collection size. Any other of the available metrics (min, max) can also be charted.

Graph Entity Table

The graph entity table in the Collections view lists all of the collections 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.

Common Tasks

The Common Tasks pane show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common task links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see “About the Common Tasks and Navigations Panes” on page 115.

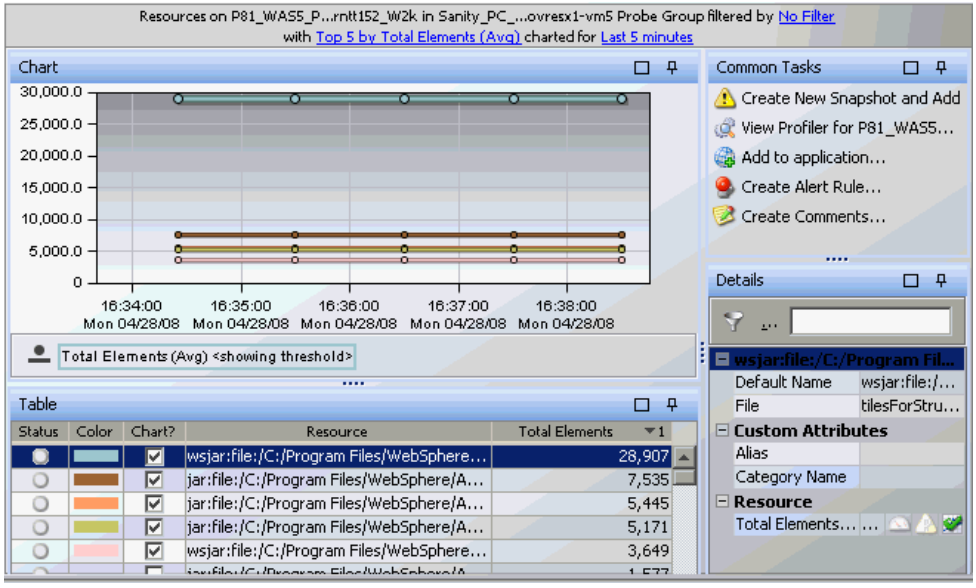
When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks pane so that they contain only **relevant** tasks that are appropriate for the selected entity.

Details Pane

The Details pane in the Collections view lists the metrics for the selected row of the graph entity table.

Description of the Resources View

By default, Diagnostics displays the resources that have the highest average number of elements during the currently selected time frame. Diagnostics displays the average number of elements for each resource, as shown in the following example:



Resources View Graph

By default, the Resources graph shows the trend for the Total Elements metric. Any other of the available metrics (largest collection size, number of collection instantiations) can also be chosen.

Graph Entity Table

The graph entity table in the Resources view lists all of the resources 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.

Common Tasks

The Common Tasks pane show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common task links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see "About the Common Tasks and Navigations Panes" on page 115.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks pane so that they contain only **relevant** tasks that are appropriate for the selected entity.

Details Pane

The Details pane in the Resources view lists the metrics for the selected row of the graph entity table.

Customizing the Collections View or Resources View

You can use the standard Diagnostics view controls to adjust the amount and type of data displayed in the Collections view and Resources. For more information, see Chapter 4, "Working with Diagnostics Data Displays."

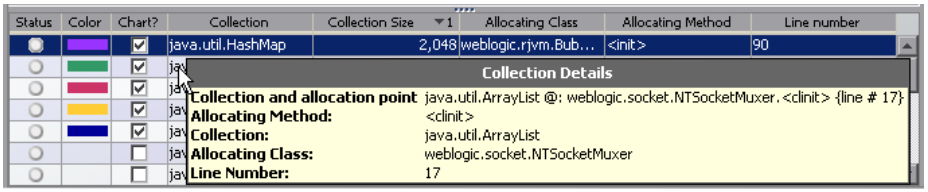
Interpreting the Collections View

Using the information displayed in the Collections view, you can get an immediate understanding of the largest collections in your system as well as where those collections are being allocated (class, method, line number).

If a collection object (any subclass of java.util.Collection) starts utilizing larger amounts of memory, the Collections view helps isolate the problem by showing which collection object is taking up the memory and where it is allocated.

Displaying Collection Details from the Graph Entity Table

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

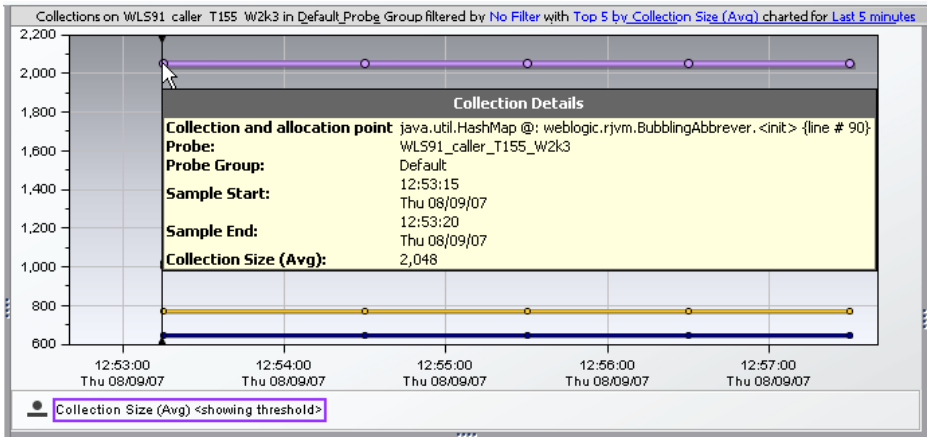


The **Collection Details** tooltip displays the following information:

- **Collection and allocation point.** This is made up of all four items below (concatenated).
- **Allocating Method.** The method where this collection was allocated.
- **Collection.** The name of the collection.
- **Allocating Class.** The class where this collection was allocated.
- **Line Number.** The line number where this collection was allocated.

Displaying Collection Details from Charted Metrics

You can view more information about a collection displayed in the graph by holding the mouse pointer over a line in the graph until the **Collection Details** tooltip is displayed, as shown in the following example:



The tooltip displays the following information:

- **Collection and allocation point.** This is made up of these four items: allocating class, allocating method, line number, collection.
- **Probe.** The name of the probe.
- **Probe Group.** The name of the probe group.
- **Sample Start.** The start time for the sample aggregation period represented by the selected data point.
- **Sample End.** The end time for the sample aggregation period represented by the selected data point.
- **Collection Size (Avg).** The average size of the collection.

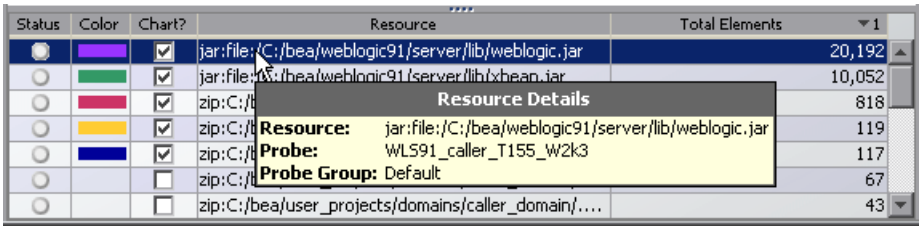
Interpreting the Resources View

The Resources view shows the resource that is allocating a collection object, whether it is part of a zip file, jar file, or other kinds of archives, or directly on the file system. You can see the total number of collection instantiations, largest collection size, and total memory occupied by collection objects.

Using the information displayed in the Resources view, you can get an immediate understanding of which resource is allocating the most collections. You also can see which resource holds the collection with the largest size, as well as the resource with the largest number of elements.

Displaying Resource Details from the Graph Entity Table

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

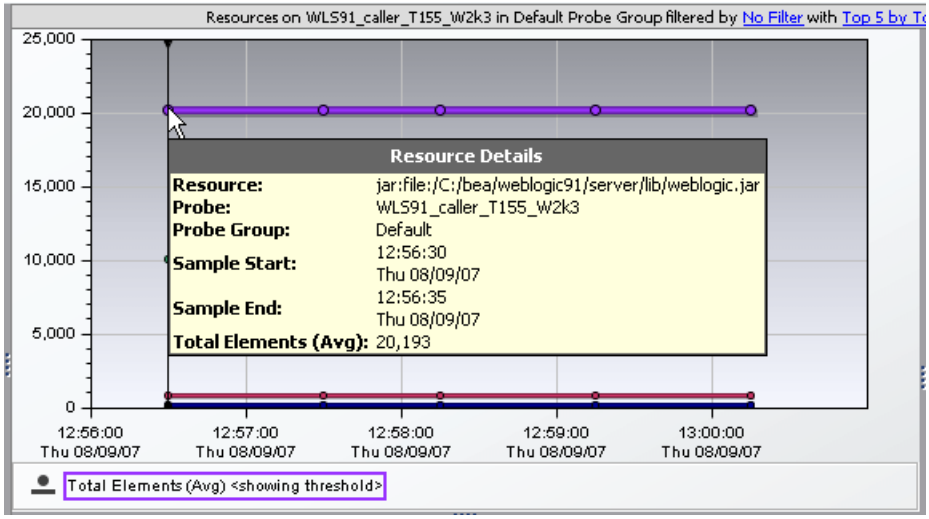


The **Resource Details** tooltip displays the following information:

- **Resource** The resource name.
- **Probe.** The name of the probe.
- **Probe Group.** The name of the probe group.

Displaying Resource Details from Charted Metrics

You can view more information about a resource displayed in the graph by holding the mouse pointer over a line in the graph until the **Resource Details** tooltip is displayed, as shown in the following example:



The tooltip displays the following information:

- **Resource.** The resource name.
- **Probe.** The name of the probe.
- **Probe Group.** The name of the probe group.
- **Sample Start.** The start time for the sample aggregation period represented by the selected data point.
- **Sample End.** The end time for the sample aggregation period represented by the selected data point.
- **Total Elements (Avg).** The average number of elements in this resource.

Part IV

Understanding Diagnostics Specialized Views

28

Portals Views

The **Portals** view group displays performance metrics for portlets (and other portal components such as books and pages) that are managed in the following portals:

- BEA WebLogic Portal
- IBM WebSphere Portal
- SAP NetWeaver Portal
- Oracle Portal

For detailed information on supported versions, see the Product Availability Matrix at http://support.openview.hp.com/sc/support_matrices.jsp or contact customer support.

This chapter includes:

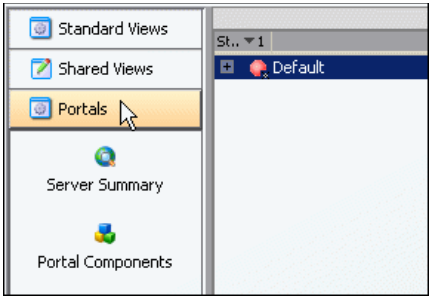
Accessing the Portals Views

You can access the Portals views from the View bar in the Diagnostics views.

To access the Portal views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **Portals** to open the Portals view group.



If the Portals view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **Portals** and click **OK**. The Portals view group is displayed in the View bar.

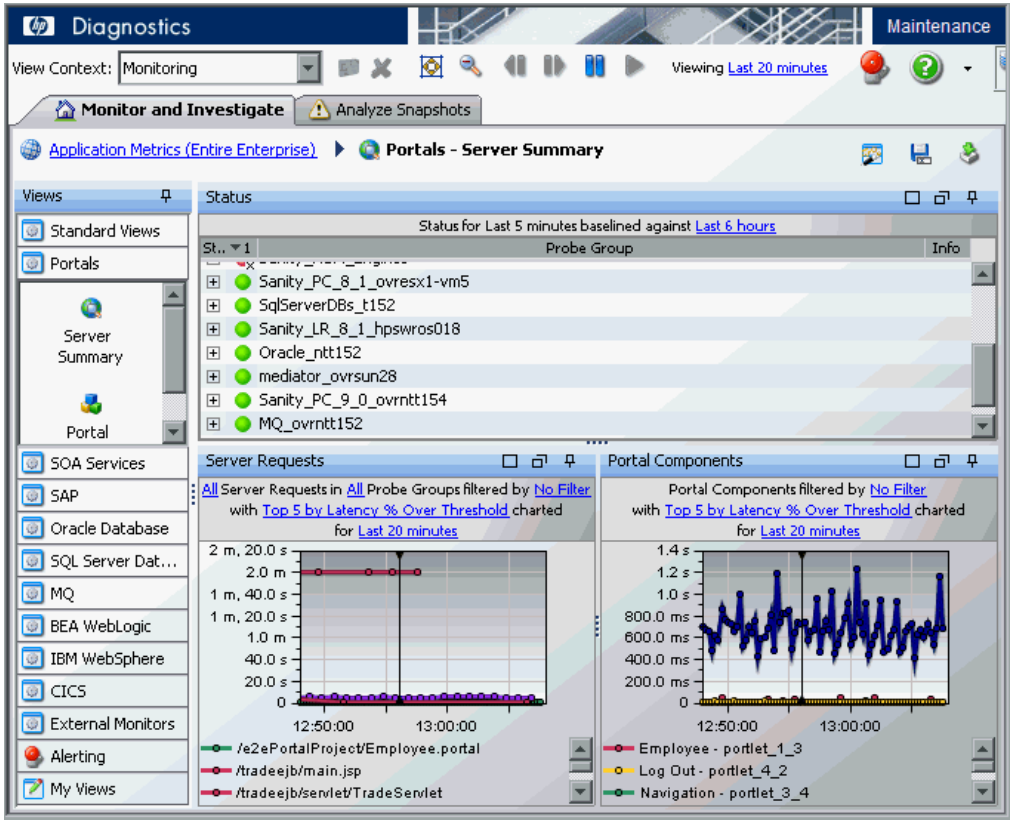
3 Select the appropriate view from the Portals view group.

Portal Server Summary View

The Portals view group contains a summary view that provides a high-level overview of portal related metrics. Depending on how you access Diagnostics (integrated with another HP product or in standalone mode), the Portal summary view is presented differently and includes different concise views.

If you are using Diagnostics in standalone mode, the Portals view group contains the Server Summary view and the Portal Components 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. All Diagnostics probe groups, probes, and hosts are displayed in this view, and not only those that are portal-related. For more information about the Status view, see Chapter 21, "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 portal-related. For more information about the Server Requests view, see "Aggregate Requests and Server Requests Views" on page 311.
- **Portal Components**. A monitoring version of the Portal Components view. For more information about the Portal Components view, see "Portal Components View" on page 405.



Integrated with Business Availability Center: Business Process Summary View

If you are using Diagnostics with 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 21, "Status View."
- **Transactions.** A monitoring version of the standard Diagnostics Transactions view. For more information about the Transactions view, see see "Transactions View" on page 349.
- **Portals Components**. A monitoring version of the Portal Components view. For more information about the Portal Components view, see "Portal Components View" on page 405.

Integrated with LoadRunner or Performance Center: Scenario Summary View

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 see "Transactions View" on page 349.
- **Server Requests.** A monitoring version of the standard Diagnostics Server Requests view. For more information about the Server Requests view, see "Aggregate Requests and Server Requests Views" on page 311.
- **Portal Components**. A monitoring version of the Portal Components view. For more information about the Portal Components view, see "Portal Components View" on page 405.
- **Load.** A monitoring version of the standard Diagnostics Load view. For more information about the Load view, see "Load View" on page 281.

Portal Components View

Important: The Portal Components view can be used only with a Diagnostics probe from version 7.0 or later.

In the Portal Components view you see performance metrics for portlets and other portal components such as pages and books. This view has the format of a detail layout view. For more information about views presented in detail layout, see Chapter 4, "Working with Diagnostics Data Displays."

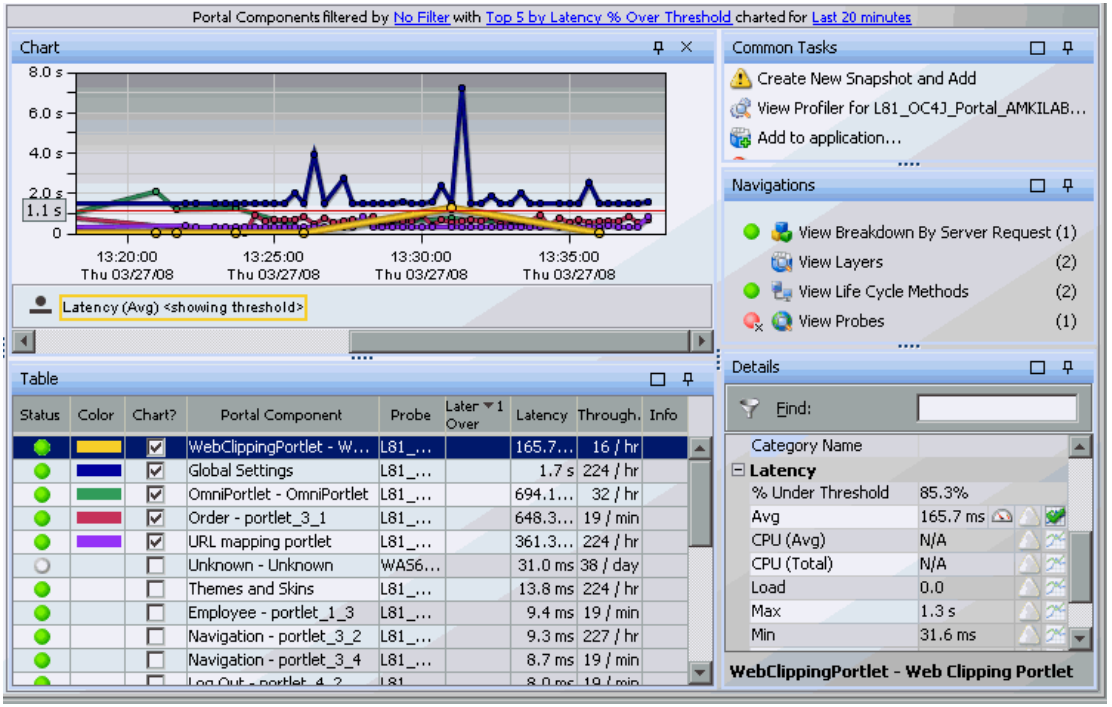
To access the Portal Components view:

Open the **Portals** view group (see "Accessing the Portals Views" on page 402) and click **Portal Components**.

Note: You can also drill down to the Portal Components view for a particular probe To do this, right click a probe and select **View Portal Components**. In this case, the Portal Components view displays the specific portlets that are being monitored by that probe.

Description of the Portal Components View

The Portal Components view displays average latency trends for portal components (portlets, pages, books, iViews) that are monitored by Diagnostics. The following image is an example of the Portal Components view:



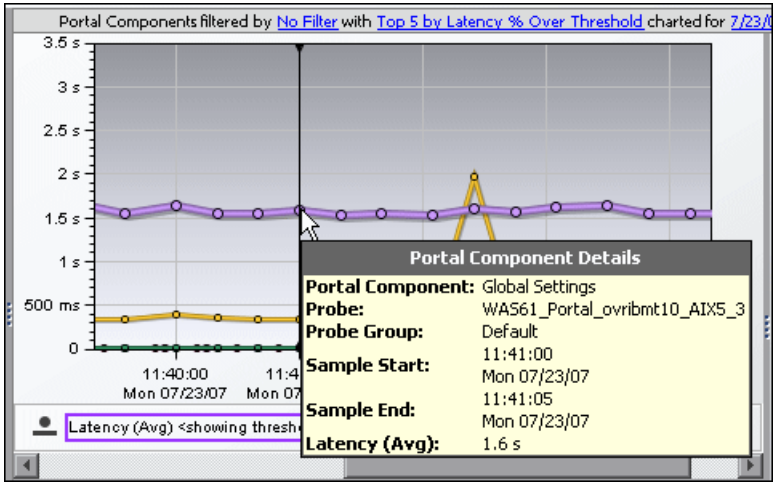
Graph

By default, the graph displays the top five portal components with the highest average latency percent over threshold during the previous five minutes.

Diagnostics displays the average latency for each portal component using a trend line. If a threshold has been set for the metric displayed (average latency) you will see a red horizontal line indicating the threshold value.

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.

You can view more information about a portal component by holding the mouse pointer over a node in a charted metric until a tooltip is displayed.



The Portal Component Details tooltip includes the names of the probe and probe group to which the portlet belongs, the average latency at that point, and the start and end period for which the average latency was collected.

Common Tasks and Navigations Pane

The Common Tasks and Navigations panes show actions that are relevant to the entity selected in the graph entity table. They provide suggested actions you can take to further analyze a performance problem and diagnose the cause. You can also access these common tasks and navigation links by right-clicking on a row in the graph entity table and selecting from the menu displayed. For more information see “About the Common Tasks and Navigations Panes” on page 115.

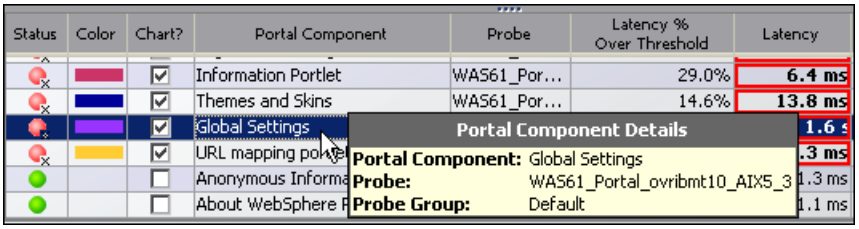
The navigation links listed also show the status of the related entities and their count, where this information is available.

When you select a row in the graph entity table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity.

Graph Entity Table

The graph entity table lists the portal components that pertain to the context shown in the breadcrumbs displayed at the top of the view. The metrics for each portlet that is reported in the table are aggregated and reported, based on the time period specified in the **Time Range** view filter.

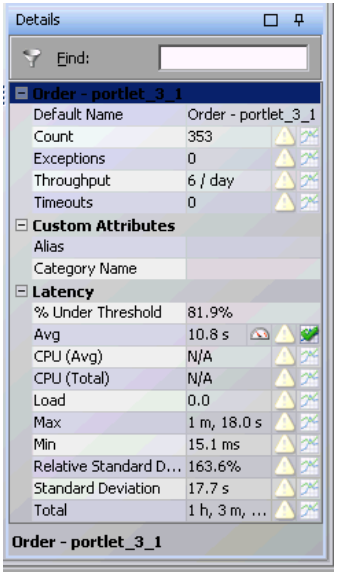
When you hold your mouse pointer over the portal component in the **Portal Components** column, you can view the names of the probe and probe group to which the portal component belongs.



Details Pane

The **Details pane** in the Portal Components view lists the metrics for the selected portal component (row in the graph entity table). You can use the Details Pane to set thresholds, alerts, add comments, and create incidents. For more information about the Details pane, see “Working with Metrics, Thresholds and the Details Pane” on page 123.

For an explanation of each of the metrics in the details pane, see “Diagnostics Metrics Descriptions” on page 674.



In the details pane, CPU (Avg) and CPU (Total) metrics will be shown as N/A if CPU time collection is not enabled for the relevant probe.

CPU Time for Portlets

The CPU time metrics are **CPU (Avg)** and **CPU (Total)**. If collection of CPU time metrics is disabled or not configured for methods, you will see N/A in the Details pane for these metrics.

If you want to have CPU time for portlets collected and displayed in the details pane, CPU time collection needs to be configured for portlets in the relevant probe, or configured for All Methods.

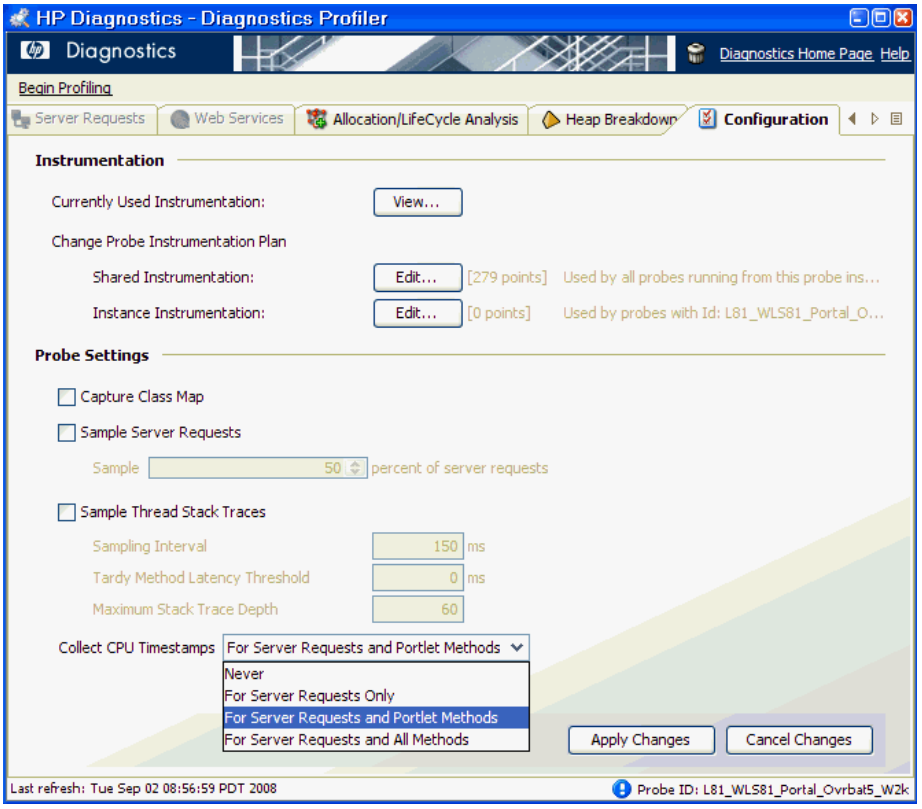
Important: In VMware, the CPU time metric is from the perspective of the guest operating system and is affected by the VMware virtual timer. See the VMware whitepaper on timekeeping at http://www.vmware.com/pdf/vmware_timekeeping.pdf and for more information, refer to the *HP Diagnostics Installation and Configuration Guide* section on Time Synchronization for Probes running on VMware.

Note: Use caution when configuring 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.

You can use either of the following methods to enable collection of CPU time for portlet lifecycle methods or for all methods. Your changes take effect immediately. There is no need to restart the Probe. See the *HP Diagnostics Installation and Configuration Guide* section on Configuring Collection of CPU Time Metrics.

- In the **dynamic.properties** file for the relevant probe set **cpu.timstamp.collection.method=3** to turn on CPU collections for portlet life cycle methods or **2 to turn on CPU collections** for All methods.

- Or, in the **Profiler** for the probe, select the **Configuration** tab (use the right arrow next to scroll to find this tab). The profiler does not need to be started to make this configuration change. In the Configuration screen set Collect CPU Timestamps to **For Server Requests and Portlet Methods** or **For Server Requests and All Methods** and apply changes.



Interpreting the Portal Components View

A portal component (portlet) is not a single instrumented entry point. Portal components consist of the top-level methods of the layer type **portlet**. The statistics for these methods are aggregated by name and presented as a single portal component.

The information displayed in the Portal Components view allows you to see a breakdown of Web application server activity by portlet. As a result, you can see where portlet time is spent and you can assess latency (response time) trends.

You can see which portlet, or which layer or life cycle method under the portlet, is contributing the most to a performance problem. You can also view the server requests that contributed to the latency performance of a particular portlet.

Selecting Related Views

From the Navigations pane in the Portal Components view or by right-clicking a Portal Component in the graph entity table you can drill down to the following views:

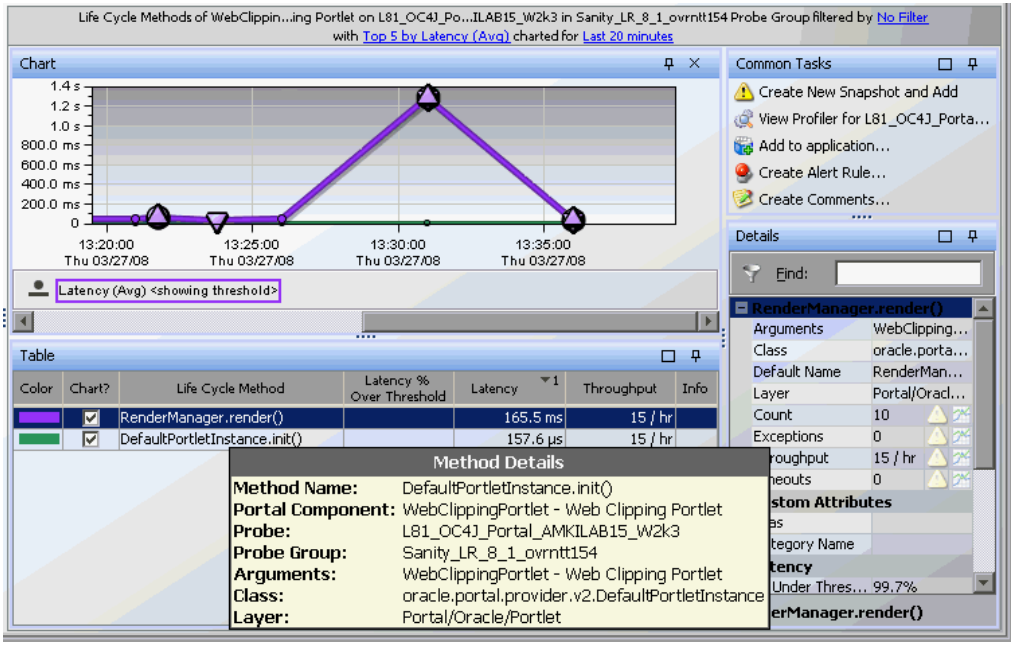
- View Life Cycle Methods
- View Breakdown by Server Requests
- Server Request Breakdown by Portal Component
- View Layers

You can drill down to a call profile view for a portlet (see "View Portlets in the Call Profile View" on page 413).

View Life Cycle Methods

You can drill down to the top-level life cycle methods for any portlet. To do this, right-click the relevant portlet in the graph entity table and select **View Life Cycle Methods**. The Life Cycle Methods view opens, displaying the top five life cycle methods with the highest average latency.

When you hold your mouse pointer over the method in the **Life Cycle Methods** column, you can view a tooltip displaying details about the selected method, including the name of the method and the associated class and layer.



The trend line includes small icons to indicate where the minimum, maximum and average instance trees were recorded. You can drill down from the graph in the Life Cycle Methods view to see the call profile for any of these instances.

Portlet Life Cycle instance trees are enabled by default. You can turn this off in the dispatcher.properties file with enable.portlet.method.trees=false.

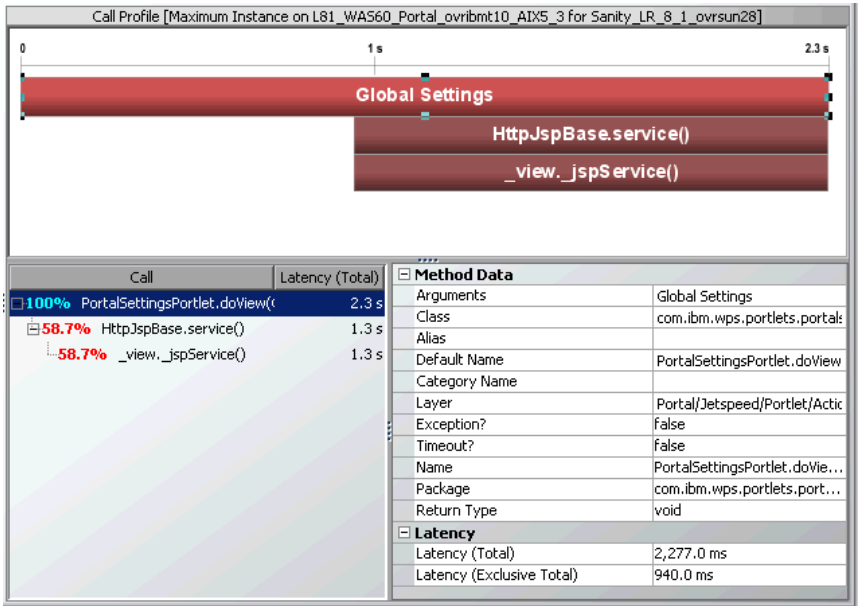
View Portlets in the Call Profile View

You can view the portlet in the Call Profile view as a method call argument. You access the Call Profile view by selecting an instance (Min, Max, Avg) on the trend line in the Life Cycle Methods view.

The Life Cycle Method call profile only shows the call tree from that method call down. It just shows that unit of work. To see the call tree for the whole server request, which may include other calls, other portlets, you would drill down to the portlet call profile from the Server Requests view. To access the Call Profile view from the Server Requests view, see “Drilling Down to an Instance Tree for a Server Request” on page 322.

Note: You can identify the originating server requests of a portlet by right-clicking the relevant portlet in the Portal Components view and selecting **View Breakdown by Server Request**. After identifying the originating server requests, you can locate them in the Server Requests view and then drill down from those server requests to the Call Profile view.

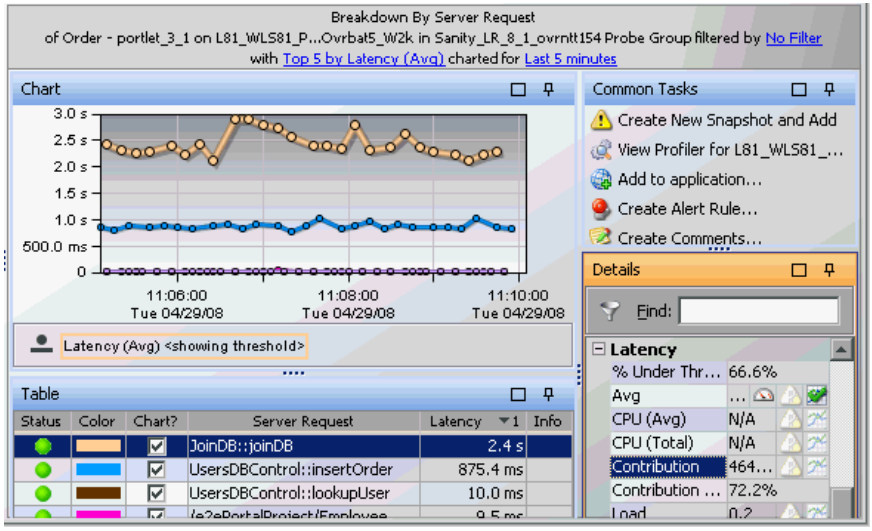
In the following example of an instance tree in the Call Profile view, a method call containing a portlet argument is displayed in the call tree table. The corresponding call box in the call profile graph is labelled by the name of the portlet (not the whole method). For more information about the Call Profile view, see Chapter 26, “Call Profile View.”. Data about the selected method call is displayed in the details pane.



View Breakdown by Server Requests

You can view a breakdown of the server requests that contributed to the latency performance of a particular portlet. For example, you may want to view the server requests breakdown to determine whether the behavior of a portlet varied greatly from one server request to another.

To view the server requests breakdown, right-click the relevant portlet in the graph entity table and select **View Breakdown by Server Request** or select the view in the Navigations pane. The Breakdown by Server Requests view opens, displaying the Server Requests that contributed to the portlet's latency performance.



The metrics for the server requests displayed in this view do not represent the whole server request. Instead, they represent the part of the server request that worked on processing the relevant portlet.

Under the **Latency** category in the Details pane you can view the contribution of the selected server request to the average latency of the portlet.

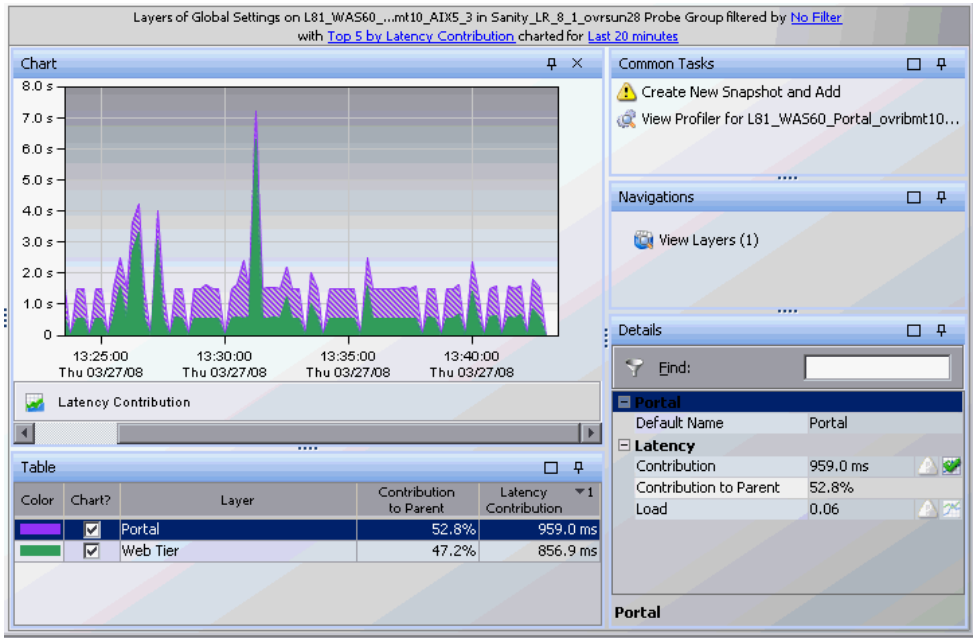
Note: In certain cases, WebSphere server request URIs are very long and can adversely affect usability. To replace URIs with more usable names or to automatically truncate the URIs, use the **uri.pattern.replace** property in the **< probe_install_dir>\etc\dynamic.properties** file.

Instructions for using this property are documented in the file. This feature should not be overused, as it can adversely affect performance.

Note: You can also drill down from an originating portal server request to a portlet breakdown for that server request by selecting the request in the **Server Requests** view and then selecting View Breakdown by Portal Components in the Navigations pane. See "Server Request Breakdown by Portal Component" on page 417.

View Layers

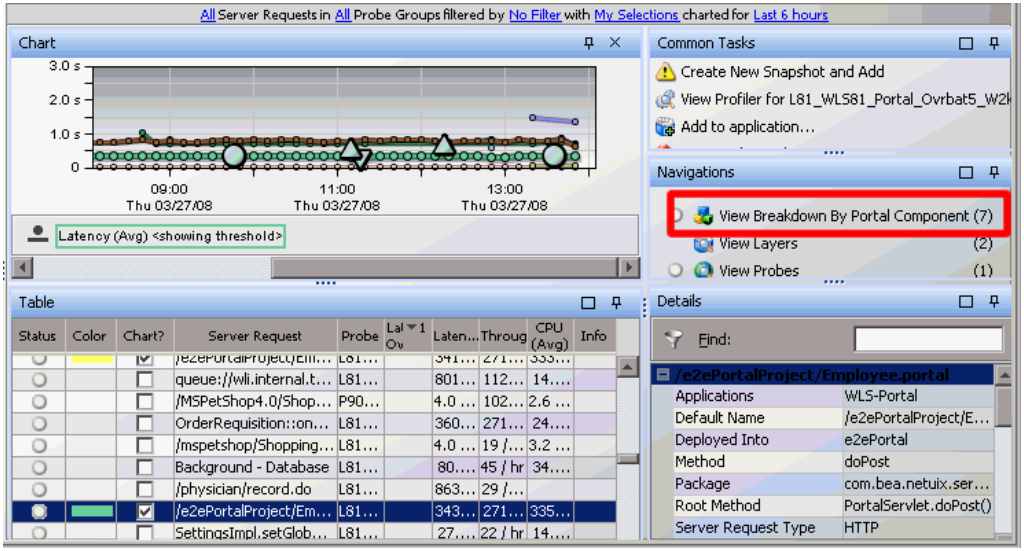
You can drill down to the layers that contribute to a portlet's latency performance. To do this, right-click the relevant portlet in the graph entity table and select **View Layers** or select this view in the Navigations pane. By default, Diagnostics displays the latency contribution for the five layers that have the highest percent contribution values during the currently selected time frame.

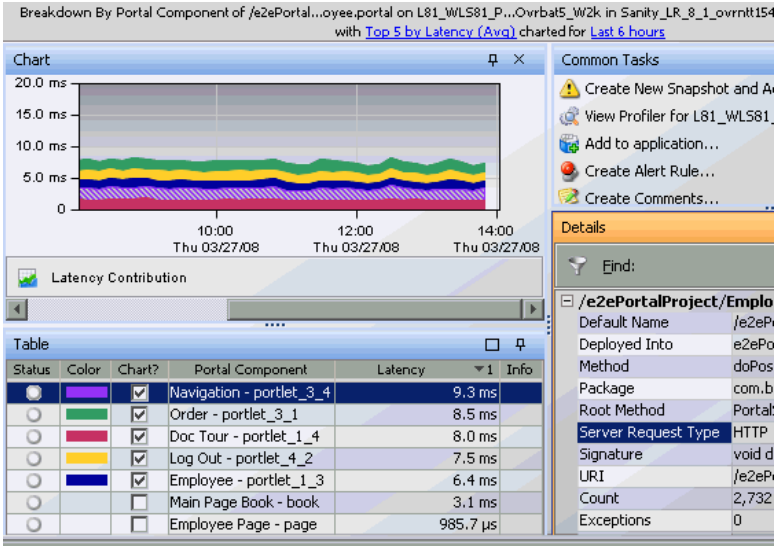


Diagnostics displays the latency contribution for each layer using a stacked area graph. For more information about working with the Layers view, see Chapter 25, "Layers View."

Server Request Breakdown by Portal Component

You can also drill down from an originating portal server request to a portlet breakdown for that server request by selecting the request in the **Server Requests** view and then selecting View Breakdown by Portal Components in the Navigations pane. This gives you a breakdown of the server request by portal component as shown below.





29

SOA Services Views

The **SOA Services** view group shows latency and other performance data for service oriented architectures.

Diagnostics automatically detects server requests that are Web services. Diagnostics aggregates Web service operations to Web services (by aggregating all Web service operations for a particular named Web service) to create a higher level aggregation of service performance. This data is used to provide a service topology showing which services connect to other services.

From a service you can drill-down to see the performance of its operations. From a service you can also see the status of its operations because any "red" status from an operation will be rolled-up to its services.

This chapter includes:

About SOA Services Monitoring

A SOA service (for example, Web service or REST service) is a software component that provides functionality via a set of operations accessible over the network using various protocols.

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.

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 operation. 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 operation.

Outbound and inbound web service calls that use JMS as the transport are monitored as Web services in Diagnostics. In this case the JMS target (queue name or topic_name) is used as the target for the web service calls. In the Diagnostics Outbound service calls view and in the service topology the remote target is displayed as queue_name/web_service_name or topic_name/web_service_name.

Note: Diagnostics monitors the following:

- SOAP over HTTP/HTTPS operations
- SOAP over JMS operations
- .NET Windows Communication Foundation (WCF) services are monitored as web services.
- In addition, the Diagnostics Java Probe monitors XML over HTTP/HTTPS operations when REST style services are configured to be monitored as Web services in Diagnostics. The .NET Probe does not support monitoring REST style services.

Key features for Web services monitoring provided by Diagnostics include:

- Creation of samples of Web services data that are sent to Business Availability Center when the two products are integrated. See the Chapter 44, "Diagnostics Data in Other HP Software Products" for details.
- The collection of performance data based on Web service operation and aggregation of the data into named services.
- Tracking of Web service consumers and breakdown views of consumer IDs by Web service operations and aggregated by Web Service (across all operations).
- Drill down to the call profiles for key instances includes details on SOAP faults and SOAP payloads captured on failures.
- Service Topology view providing information about the connection between Web services.

- Views for outbound call to Web services and outbound calls to Web service operations.

Important: For SOA customers, Business Availability Center 7.50 or later is required in order to take advantage of Diagnostics Web services enhancements (such as configurable Consumer ID) in the Business Availability Center integration.

Supported Web Service Platforms

Diagnostics supports Web services tracking for the following:

- BEA WebLogic
- IBM WebSphere
- JBOSS
- Microsoft .NET and .NET Windows Communication Foundation (WCF)
- SAP NetWeaver
- Apache Tomcat
- AquaLogic Service Bus (ALSB)
- IBM WebSphere Enterprise Service Bus
- TIBCO ActiveMatrix Enterprise Service Bus and BusinessWorks
- HP SOA Policy Enforcer broker can be monitored by the Diagnostics Java Probe so that proxy services are seen as Web services and Diagnostics can monitor the operations on these services. See "Monitoring of the SOA Policy Enforcer Broker" on page 458.

  For more information on supported versions and supported features for each version see the Product Availability Matrix at http://support.openview.hp.com/sc/support_matrices.jsp or contact HP Customer Support.

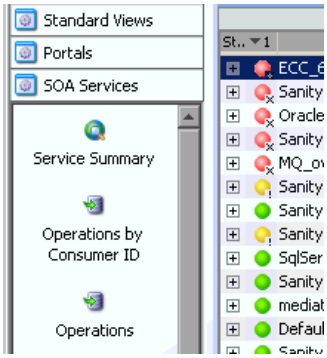
Accessing the SOA Services Views

You can access the SOA Services views from the View bar in the Diagnostics views.

To access the SOA Services views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **SOA Services** to open the SOA Services view group.



If the SOA Services view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **SOA Services** and click **OK**. The SOA Services view group is displayed in the view bar.

3 Select the appropriate view from the SOA Services view group.

Using the SOA Services Views

The SOA Services view group includes the following views:

- **Service Summary.** A dashboard that displays the graphs from each of the following views.
- **Services.** Shows statistics for the services in your application. Services are made up of server requests that are operations aggregated into a named service.

- **Operations.** Displays server requests that are Web service operations.
- **Service Topology.** Displays a topology view of monitored Web services and service connections.
- **Services by Consumer ID.** Displays a breakdown by consumer for the Services in your application.
- **Operations by Consumer ID.** Displays a breakdown by consumer for the server requests that are Web service operations.
- **Outbound Service Calls.** Shows statistics for outbound calls to Web services from your application.
- **Outbound Operations Calls.** Shows statistics for outbound calls to Web service operations from your application.

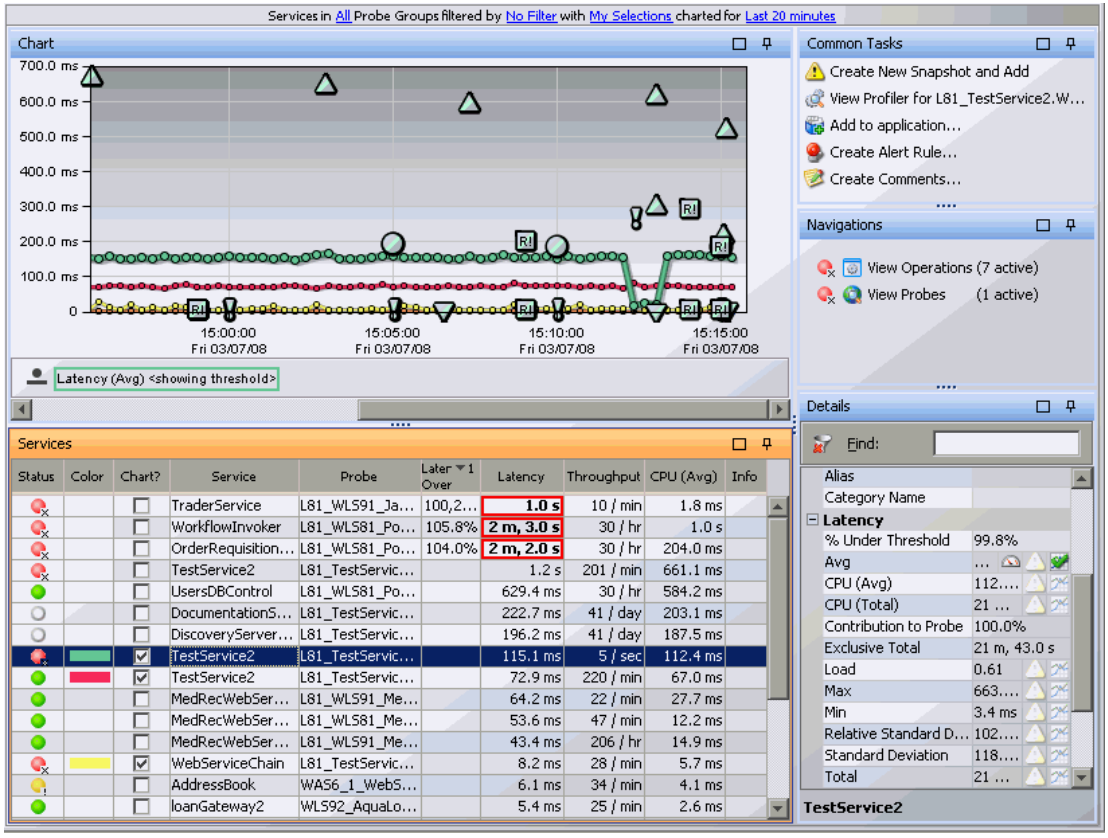
  The Web service name and/or operation may appear as **Unknown**.

Service Summary

The Service Summary view gives you a dashboard displaying the graphs from each other the other views in the SOA Services view group (except Service Topology). The graphs show top 5 latency over threshold. You can double-click on any of the graphs to drill into the related view.

Services

The Services view gives you detailed information for the services being monitored. Server requests that are Web service operations are aggregated by the service name as shown in the screen shot below.



You can see the status of each service in the graph entity table. From a

service you can also see the status of its operations because any "red" status from an operation will be rolled-up to its services. Threshold violations are highlighted.

You can also see which probe is monitoring the service and key performance metrics such as latency and throughput. The details pane gives you more metrics to help in diagnosing performance problems.

You can select any of the icons displayed in the graph to drill down to instance trees (minimum, maximum, average, exceptions, SOAP faults) that display the method calls and their latencies for a particular instance of a Web service. For more information, see “SOA Services Call Profiles” on page 441.

You can view information about SOAP faults and drill down to a call profile window for a particular SOAP fault. For more information, see “SOAP Fault Instances” on page 444.

You can also drill down to see the operations calls for a service by right-clicking the service in the services table and selecting **View Operations** or using the Navigations pane. And you can select **View Probes** to see related information from the probe for the service.

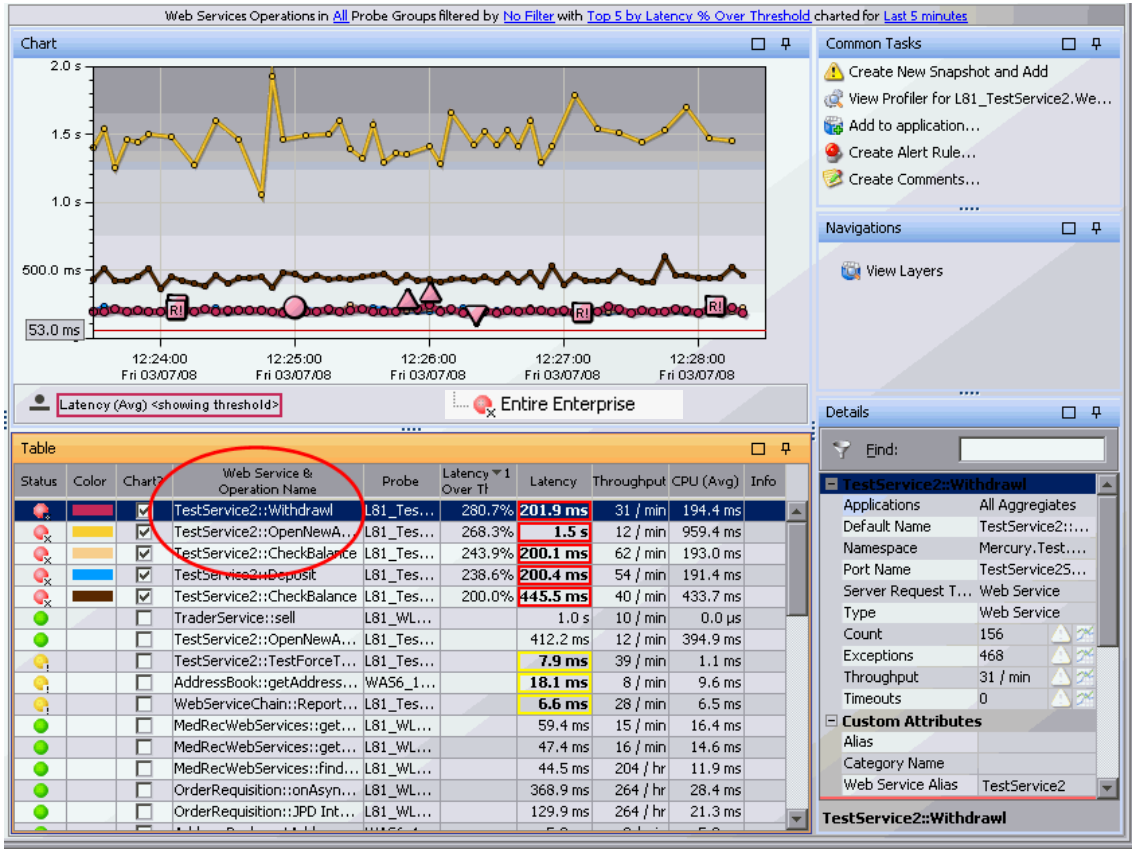
Operations

The Operations view displays all inbound Web service operations. Each unique Web service operation is displayed in the services table.

Operations are displayed in the **Web Service & Operation Name** column of the services table. They appear in the following format:
<Web-service-name>::<operation-name>.
For example, **TestService2::Withdrawal**.

If the same Web service operation was called by multiple consumers, the data from all these calls is aggregated and presented as a single Web service operation in the services table

.



You can select any of the icons displayed in the graph to drill down to instance trees (minimum, maximum, average, exceptions, SOAP faults) that display the method calls and their latencies for a particular instance of a Web service. For more information, see “SOA Services Call Profiles” on page 441.

You can view information about SOAP faults and drill down to a call profile window for a particular SOAP fault. For more information, see “SOAP Fault Instances” on page 444.

You can also drill down to see the operations calls for a service by right-clicking the service in the services table and selecting **View Operations** or using the Navigations pane. And you can select **View Probes** to see related information from the probe for the service.

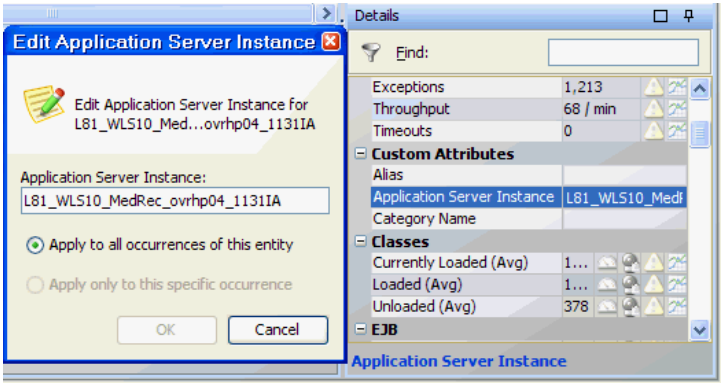
You can also drill down to the layers of each call by right-clicking the operation in the services table and selecting **View Layers** or using the Navigations pane.

Sometimes, an inbound server request or Web service call can trigger external outbound Web service calls to another Web service. In this case you can drill down to the outbound calls made from this server requests by selecting **View Outbound Calls**.

Specifying a Name for the Application Server Instance

Diagnostics creates samples with Web services performance data. The samples are sent to Business Availability Center when the two products are integrated. In order to differentiate between application servers when there are multiple application servers on the same host (potentially hosting the same Web service), Business Availability Center uses the **end_point identifier**. By default, the endpoint in the samples from Diagnostics is the probe name. However, you can set it to a more meaningful name in the **Probes** view of Diagnostics.

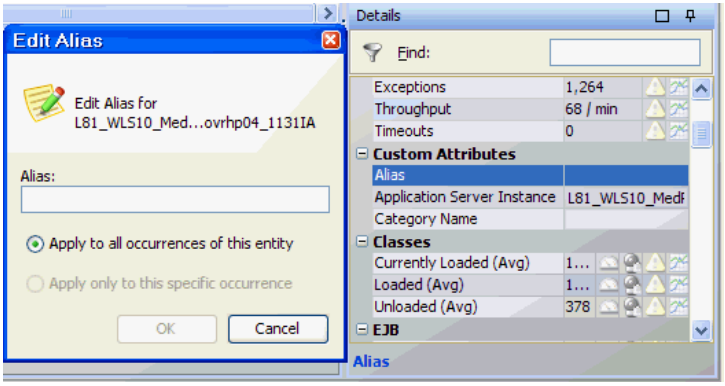
Navigate to the Probes view and select a probe that is monitoring your Web service. Select Application Server Instance in the details pane. Enter the name you want to use for the application server instance. This is the endpoint that is passed to Business Availability Center. See the Chapter 44, “Diagnostics Data in Other HP Software Products” for more information.



Changing the Display Name of a Web Service

Diagnostics and Business Availability Center use different ways of identifying the names of Web services and operations. As a result, there may be certain cases where Diagnostics and Business Availability Center identify different names for the same Web service and operation.

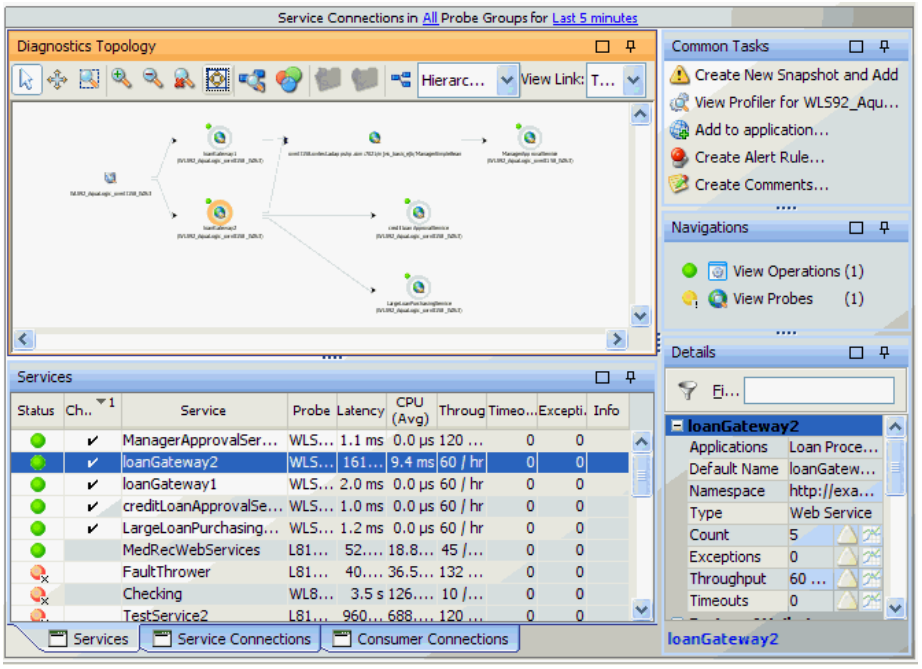
In such cases, you can change the Web Service and operations names as displayed in Diagnostics to match the names used in Business Availability Center. You can use the **Alias** field in the Details pane to change the Service name and the service::operation name. For detailed instructions about using Alias, see “Setting Custom Attributes Values” on page 127.



Service Topology

The Service Topology view provides a visualization of how Web services that communicate with each other are connected together. This is particularly useful in large complex Web service environments.

You can drill from Business Availability Center into this view in the Diagnostics UI. See the Chapter 44, “Diagnostics Data in Other HP Software Products” for more information.



Topology Diagram

The topology view displays services (shown as icons with a ’halo’ around them) and service/consumer connections. You can select an entity in the table to view its topology in the diagram. The selected entity is highlighted in the diagram: services are highlighted with a yellow halo, connections are highlighted with a striped line.

The controls in the topology view are similar to those in other topology views, see Chapter 6, “Working with the Topology Views” for details.

Note that the Service Topology view does not provide the ability to group by host and probe group.

A probe icon is displayed in the diagram when you select a service arc that originates from a probe. This happens when the call to a Web service originates from an instrumented application that is itself not a web service. A probe icon is displayed so you have some information about where the call originated from.

Services Table

There are three tabs in the Services Table:

- The **Services** tab displays information for each service including status, performance metrics like latency and throughput, and timeouts and exceptions.
- The **Service Connections** tab provides information on how services are 'stitched' together. You can see the origin and destination of the calls to and from web services (inbound and outbound) as well as status, latency and throughput.

  The Origin columns for JMS web service calls will show the queue_name or topic_name/web_ service_name. For example:

  queue://MedRec-jms!weblogic.wsee.reliability.wseeMedRecDestinationQueue/JMSTransportService

- The **Consumer Connections** tab provides information on service calls from the consumer node. The consumer node represents calls to a web service coming from a non-instrumented source (not monitored by a probe). These calls are grouped together and the Consumer Connections displays allows you to identify specific consumers calling the web service in the connection.

Common Tasks and Navigations

When you select a row in the services table, Diagnostics updates the right-click menu items and the Common Tasks and Navigations panes so that they contain only **relevant** tasks and navigations that are appropriate for the selected entity. The navigation links listed also show the status of the related entities and their count, where this information is available. For more information see "About the Common Tasks and Navigations Panes" on page 115.

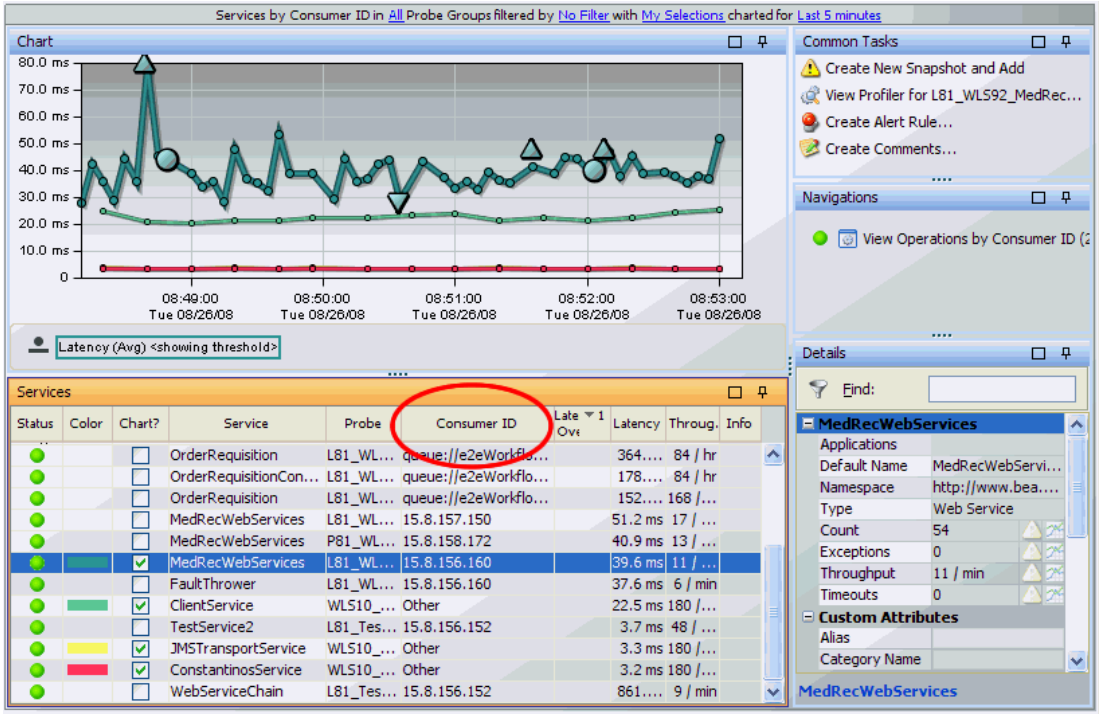
When you select a service, you can also drill down to see the operations calls by right-clicking the service in the services table and selecting **View Operations** or using the Navigations pane. And you can select **View Probes** to see related information from the probe for the service.

Details

Performance metrics and other information for the selected entity are displayed in the Details pane.

Services by Consumer ID

The Services by Consumer ID view shows a breakdown by consumer for a service. This allows you to see who is making a Web service request or where the traffic is coming from. You can identify who is impacted by performance problems on a particular service and see differences in performance by consumer.



About Consumer IDs

The consumer represents calls to a web service coming from a non-instrumented source (not monitored by a probe). These calls are grouped together in the service topology under the consumer node. You can configure consumer IDs to identify specific consumers.

Important: For SOA customers, Business Availability Center 7.50 or later is required in order to take advantage of Diagnostics Web services enhancements (such as configurable Consumer ID) in the Business Availability Center integration.

Consumer ID is an arbitrary string that can be configured from any of the following:

- IP address or IP range.
- HTTP Header.
- SOAP request Headers, Body or Payload. SOAP payload is the entire SOAP envelope.
- JMS queue name (or topic name) and JMS message header or JMS message property for SOAP over JMS web services.

IP address is used as the default consumer ID for SOAP over HTTP/S web services and inbound queue name (or topic name) is used as the default consumer ID for SOAP over JMS web services.

You can configure what you want in the Consumer ID field using probe configuration files. Note though that there can be a performance impact depending on what you configure for consumer ID.

You can also configure the number of consumer IDs that are monitored and displayed per probe. The default is five consumer IDs per probe. Assume for example that the probe was configured to display five unique consumer IDs. In this case, if there are more than five consumer IDs, the first five are displayed on separate rows in the graph entity table. The remaining consumer IDs are aggregated into a single row in the table with the consumer ID label **other**.

Configuration of consumer IDs is done on the .NET Probe and the Java Probe. See the *HP Diagnostics Installation and Configuration Guide* for details.

Important: Defining consumer ID based on SOAP header, envelope or body requires the Diagnostics SOAP message handler. For some application servers, special instrumentation is provided in Diagnostics to automatically load the Diagnostics SOAP message handler.

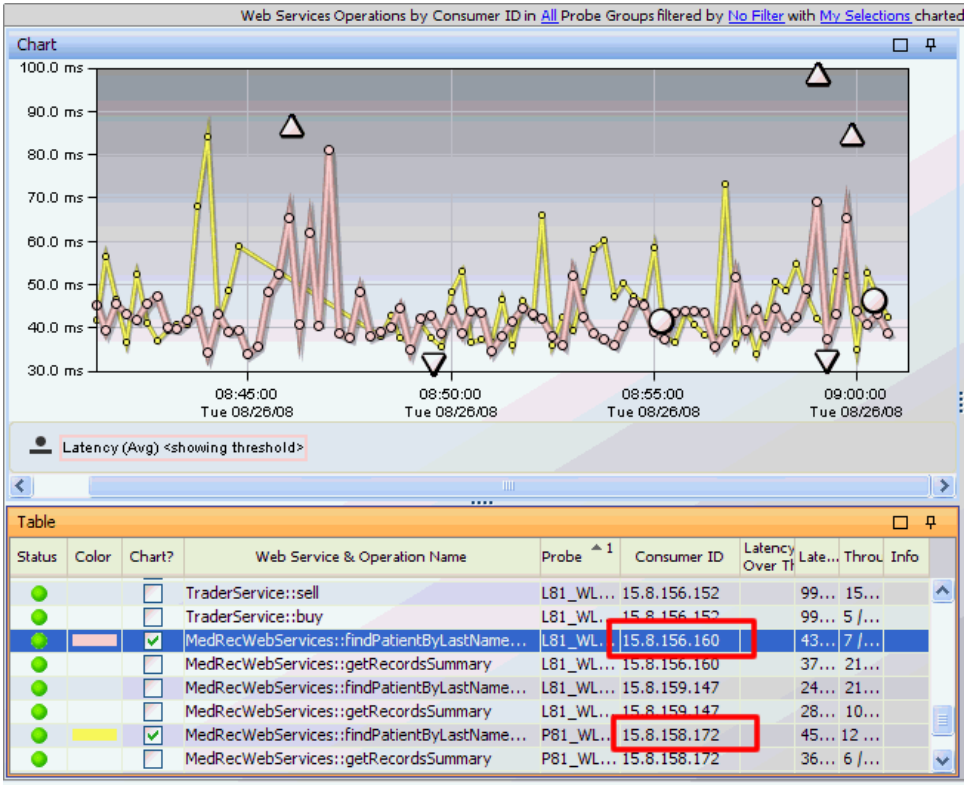
However, some manual configuration is required for WebSphere 5.1 JAX-RPC and Oracle 10g JAX-RPC, dee the *HP Diagnostics Installation and Configuration Guide* for details on configuring the SOAP message handler as well as configuring the probes to capture SOAP fault data.

In addition, the Diagnostics SOAP message handler is not available for all application servers nor is custom instrumentation available to capture SOAP faults or consumer IDs from SOAP payloads, and therefore this feature is not available on all versions of all application servers. For the most recent information on Diagnostics SOAP message handler support refer to the Diagnostics Product Availability Matrix at http://support.openview.hp.com/sc/support_matrices.jsp.

Operations by Consumer ID

The Operations by Consumer ID view displays inbound Web service operation calls according to the consumer ID from which they originated.

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



In the above example of a Diagnostics screen, the Web service operation, **MedRecWebServices::findPatientByLastNameWild** is called from two different Consumer IDs.

You can configure what you want in the Consumer ID field and the number of Consumer IDs to be monitored using property files (see “About Consumer IDs” on page 433). Configuration of Consumer IDs is done differently for the .NET Probe and the Java Probe. See the *HP Diagnostics Installation and Configuration Guide* for details.

Note: Diagnostics creates samples for each consumer ID that calls a Web service. These samples are sent to Business Availability Center when the two products are integrated. See the Chapter 44, “Diagnostics Data in Other HP Software Products”)

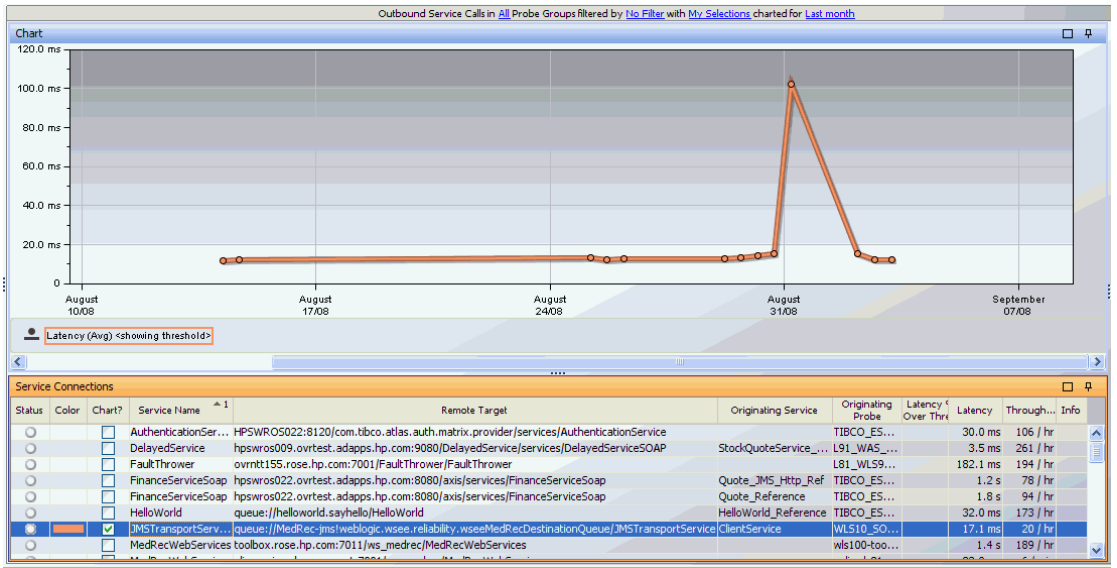
Increasing the number of consumer IDs captured by Diagnostics will therefore increase the amount of data sent to, and processed by, Business Availability Center. This will also increase the amount of CPU/Memory/Disk space used by the Diagnostics Servers and probes.

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.

Outbound Service Calls

Information about the outbound service calls made from your monitored environment is displayed in the Outbound Service Calls view.



The Outbound Service Calls graph entity table includes the following data:

- **Service Name.** The name of the service being called.
- **Remote Target.** For SOAP over HTTP/S web services, the hostname and port number of the Web service that is being called (displayed as <**host:port/URI**>).

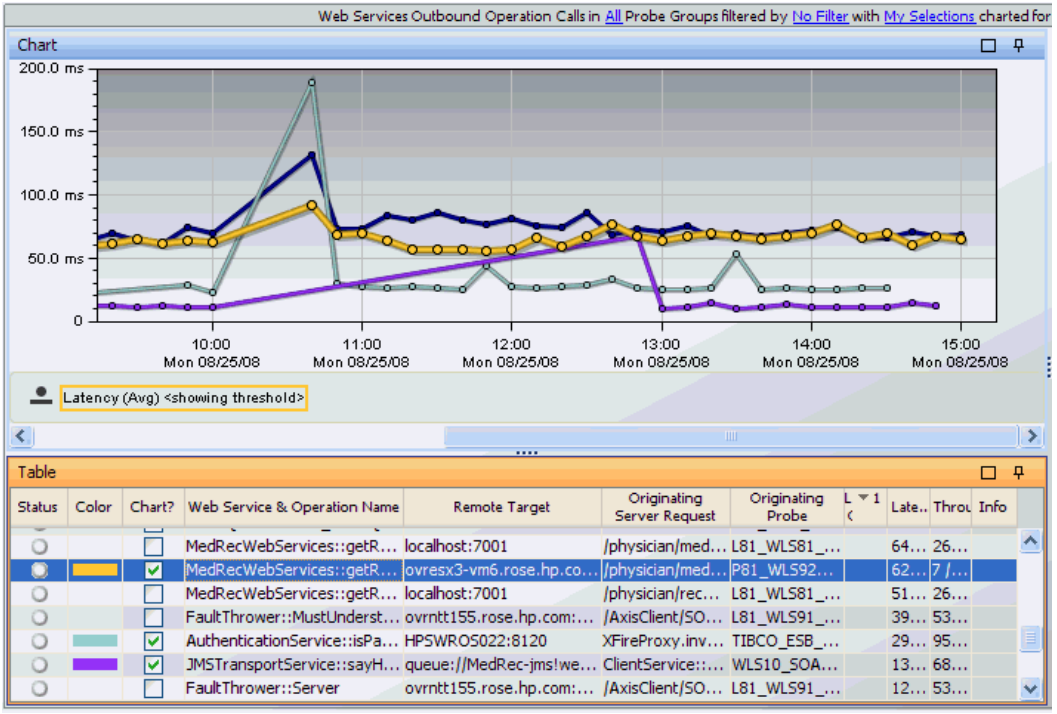
  For SOAP over JMS web services, the queue name (or topic name) of the web service being called (**queue://<queue name/web service name**>). Queue name comes from the outbound JMS information and web service name comes from the outbound WS information.

For JMS outbound calls it is possible to see a blank remote target. Some Application Server vendor's internal processes use some of the same instrumented methods that Diagnostics use for the remote target of a web services. Since these calls are internal to the Application Server vendor they are not actually web service calls so the remote target will be blank.

- **Originating Service.** The service making the call. Originating service will be blank when the call was made from an instrumented application that was not itself a Web service.
- **Originating Probe.** The probe from which this Web service was called.

Outbound Operations Calls

Information about the outbound operations calls made from within your monitored environment is displayed in the Outbound Operations Calls view.



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

The Outbound Operations Calls graph entity table includes the following columns:

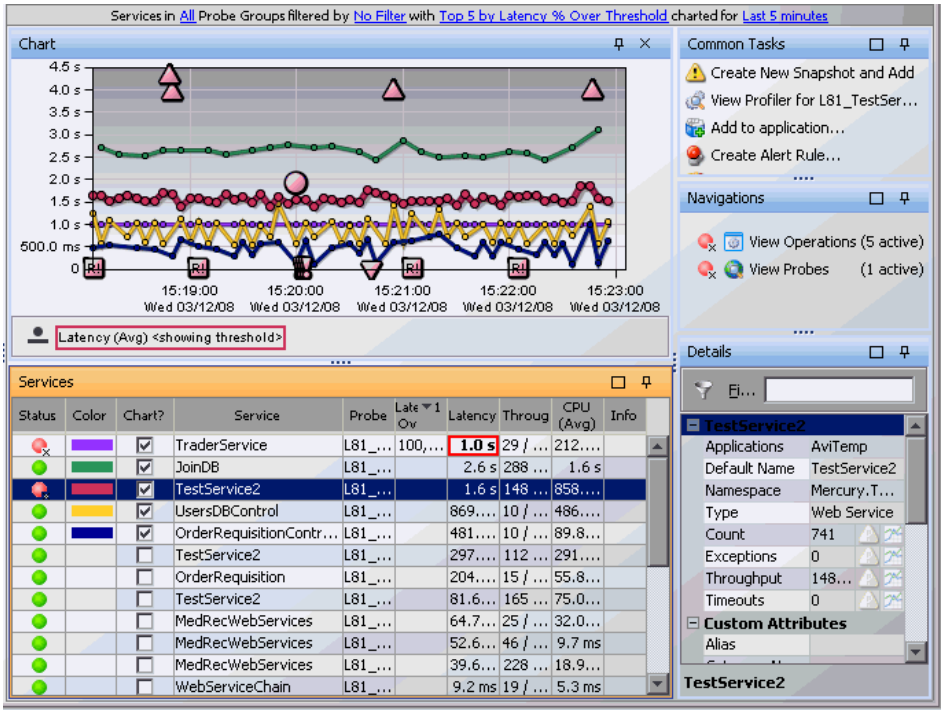
- **Web Service & Operation Name.** The name of the operation that is being called. It is displayed in the following format: **<Web-service-name>::<operation-name>.** For example, **CurrencyConversionService::ConvertToNum**.
- **Remote Target.** For SOAP over HTTP/S web services, the hostname and port number of the operation being called (displayed as **<host:port>**). For SOAP over JMS web services, the queue name or topic name of the operation being called (displayed as **queue://<queue name>**).
- **Originating Server Request.** The server request from which this operation was called.

➤ **Originating Probe.** The probe from which this operation was called.

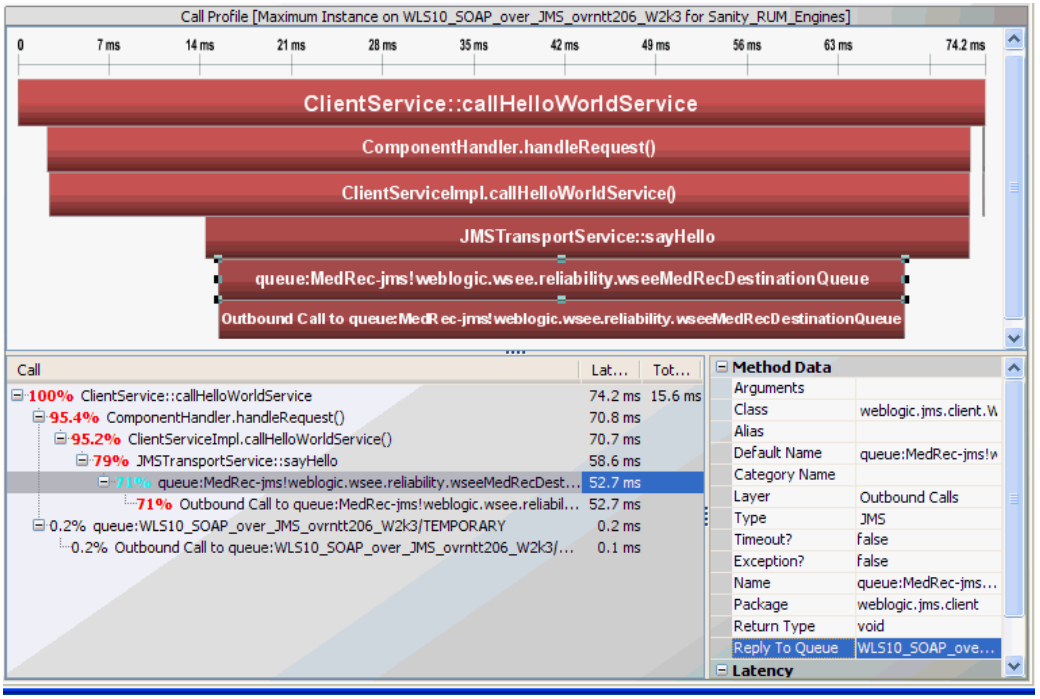
You can drill down to the originating server request displayed in the Server Requests view by right-clicking the outbound operations call in the graph entity table and selecting **View Server Requests**.

SOA Services Call Profiles

Instance trees for the most interesting instances of services and operations are saved. Instance trees for the minimum, maximum and average instances are saved along with instance trees for exceptions, SOAP faults and cross VM correlations. You can select the appropriate marker to drill down to the Call Profile view for each of these types of instances.



An example of a Call Profile view for a service is shown below.

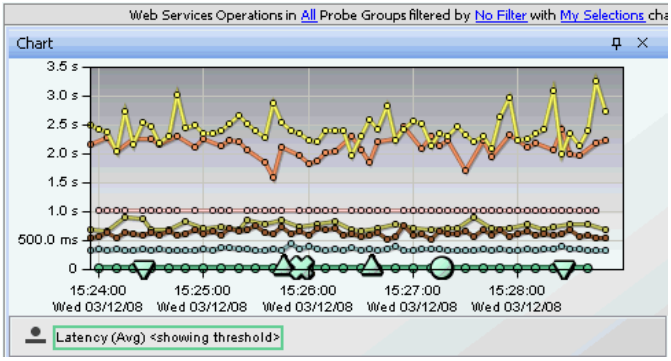


For outbound JMS calls, you will see the **Reply To Queue** name in the method data of the call profile (if the Reply To queue name is present on the outbound call). If the Reply To queue name is a temporary queue it will be renamed accordingly. You can use the Reply To queue name to help differentiate between producer and consumer on a JMS instance tree.

Cross VM Instances

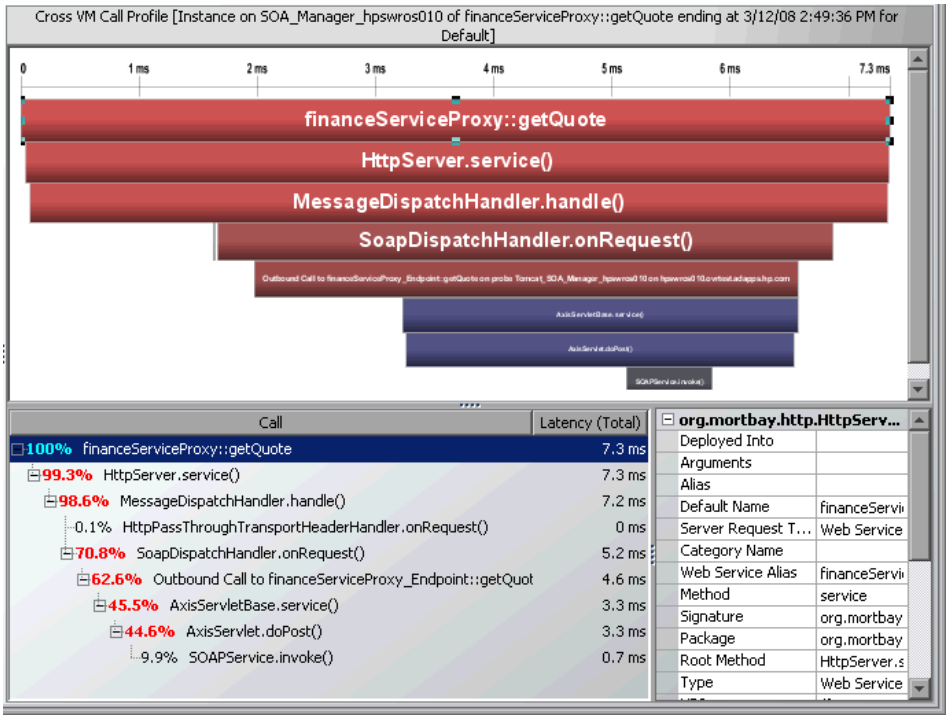
If both the outbound and inbound Web service calls for a particular Web service are monitored in your environment, you can view the correlation between outbound and inbound calls in an cross VM instance tree in the Call Profile view. You can view Web service correlations across multiple VMs and across different platforms, such as Java and .NET.

Click the X on the graph in the Operations view containing the outbound Web service call.



The Call Profile view opens, displaying a correlation instance tree. In the following example the call displayed at the top of the diagram and table contains an outbound Web service call and the inbound call for the same Web service is displayed in blue. Note that instance trees are not shown for the Outbound Operation Calls view or Outbound Service Calls views.

.



SOAP Fault Instances

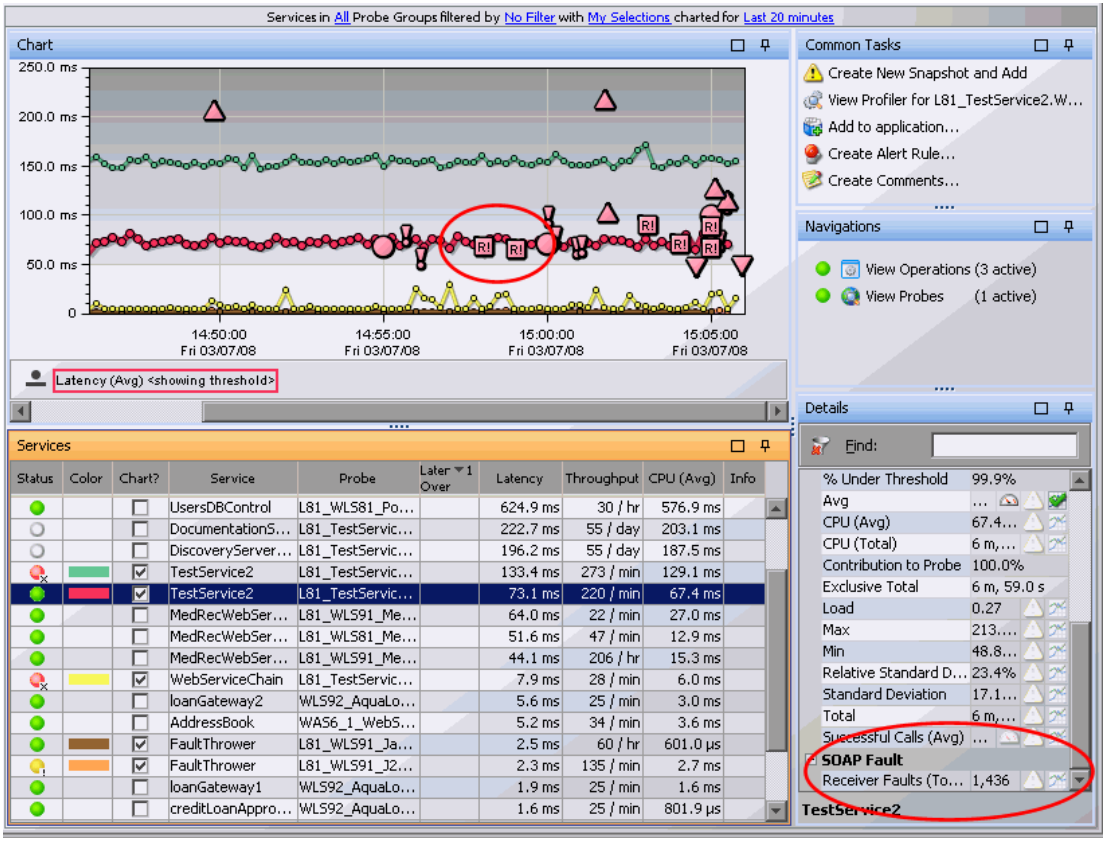
SOAP is an XML-based messaging protocol used in the exchange of Web services over a network. A SOAP fault is used to carry error information within a SOAP message.

Diagnostics captures SOAP faults and saves instance trees for the faults. By default, up to two SOAP faults per time period are captured. This can be configured using the probe configuration files (see the *HP Diagnostics Installation and Configuration Guide* for details on configuring the probes to capture SOAP fault data).

Note: SOAP faults are captured on both SOAP over HTTP and SOAP over JMS web services since the mechanism used to capture these faults is transport independent.

Diagnostics classifies SOAP faults into different types, based on their SOAP fault codes. Captured SOAP faults are represented in the SOA Services views by a unique icon on the graph. You can also see the number of SOAP faults in the details pane.

You may need to turn off the filter in the details pane to see the SOAP fault count.



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.

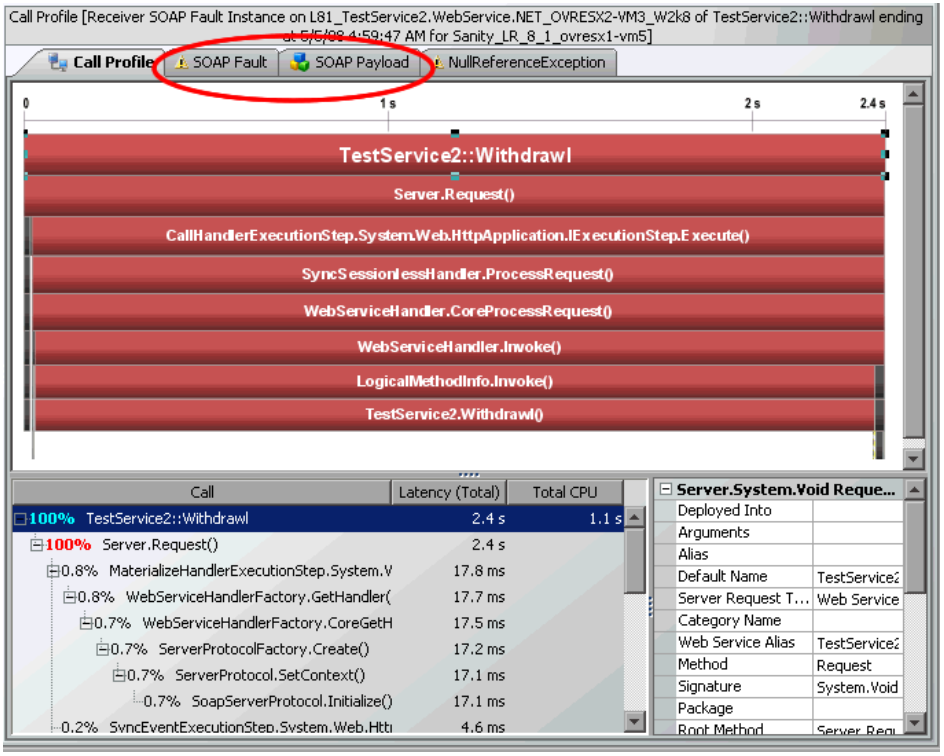
Types of SOAP Faults

The following table describes each type of SOAP fault and indicates the icon used to represent it in the SOA Services - Operations 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. This includes the Invalid Operation SOAP faults. 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.

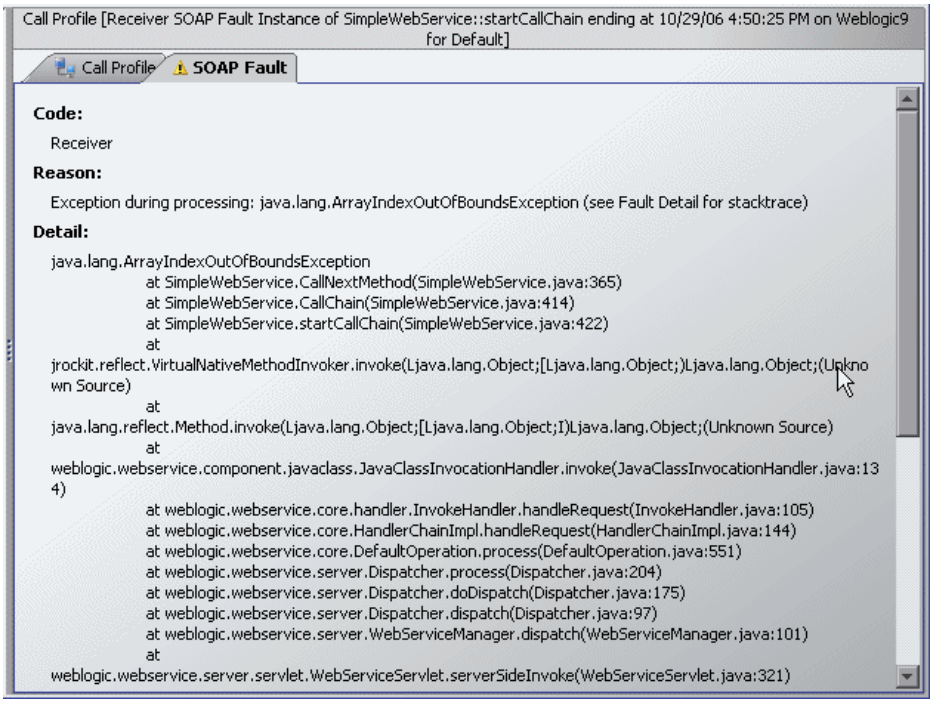
SOAP Fault Call Profile

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.

Payload on SOAP Faults

If a SOAP fault is detected, the SOAP payload is added to the SOAP fault data being collected, (for all SOAP faults except the Invalid Operations SOAP Fault, or as configured). You can view the payload information as part of the SOAP fault instance tree.

Payload capture on SOAP faults is enabled by default. But you can configure this for trimming.

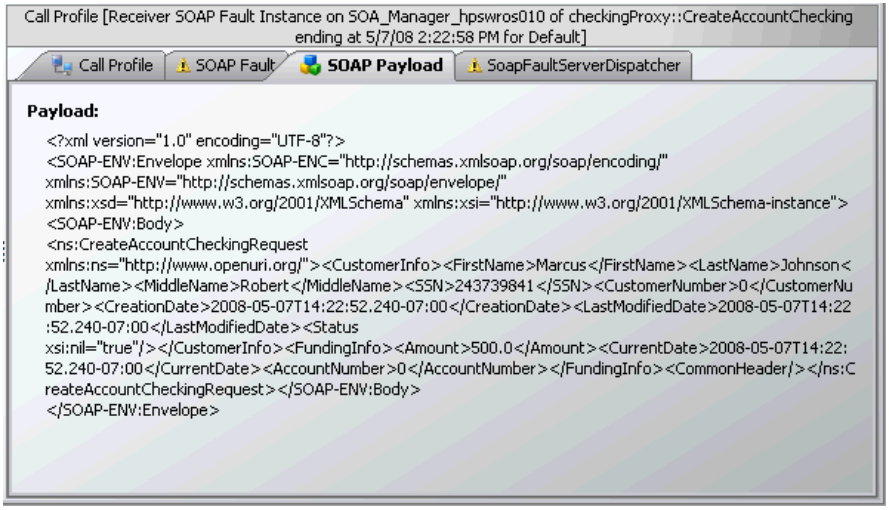
Important: Capturing SOAP payload requires the Diagnostics SOAP message handler. For some application servers, special instrumentation is provided in Diagnostics to automatically load the Diagnostics SOAP message handler.

However, some manual configuration is required for WebSphere 5.1 JAX-RPC and Oracle 10g JAX-RPC, see the *HP Diagnostics Installation and Configuration Guide* for details on configuring the SOAP message handler and configuring probes to capture SOAP fault data.

In addition, the Diagnostics SOAP message handler is not available for all application servers nor is custom instrumentation available to capture SOAP faults or consumer IDs from SOAP payloads, and therefore this feature is not available on all versions of all application servers. For the most recent information on Diagnostics SOAP message handler support refer to the Diagnostics Product Availability Matrix at http://support.openview.hp.com/sc/support_matrices.jsp.

Note: SOAP payload capture on SOAP faults is for both SOAP over HTTP and SOAP over JMS web services since the mechanism used to capture the payload is transport independent.

The SOAP Payload tab in the Call profile window includes the SOAP request information (data that the client sent).



Monitoring of REST Style Services

If you have Web services that are implemented with the REST style technology these will only be monitored by the Java Probe as server requests unless you configure Diagnostics to monitor them as Web services. For a Java Probe (not supported for the .NET probe) you define which of these REST services (HTTP server requests) you want monitored as Web services. This is done using the **rest.properties** file. See the *HP Diagnostics Installation and Configuration Guide* for details and examples.

Note: Unlike SOAP based Web services which are not latency trimmed, REST services are latency trimmed by default. This can be changed by setting **minimum.fragment.latency = 0ms** in the **dispatcher.properties** file for the probe.

Monitoring Web Services in an Enterprise Service Bus Environment

In some environments an Enterprise Service Bus (ESB) lies between business applications and enables communication among them. In an enterprise architecture making use of an ESB, a Web service application will communicate via the bus, which acts as a message broker between applications. The primary advantage of such an approach is that it reduces the number of point-to-point connections required to allow Web service applications to communicate.

Enterprise Service Bus software is open standards-based distributed synchronous or asynchronous messaging middleware that provides secure interoperability between enterprise applications via XML, Web services interfaces and standardized rules-based routing of documents. It can handle the incoming Web service calls and process them for validation, extraction, branching and routing through different sets of internal proxy and external business services.

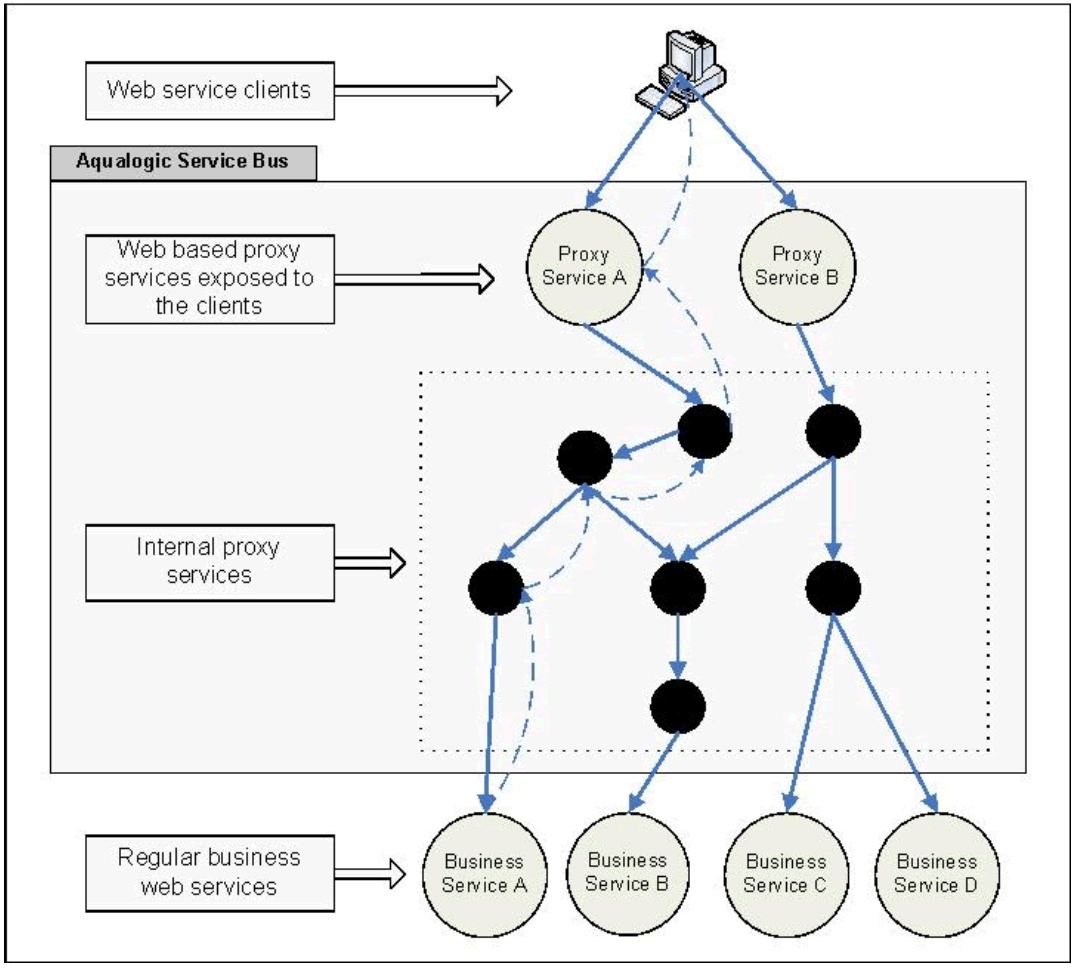
Diagnostics allows you to monitor Web services in an Enterprise Service Bus environment by providing Java Probe instrumentation for BEA AquaLogic Service Bus, IBM WebSphere Enterprise Service Bus, TIBCO ActiveMatrix Service Bus and HP SOA Policy Enforcer broker.

BEA AquaLogic Service Bus Support

The Java Probe includes instrumentation points for capturing the inbound proxy services and the outbound call to HTTP and JMS business services running in a BEA AquaLogic Service Bus (ALSB) environment. AquaLogic Service Bus is an enterprise service bus (ESB) for service-oriented integration, managing service interactions, and brokering messages across several heterogeneous IT environments.

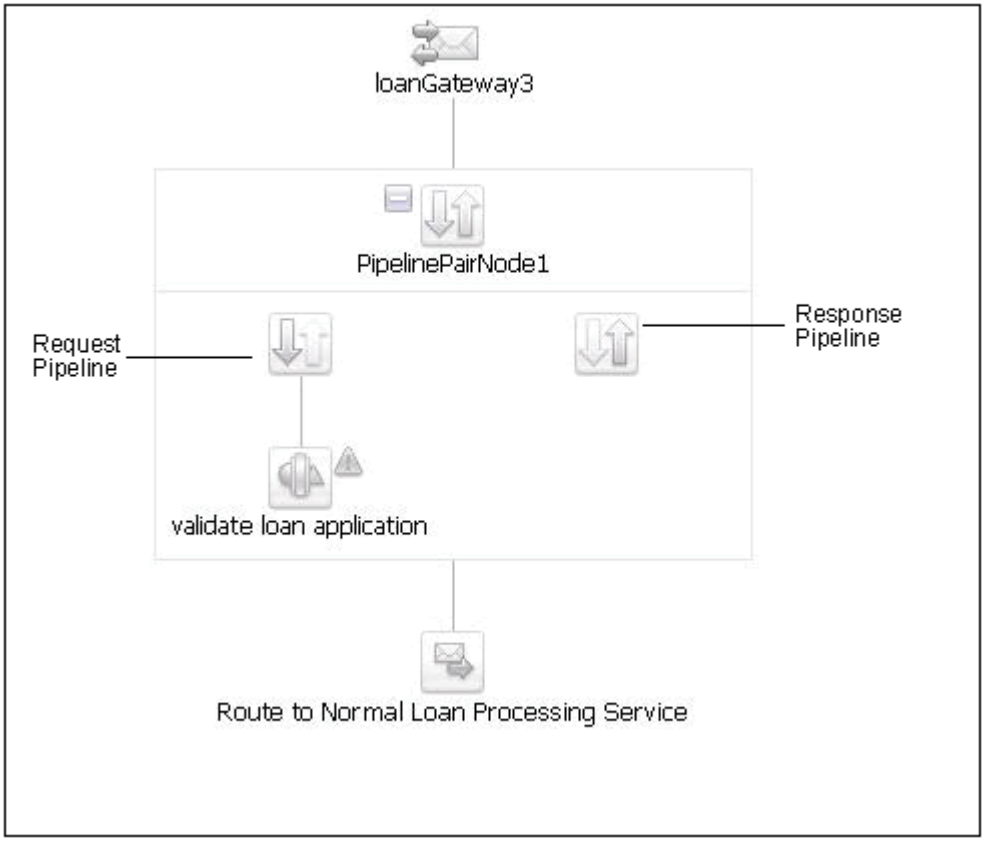
ALSB version 2.6 is supported with the Diagnostics Probe for Java.

The diagram below shows a general ALSB mechanism for handling server requests.

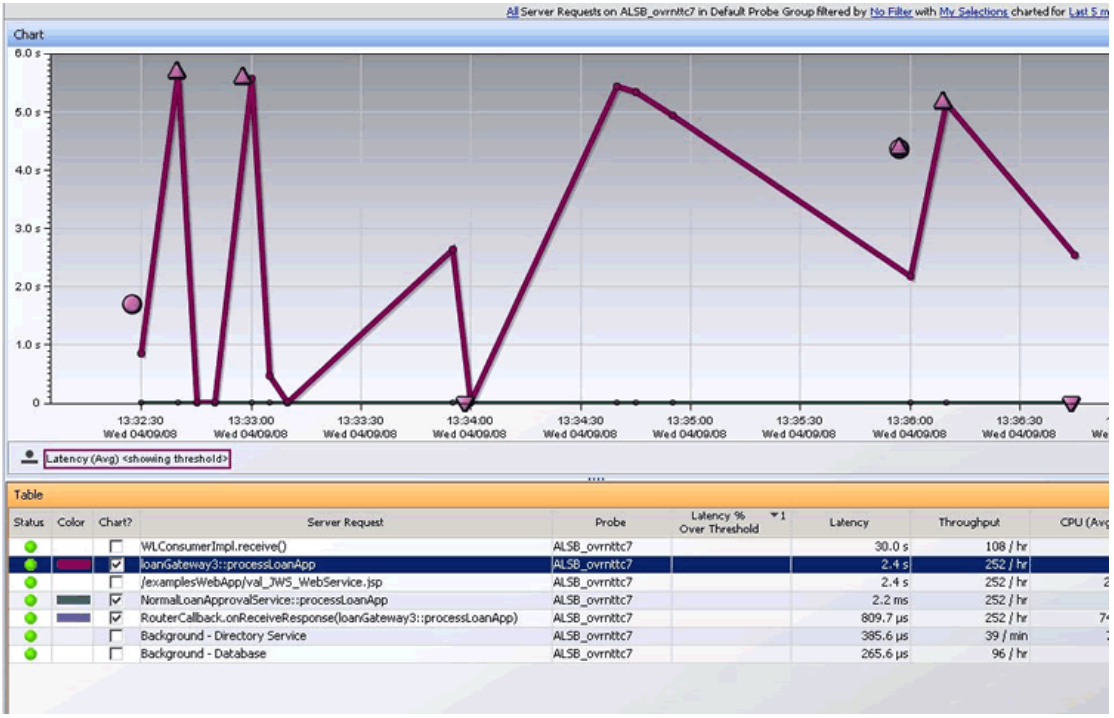


An incoming service request can flow through a series of proxy and business services before the response is returned to the user (client). The proxy and business services in ALSB can be based on several protocols (HTTP, local, JMS). The services based on local protocol are usually internal services which are not exposed to the web. These local services, therefore, are not usually defined using a WSDL and are not supported in Diagnostics.

In AquaLogic service bus, the incoming client request goes through a 'request pipeline' before it is eventually routed to a business service. The response from the business service then goes through a 'response pipeline'. The request and response pipelines work on different threads as shown in the example ALSB message flow view below.



In Diagnostics, the Server Requests view will contain two entries for each proxy service call (as shown in the example below).



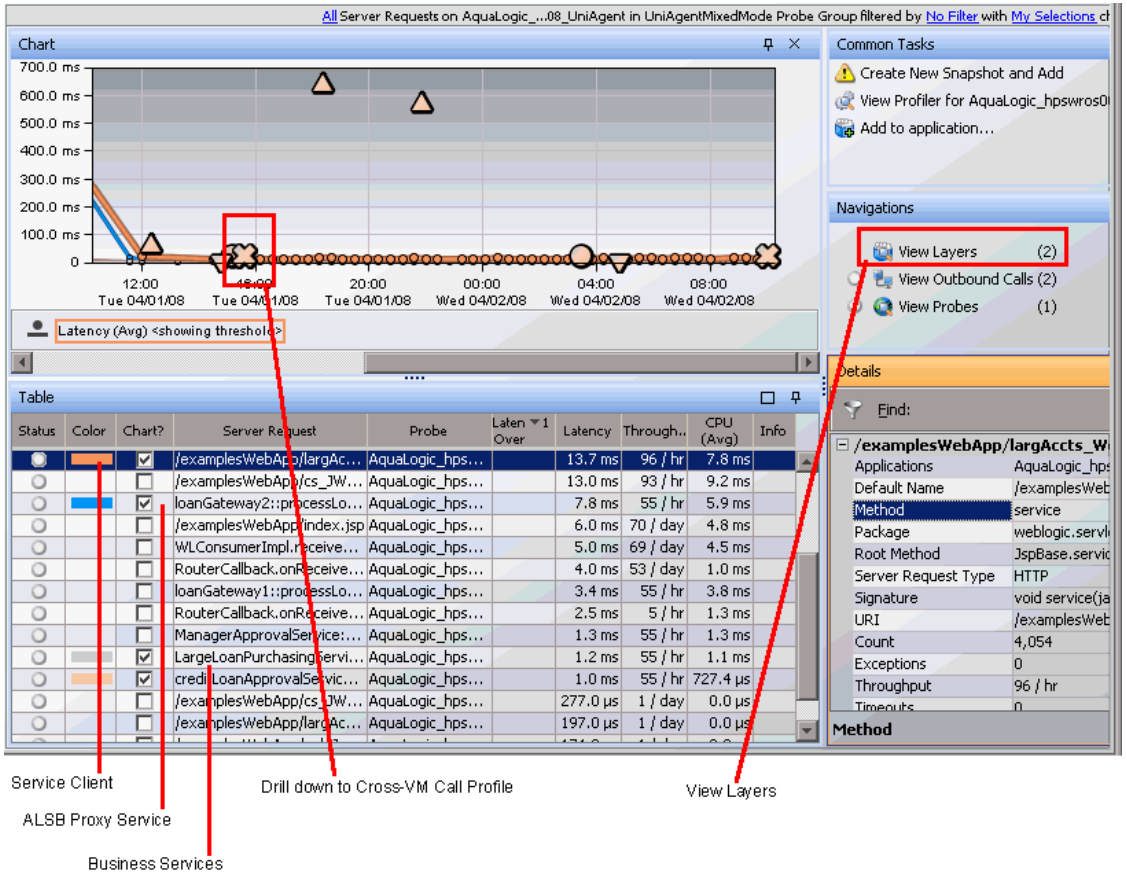
The proxy service request (i.e. loanGateway3::processLoanApp) includes the latency in the request pipeline. When you look at the web service operation, you won't see the corresponding response server request.

The RouterCallback.onReceiveResponse(loanGateway3::processLoanApp) shows the latency of the response pipeline for the corresponding web service operation.

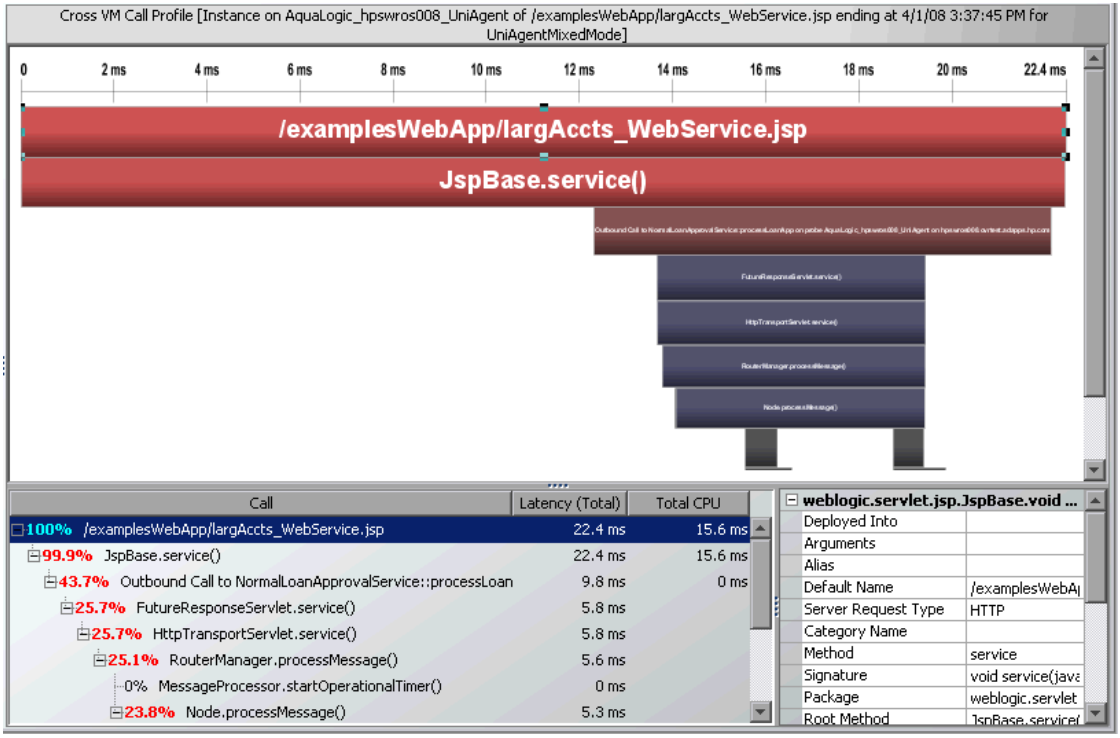
The business service call latency is usually shown as a separate server request (as shown in the figure above with NormalLoanApprovalService::processLoanApp).

Note that in case of an error in the business service invocation, the RouterCallback.onError(loanGateway3::processLoanApp) will show the latency of the response pipeline.

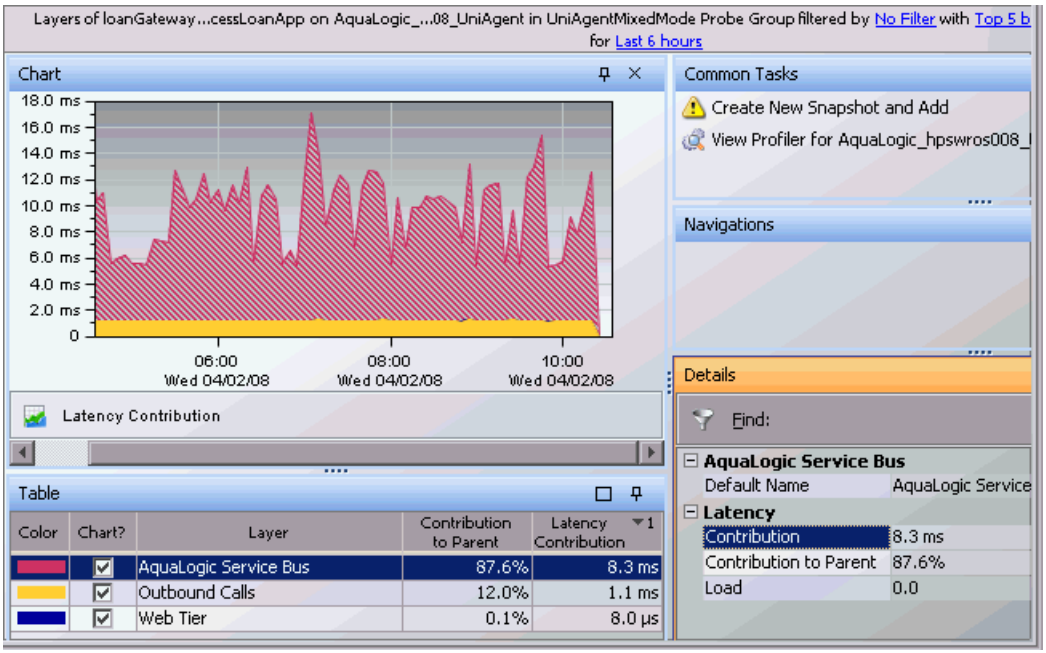
In AquaLogic, the route node can route the request to the business service either synchronously or asynchronously. This can be configured using the 'Quality of Service' attribute of the routing option. By default, the business services are invoked asynchronously. If the business service is invoked synchronously then the proxy service request latency will include the latency of the business service call. Otherwise, it will not include the latency of the business service call.



The cross-VM correlation for multiple levels of outbound service invocations is shown at the top level server request. It does not include the response from the business service.



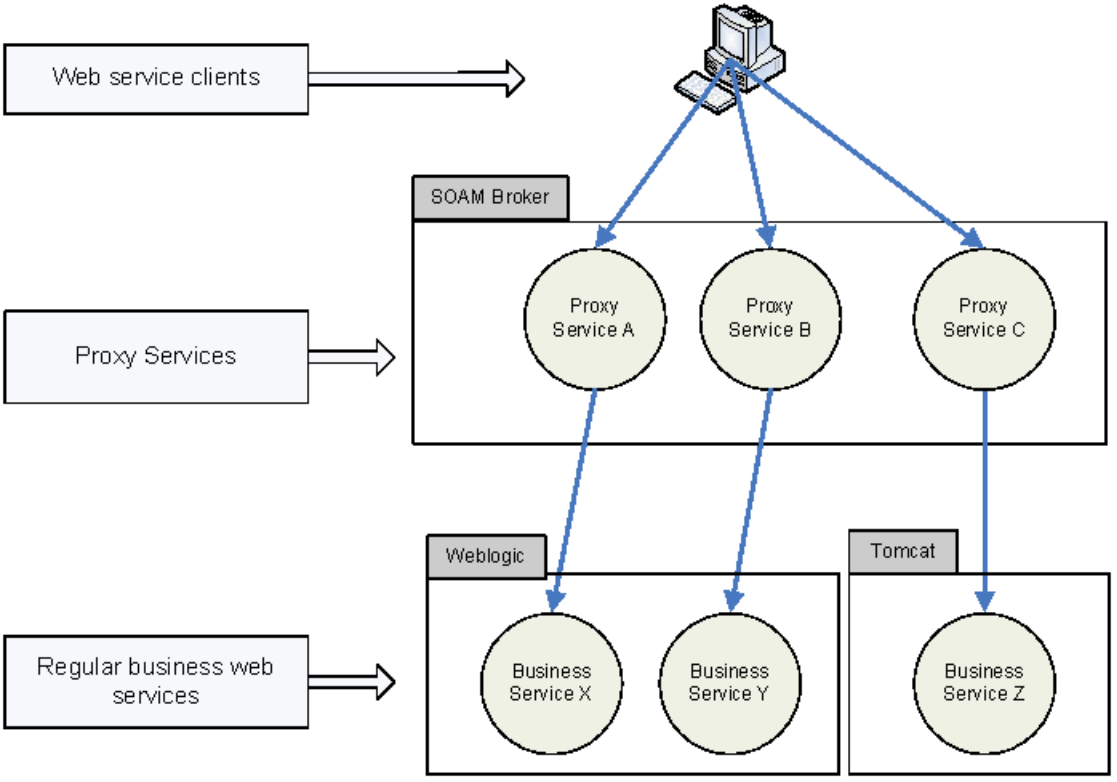
From the Server Requests view you can select View Layers to see the latency by layer.



Monitoring of the SOA Policy Enforcer Broker

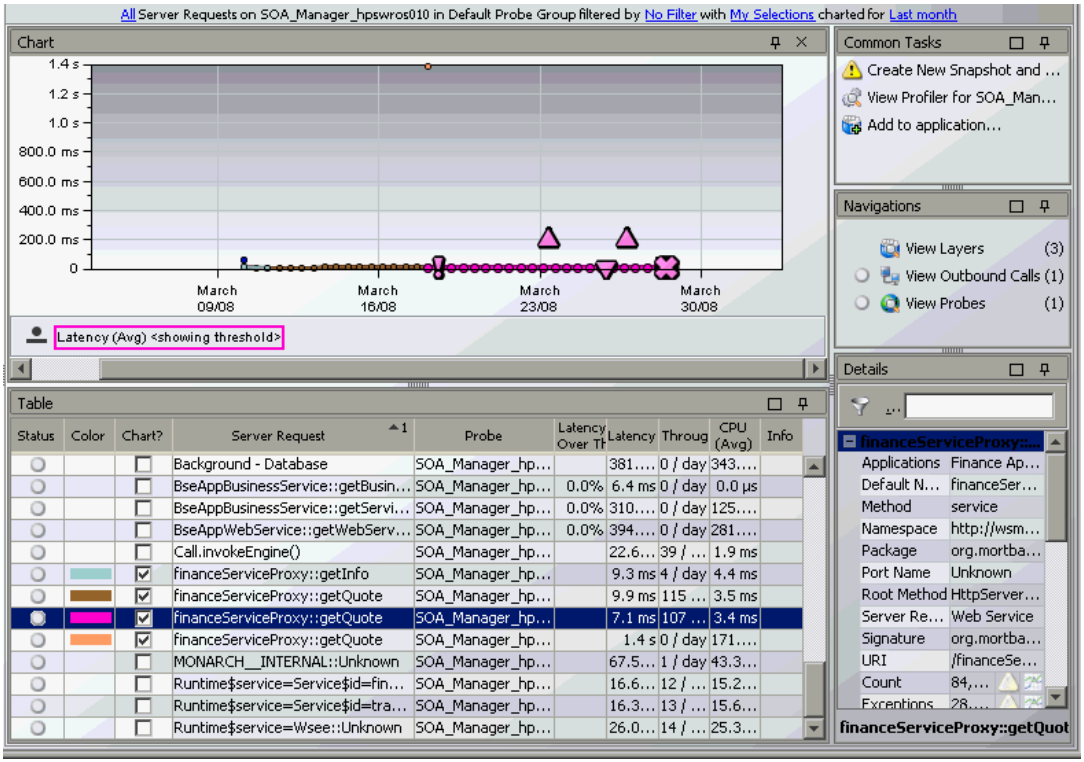
The Diagnostics Probe for Java integrates with the HP SOA Policy Enforcer broker and can be used to monitor proxy web services deployed in the policy enforcer environment. Please note that for SOA Broker 2.5 you would need to manually set up the integration with Diagnostics (see the auto_detect.points file).

The Java Probe includes instrumentation for SOA Policy Enforcer broker inbound and outbound services as well as all the service handlers.

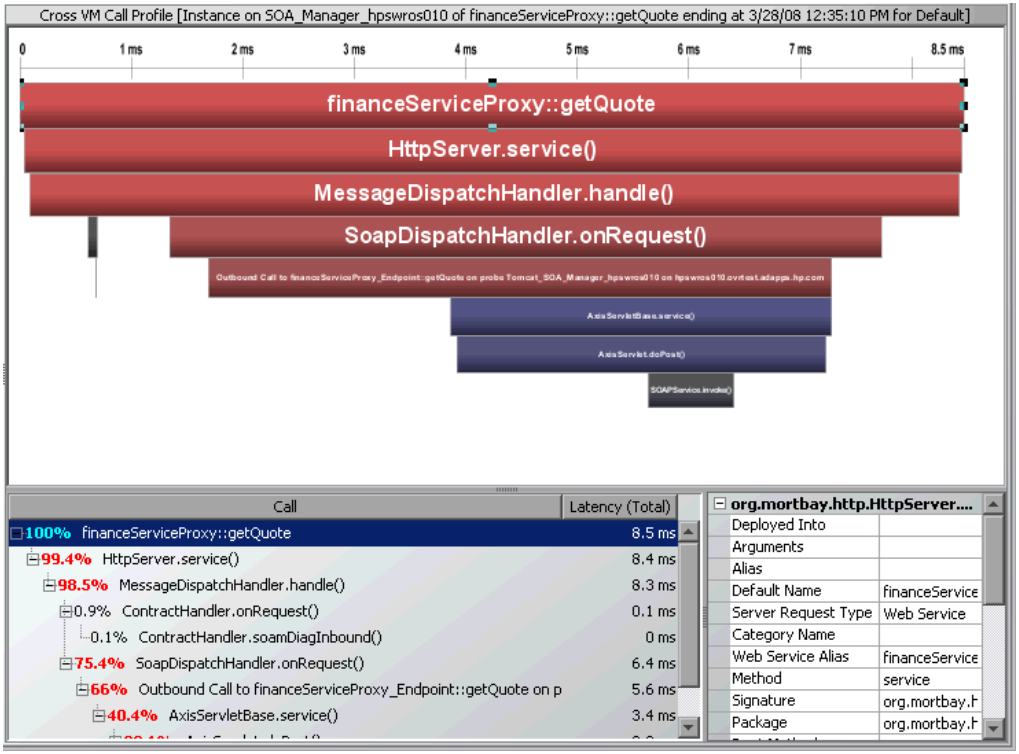


Diagnostics monitors Web services request/response from the client to SOA Policy Enforcer broker through proxy services and then to the business service and the return to the client.

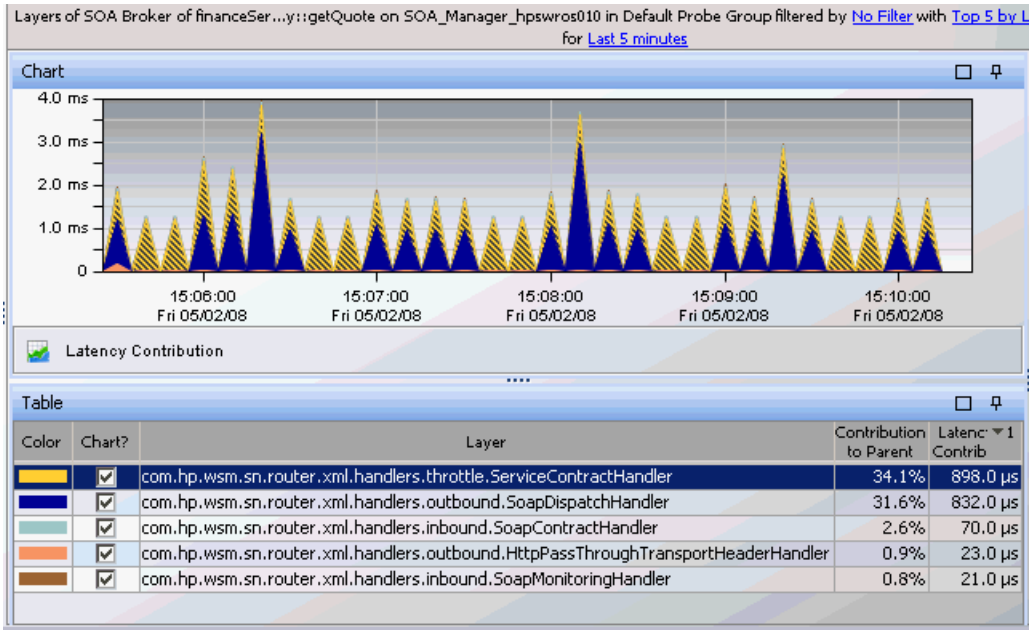
You can see the server request data from your SOA Policy Enforcer environment in the Diagnostics UI as shown below.



From the Server Requests view you can drill down to see detailed call profiles for minimum/maximum instances, cross-VM instances, exceptions and SOAP faults.



From the SOA Services Operations view or the Server Requests view you can select View Layers and then drill down to the sub-layers for the SOA Policy Enforcer broker.



WebSphere Enterprise Service Bus

IBM WebSphere Enterprise Service Bus (ESB) provides an ESB for IT environments built on open standards, SOA, messaging and Web services technologies of WebSphere Application Server.

WebSphere ESB manages the flow of messages between Service Component Architecture (SCA)-described interaction endpoints.

An ESB lets you reuse assets in a flexible way, by passing service interactions through logic called mediations. Mediations operate on messages in transit between requesters and providers; they allow facilities such as message filtering, protocol conversion, database logging, and dynamic routing.

In an SCA-based solution mediation modules are a type of SCA module. Usually mediation modules contain a specific type of SCA component called a mediation flow component. Mediation flow components define mediation flows. A mediation module can contain only one mediation flow component.

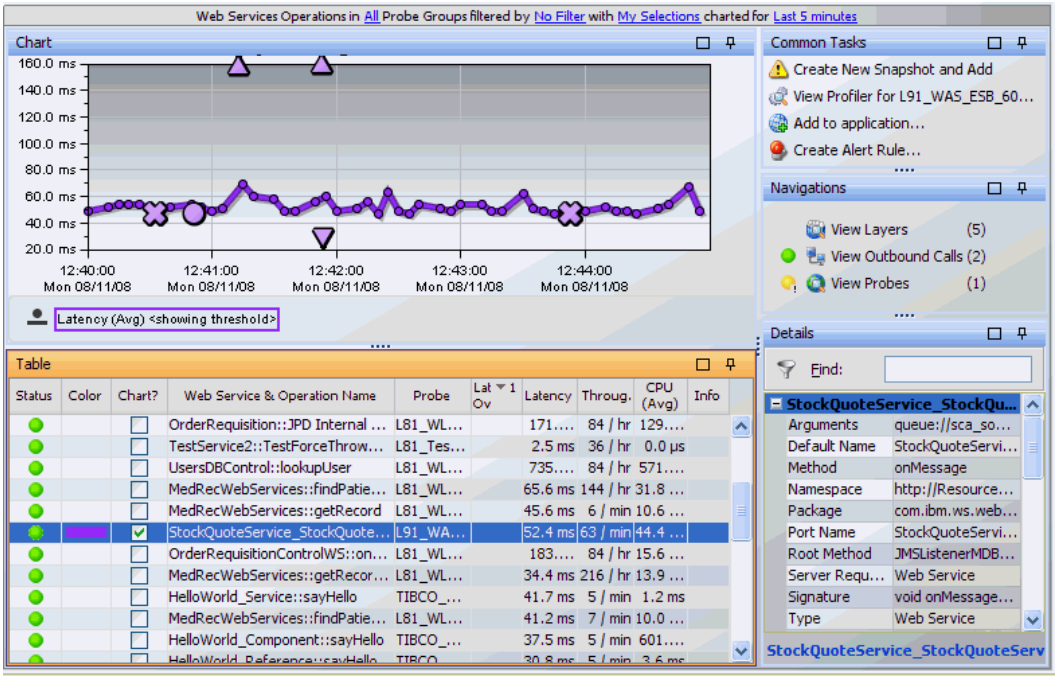
The mediation modules perform a special role, and therefore have slightly different characteristics from other components that operate at the business level.

Mediation components operate on messages exchanged between service endpoints. In contrast with regular business application components, they are concerned with the flow of the messages through the infrastructure and not just with the business content of the messages. Rather than performing business functions, they perform routing, transformation, and logging operations on the messages. Also mediation modules within the ESB handle mismatches between requesters and providers, including protocol or interaction-style mismatches and interface mismatches. The information that governs the behavior of mediation modules is often held in headers flowing with the business messages. The IBM SOA programming model introduces the service message object (SMO) pattern for Service Data Objects (SDOs) to support this pattern.

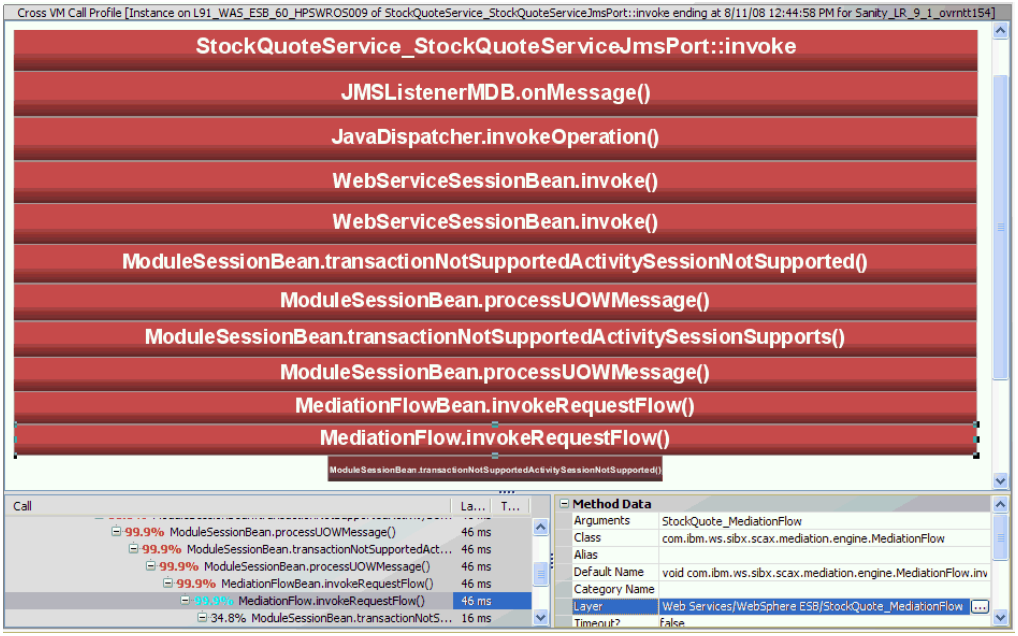
Unlike BEA's Aqualogic Service Bus that has proxy services within the ESB, WebSphere has Mediation flows, which are not web services and have no associated WSDL.

The way Diagnostics shows these mediation flows is by layers. A layer is displayed in Diagnostics for the Mediation Flow. The Mediation Flow component name prefixed by Web Services/WebSphere ESB/ is used as the layer name. See the example on the following pages that includes the Layers views.

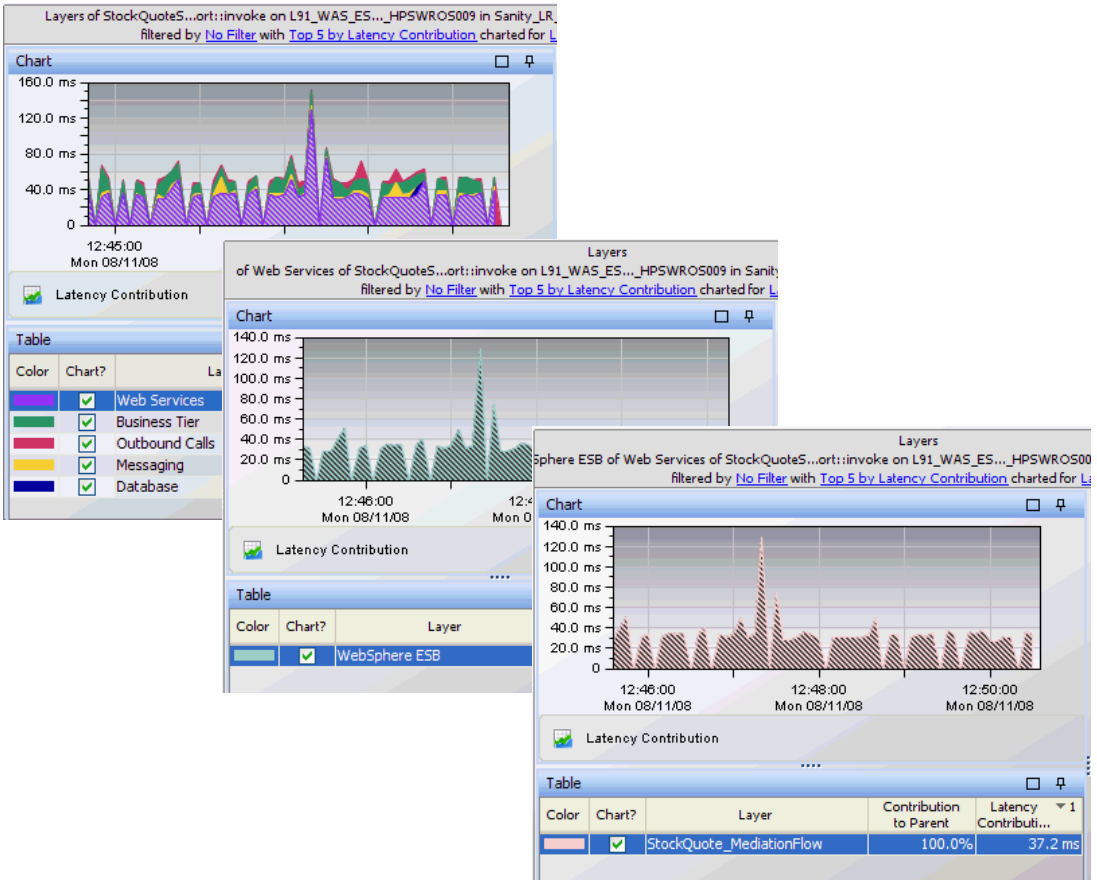
The existing WebSphere Inbound and Outbound web service instrumentation in Diagnostics collects performance data on the web services.



Stock Quote example of Web Services Operations view for an environment with a WebSphere ESB.



Call Profile view for a WebSphere ESB environment. In the Stock Quote example both the invokeRequestFlow and invokeResponseFlow methods are instrumented for capturing the mediation flows as layers. And you see the Web Services/WebSphere ESB/StockQuote_MediationFlow layer.

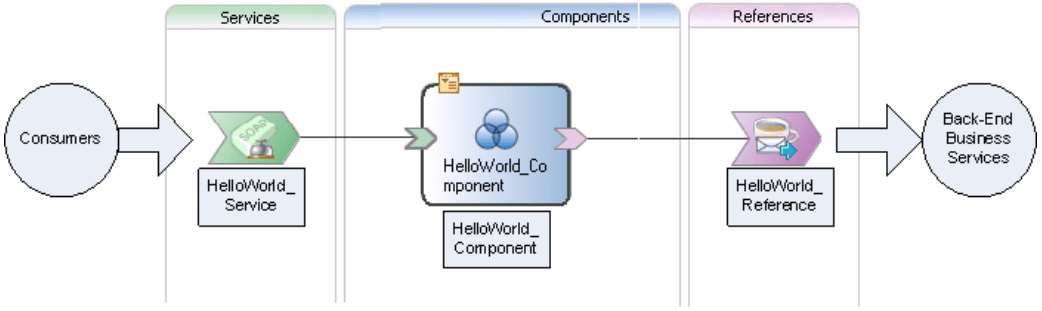


You can also see the breakdown of the layers in the screen shots above from Web Services layer - WebSphere ESB layer to StockQuote_MediationFlow layer.

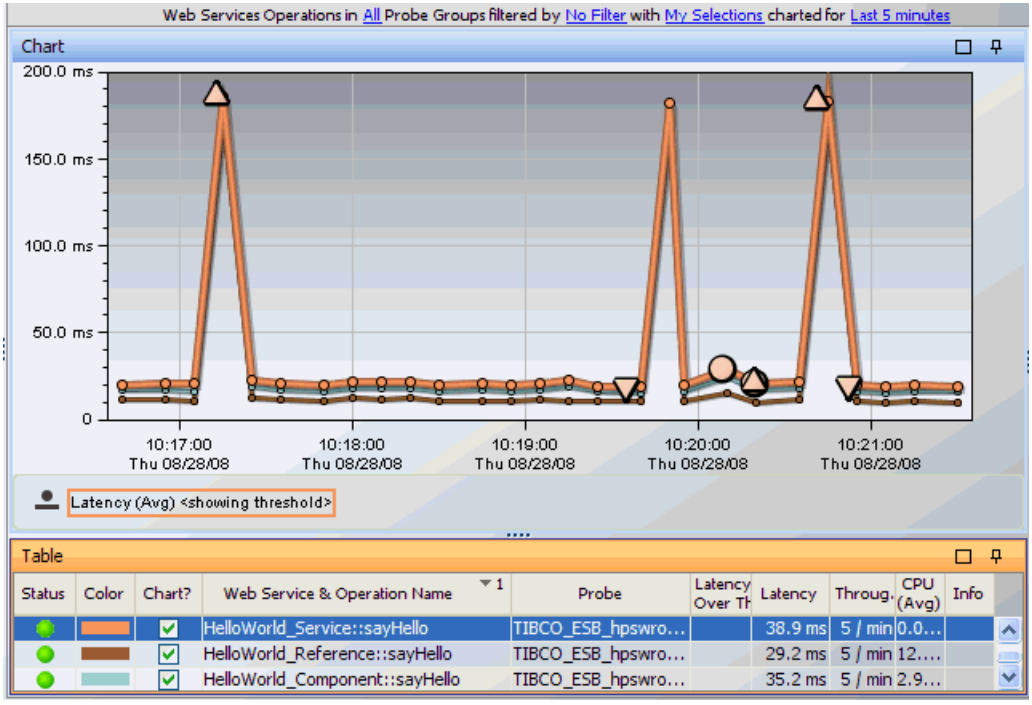
TIBCO ActiveMatrix Service Bus

TIBCO ActiveMatrix Service Bus is a lightweight enterprise service bus. Diagnostics monitors the web services in a TIBCO ActiveMatrix Service Bus environment.

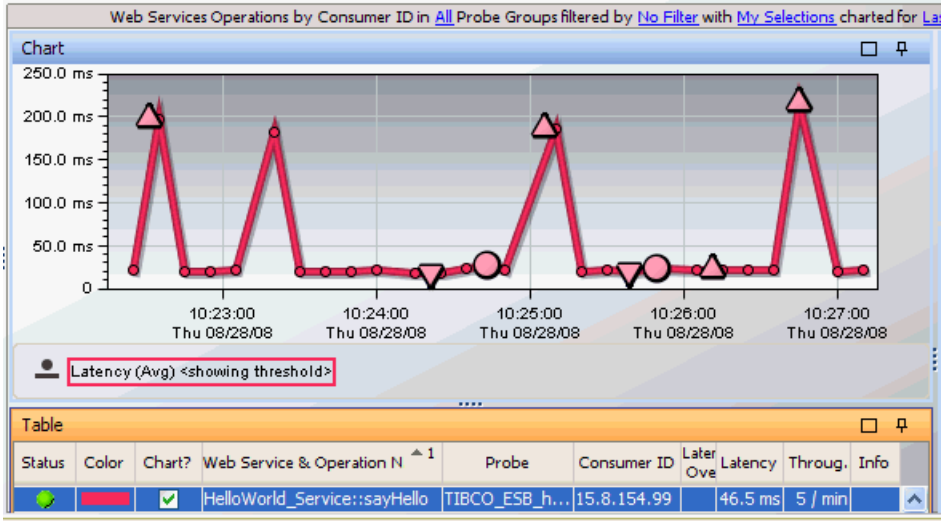
The figure below shows a typical scenario for a TIBCO ActiveMatrix Service Bus based web service. The consumers invoke the front-end "HelloWorld_Service" web service. This front-end service invokes an internal "HelloWorld_Component" mediation service. The mediation service then calls the "HelloWorld_Reference" reference service which in turn invokes the actual back-end business service.



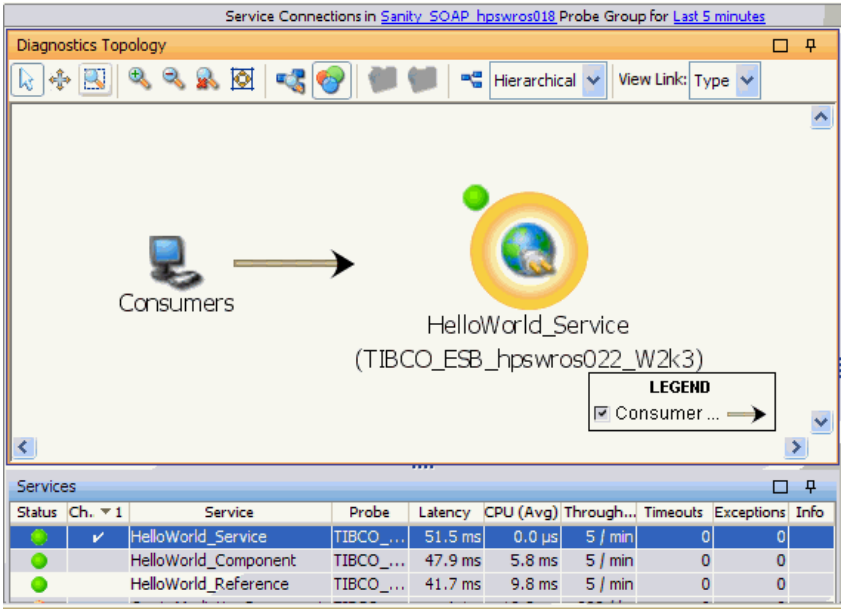
In this type of example, Diagnostics monitors and displays performance data for the front-end "HelloWorld_Service" and the internal services "HelloWorld_Component" and "HelloWorld_Reference". The internal services are shown so that any performance bottlenecks within these services can be identified. It can also help in throughput analysis of the internal services when the mediation component is designed to make routing decisions to route to different reference services. See an example below.

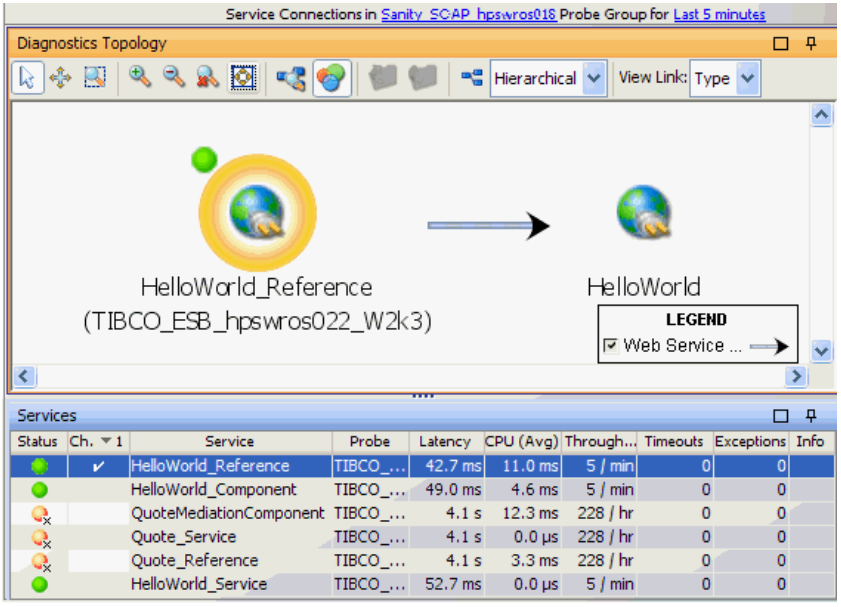


In the Web Services Operations by Consumer ID view, Diagnostics only shows the front-end service(s) because the consumer only interacts with the front-end "HelloWorld_Service" web service. See an example below.



In the two examples below of the Service Topology view; the first shows the consumer interaction with the front-end service while the second shows the reference service invoking a back-end business service.





Additional JMX metrics have been added as part of TIBCO Active Matrix Platform support. These metrics are exposed by the Apache Tomcat and Pramati J2EE server components used by the ActiveMatrix platform. Some of these metrics are inactive by default. They can be activated by uncommenting them in the **metrics.config** file. The reason to make them inactive is that they are mainly performance tuning configuration parameters and should rarely change during the lifetime of an application.

TIBCO ActiveMatrix BusinessWorks

Diagnostics monitors the web services in a TIBCO ActiveMatrix BusinessWorks environment. For BusinessWorks you can see the typical web services monitoring data in the SOA Services views **except** the following:

- captured soap faults
- captured soap payload on faults
- web service topology

Note: In order to view outbound calls, cross VM correlation and the probe topology for the SOAP/HTTP calls, you will need to enable the **[Apache-HttpClient-Outbound]** point in the probe's **etc/auto_detect.points** file.

BEA WLI Business Process Tracing Data

With BEA WLI's out of the box instrumentation, you can view your JPD, JWS and Controls within the Diagnostics console through many of the standard views, including Server Requests, Outbound and SOA Services screens.

The current versions of BEA WLI are supported in this release: BEA Weblogic Platform v8.1.4, v8.1.5, v8.1.6.

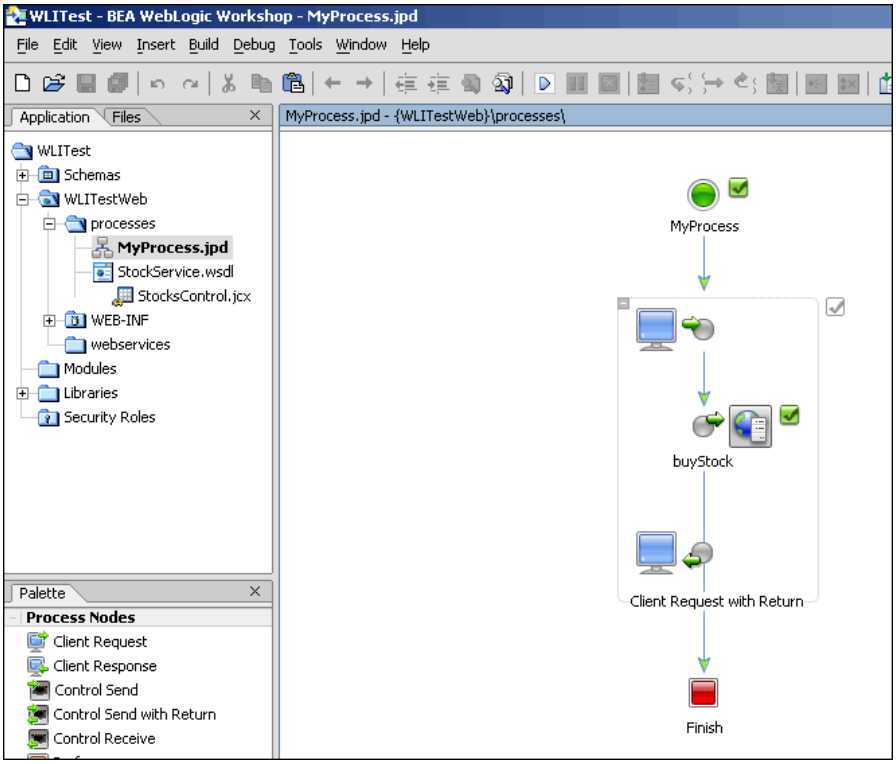
In the context of Business Process Management Suites (BPMS), transaction tracing is the correlation of Business Processes with application actions. For example when a Business Process is invoked, corresponding events are instrumented on the database, application server and any ancillary systems such as messaging and legacy systems. You can use Diagnostics to make this correlation between business processes and the systems with which they interact.

Viewing BEA WLI Data with Diagnostics

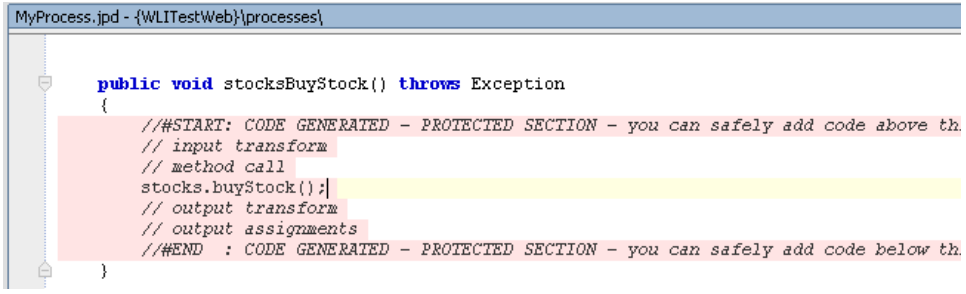
The following example demonstrates synchronous JPD calls made from MyProcess to a JWS called StockService. The example shows the BEA WLI views of these calls and the Diagnostics views of the same calls.

- MyProcess is running in wl815-jvm142-ovrntt1-wli1 WLI server (on port 7001).
- StockService is running in wl815-jvm142-ovrntt1-wli2 WLI server (on port 7003).

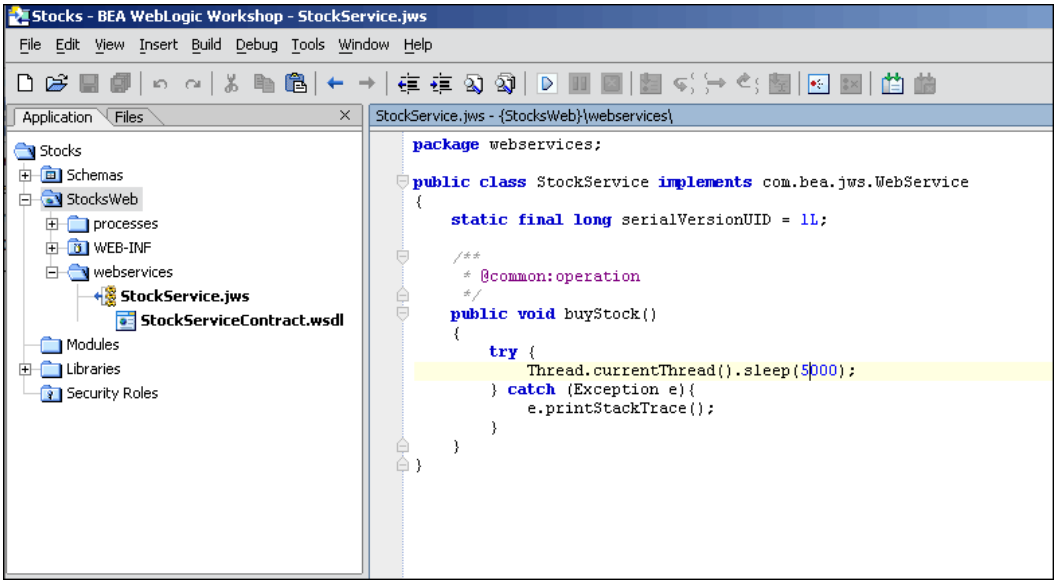
A typical view in Workshop is shown below:



Drilling into the code for MyProcess JPD reveals the outbound call to the StockService WPS Web service. MyProcess code is shown below calling out to stocks.buyStock() Web service:

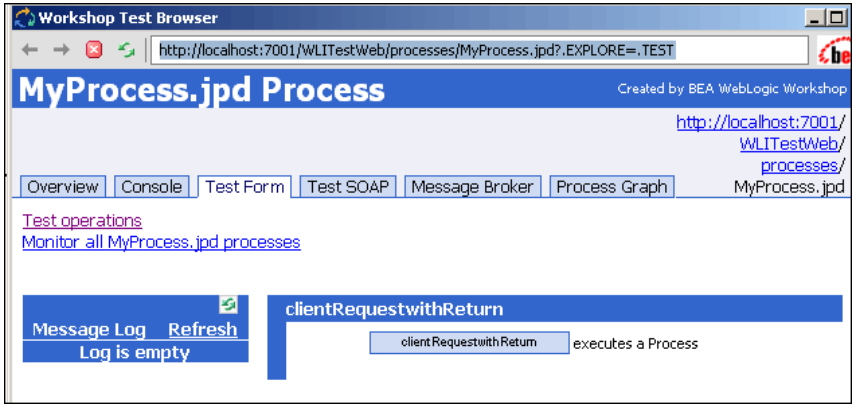


On the receiving end, the StockService receives the request and simply sleeps for five seconds. The StockService Web service application is shown below:

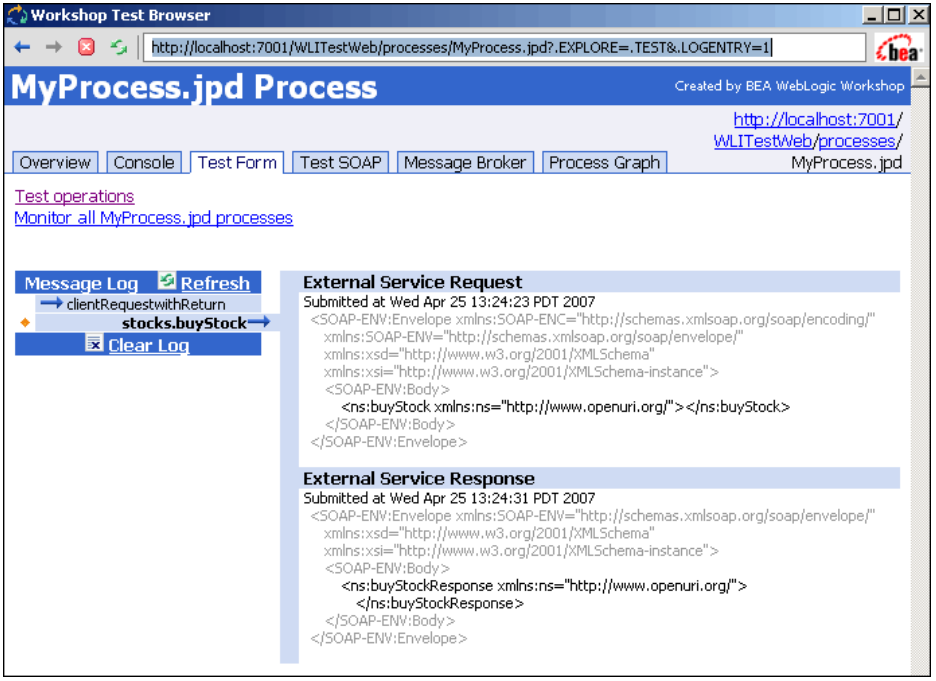


Note: This application is deployed to another domain called **wl815-jvm142-ovrntt1-wli2**. You can see this cross-JVM call in the Diagnostics Call Profile view shown later in this section.

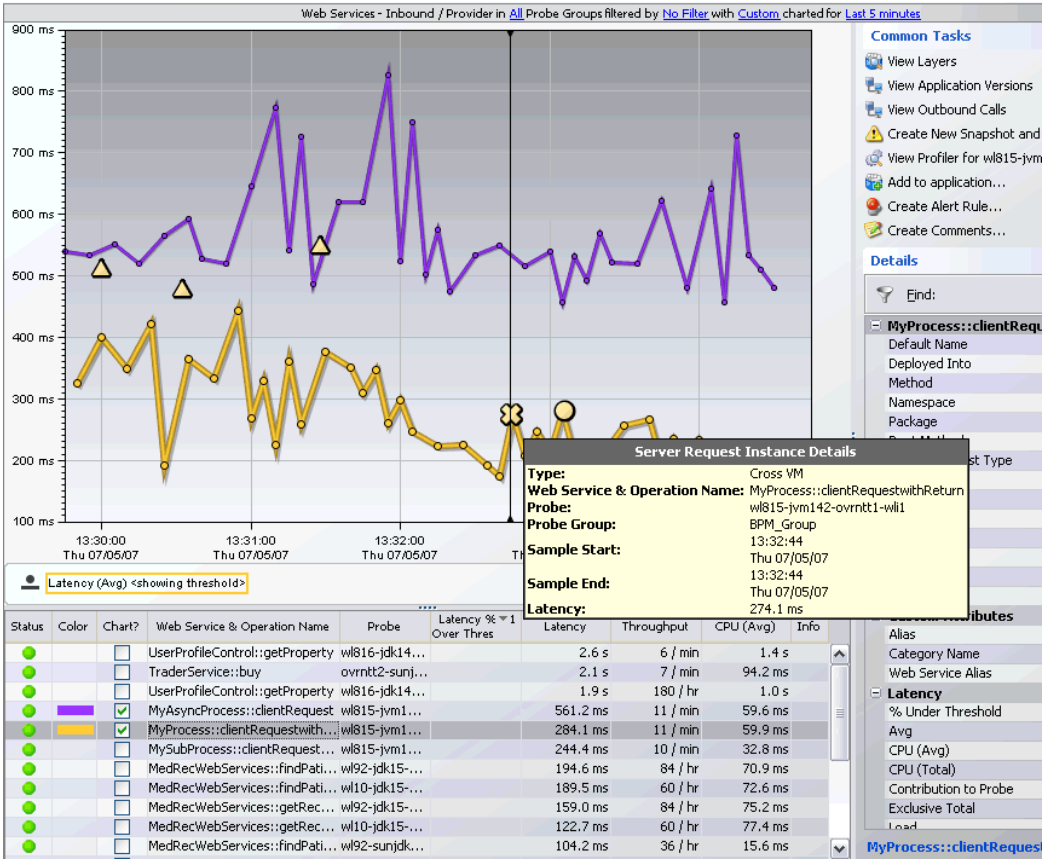
The Workshop Test Browser is used to initiate the MyProcess JPD request to the StockService WPS.



The result of the request is then viewed in the Test Browser. As you can see here, the method called on the StockService is the **buyStock** method. You can also see in the left pane that the call **MyProcess.clentRequestwithReturn** was called.

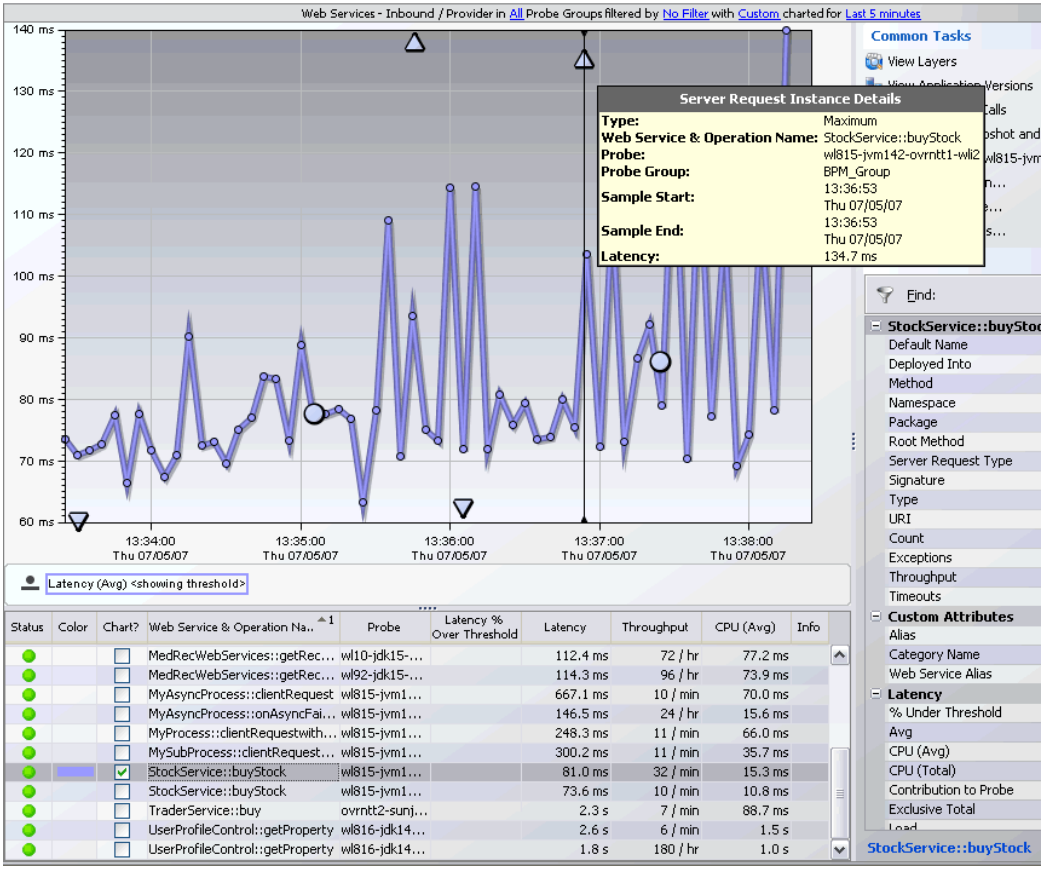


Next, open the Diagnostics server to the SOA Services - Operations view. This shows the call **MyProcess::clientRequestwithReturn** as a Cross-JVM call to the **StockService::buyStock** method.



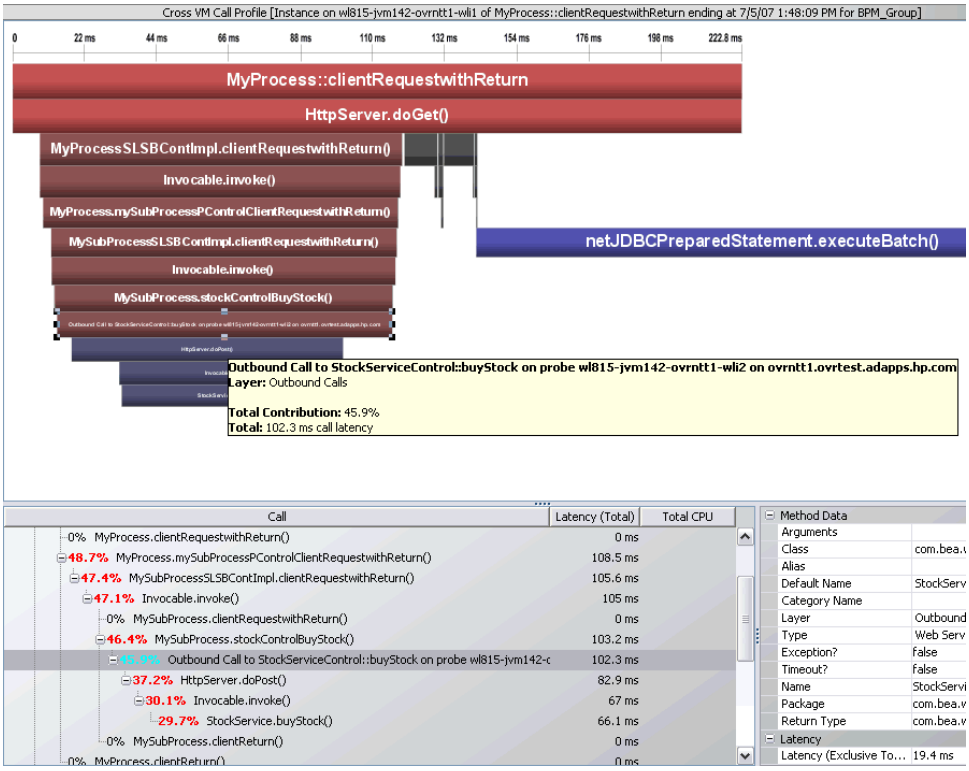
The X icon indicates a cross-jvm call between the 'MyProcess' to the 'StockService' Web service.

Since the StockService::buyStock operation is also a single server request received by the wl815-jvm142-ovrntt1-wli2 WLI server, it appears as an inbound request. Hovering over the call reveals a time of 134.7 milliseconds.



StockService::buyStock server request is also captured.

Clicking on the Cross-JVM X icon in the SOA Services - Operations views displays the resulting server request made from **MyProcess::clientRequestWithReturn** (on integration1 WLI server) out to the **StocksControl::buyStock** method.



Drill down into the X icon demonstrates the cross-JVM Web service call from the 'MyProcess' JPD to the 'StockService' Web Service.

30

SAP Views

The **SAP** view group displays two different types of SAP performance data:

- **SAP ABAP.** Represents the ABAP stack of the SAP system.

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

  HP Diagnostics displays SAP ABAP 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 Diagnostics Probe for Java. HP Diagnostics displays SAP NetWeaver performance data and metrics in the SAP view group.

This chapter includes:

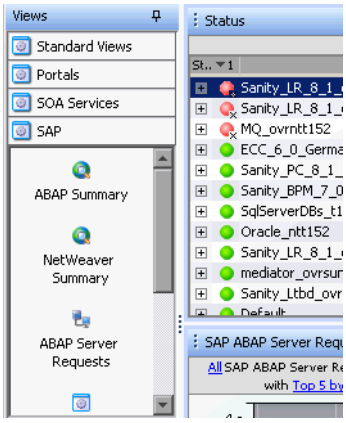
Accessing the SAP Views

You can access the SAP views from the View bar in the Diagnostics views.

To access the SAP views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **SAP** to open the SAP view group.



If the SAP view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **SAP** and click **OK**. The SAP view group is displayed in the view bar.

3 Select the appropriate view from the SAP view group.

Description of the SAP Views

The default views contained in the SAP view group, are described in the following sections. See the *HP Diagnostics Installation and Configuration Guide* chapter on "Installing the Diagnostics Collector" for details on configuring the SAP collector.

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 21, "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 "Aggregate Requests and Server Requests Views" on page 311.
- **Portal Components.** A monitoring version of the standard Diagnostics Portal Components view. For more information about the Portal Components view, see Chapter 28, "Portals Views."
- **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 295.
- **NetWeaver Threads.** A monitoring version of the NetWeaver Threads view. For more information, see "NetWeaver Threads" on page 483.
- **NetWeaver Requests.** A monitoring version of the NetWeaver Requests view. For more information, see "NetWeaver Requests" on page 482.

ABAP Summary

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

The ABAP 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 21, "Status View."

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

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

- **SAP ABAP Probes.** A monitoring version of the ABAP Probes view. For more information, see "ABAP Probes" on page 483.

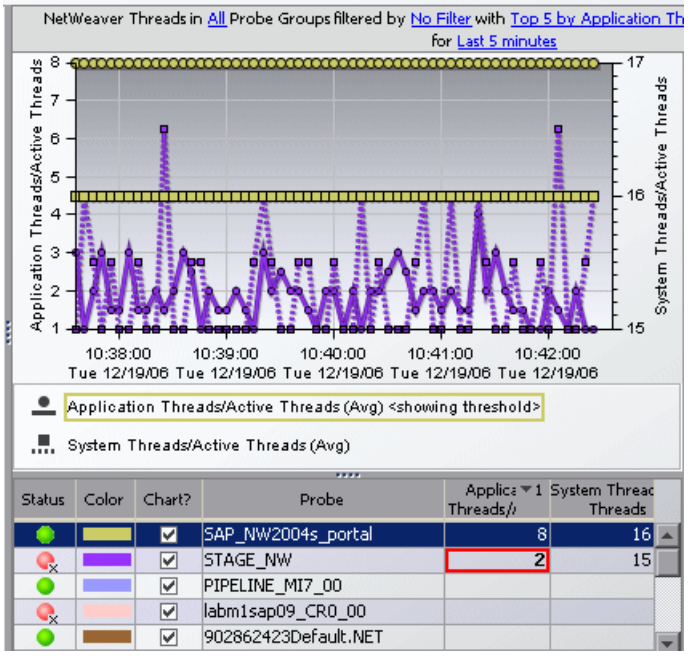
- **SAP ABAP Server Requests.** A monitoring version of the ABAP Server Requests view. For more information, see "ABAP Server Requests" on page 484.

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.



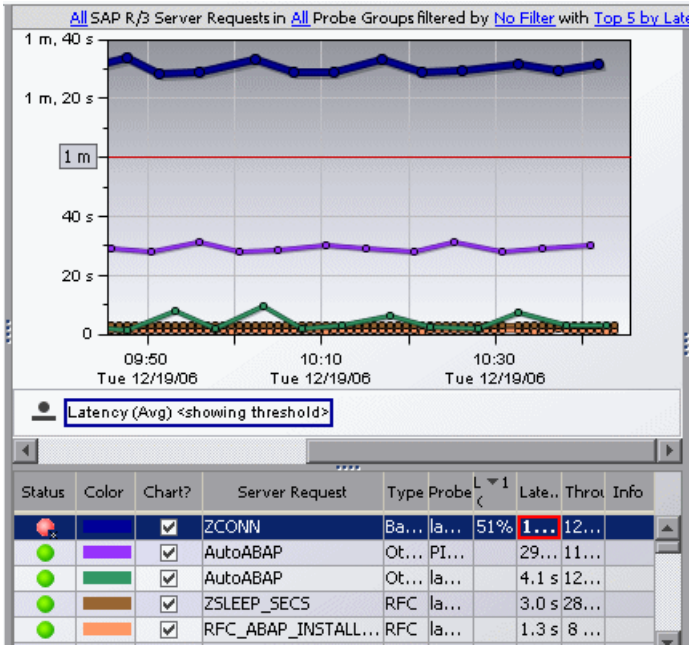
ABAP Probes

The ABAP Probes view displays unique metrics for the SAP ABAP instances.

The default metric displayed in the view for SAP ABAP Probes is **Average Latency**.

ABAP Server Requests

The ABAP Server Requests view displays information about the server requests running on the ABAP stack of the SAP system.



31

Oracle Database Views

The **Oracle Database** view group displays Oracle performance data. When you install and configure the Diagnostics Collector to gather data from an Oracle Database, you define instances of Oracle to be monitored. Each one of these instances is represented as an Oracle Probe, belonging to a probe group.

This chapter includes:

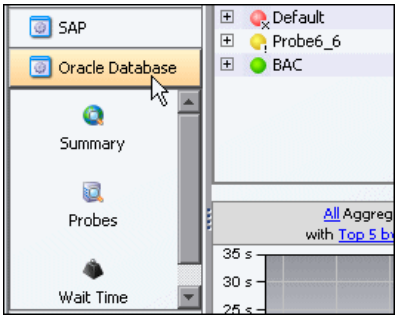
Accessing the Oracle Database Views

You can access the Oracle Database views from the View bar in the Diagnostics views.

To access the Oracle views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **Oracle Database** to open the Oracle Database view group.



If the Oracle Database view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **Oracle Database** and click **OK**. The Oracle Database view group is displayed in the view bar.

3 Select the appropriate view from the Oracle Database view group.

Description of the Oracle Views

The Diagnostics Oracle collector runs a number of queries to retrieve the information displayed in the Oracle views. Basic database information such as the name, id and status is retrieved from the V$DATABASE table along with the instance name, host name, version, startup time and status from the V$INSTANCE table. Diagnostics uses SYSDATE from the DUAL table to help in time-synchronizing the collection of data between the collector and the database.

Instance level metrics are retrieved from the V$SYSMETRIC table where INTSIZE_CSEC is greater then 3700. All the operating system statistics are collected from the V$OSSTAT table. The time model stats are collected from the V$SYS_TIME_MODEL table. And Diagnostics collects all wait events that are in the V$SYSTEM_EVENT table and matches them up to their wait class from the DBA_HIST_EVENT_NAME table. Additional details on each table can be found on Oracle's website or by querying the different table names and viewing the results.

See the *HP Diagnostics Installation and Configuration Guide* chapter on "Installing the Diagnostics Collector" for details on configuring the Oracle collector.

The Oracle view group contains the following views:

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 488.

➤ **Wait Time.** A monitoring version of the Oracle Wait Time view. For more information, see "Oracle Wait Time View" on page 488.

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.

32

SQL Server Database Views

The **SQL Server Database** view group displays SQL Server performance data. When you install and configure the SQL Server Database Collector to gather data from an SQL Server database, you define instances of SQL Server to be monitored. Each one of these instances is represented as an SQL Server Probe, belonging to a probe group.

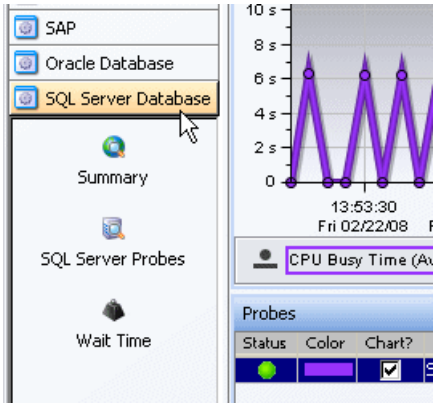
This chapter includes:

Accessing the SQL Server Database Views

You can access the SQL Server Database views from the View bar in the Diagnostics views.

To access the SQL Server views:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **SQL Server Database** to open the SQL Server Database view group.



If the SQL Server Database view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **SQL Server Database** and click **OK**. The SQL Server Database view group is displayed in the view bar.

3. Select the appropriate view from the SQL Server Database view group.

Description of the SQL Server Database Views

Diagnostics collector automatically collects data for all SQL Server databases in the instance. If you want to exclude some of these databases from collection (for example system databases), you can specify a comma-separated list in the **exclude.db.list** property in the **<collector_install_dir>\etc\sqlserver.properties** file. See the *HP Diagnostics Installation and Configuration Guide* chapter on "Installing the Diagnostics Collector" for details on configuring the SQL Server collector.

When you have ***n*** databases in your instance you actually will have ***n+1*** probes; an extra probe for the totals of the instance which includes metrics such as wait events. The extra probe uses probeName, the probes for each database use probeName_databaseName.

The Diagnostics SQL Sever collector runs a number of queries to retrieve the information displayed in the SQL Server Database views. Basic database information such as the name and status is retrieved from the sysdatabases table along with the version information from the sysservers table. Diagnostics uses the GETDATE() function that SQL Server provides to help in time-synchronizing the collection of data between the collector and the database.

Instance level metrics are retrieved from the sys.dm_os_performance_counters table. The operating system statistics are collected from the sys.dm_os_sys_info table and include the CPU count as well as physical and virtual memory. And Diagnostics collects all wait events that are in the sys.dm_os_wait_stats table. Additional details on each table can be found on Microsoft's SQL Server website or by querying the different table names and viewing the results.

The SQL Server view group contains the following views:

Summary View

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

The Summary dashboard view contains the following panes:

➤ **Status pane.** A table displaying the SQL Server 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.

All Diagnostics probe groups are displayed in this view, but when you double-click each probe group, only those with SQL Server Probes will open to display details. Detailed status tables are displayed for the probe and for the host.

Important: Host metrics are collected and displayed in the hosts status table only if the Java Probe is also installed on the host system.

➤ **Wait Time.** A monitoring version of the SQL Server Wait Time view. For more information, see "Wait Time View" on page 492.

➤ **Probes.** A monitoring version of the SQL Server Probe view. For more information, see "SQL Server Probes View" on page 492.

SQL Server Probes View

The SQL Server Probes view displays unique metrics for the SQL Server Database instances. From the SQL Server Probe view, you can drill down further to the Load view by double-clicking the probe in the graph legend.

The layers in the SQL Server Probe Load view represent the actions that take place on the specific SQL Server instance. Layer names and calculations in this view are based on the SQL Server time model.

You can drill down to more layers by double-clicking the **Database** layer in the Load graph legend.

Wait Time View

The SQL Server Wait Time view displays wait event classes for the SQL Server instances.

Wait event classes represent specific types of events that SQL Server 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.

33

MQ Views

The MQ view group displays WebSphere MQ performance data. IBM WebSphere MQ is a system for messaging across multiple platforms.

The q1ueue manager is the primary component of an IBM WebSphere MQ installation. It provides a logical container for the message queue, which is an object that stores messages in an application. The queue manager is also responsible for transferring data to other queue managers by using message channels. Communication between queue managers relies on a separate program, called a channel. The channel runs on the same host as the queue manager. It handles sending and receiving data over the network.

This chapter includes:

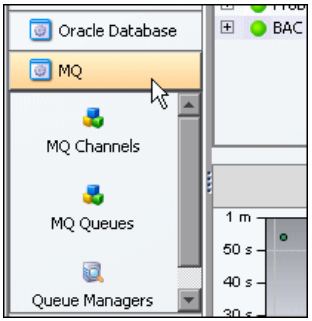
Accessing the MQ Views

You can access the MQ views from the View bar in the Diagnostics views.

To access the MQ views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **MQ** to open the MQ view group.



If the MQ view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **MQ** and click **OK**. The MQ view group is displayed in the view bar.

3 Select the appropriate view from the MQ view group.

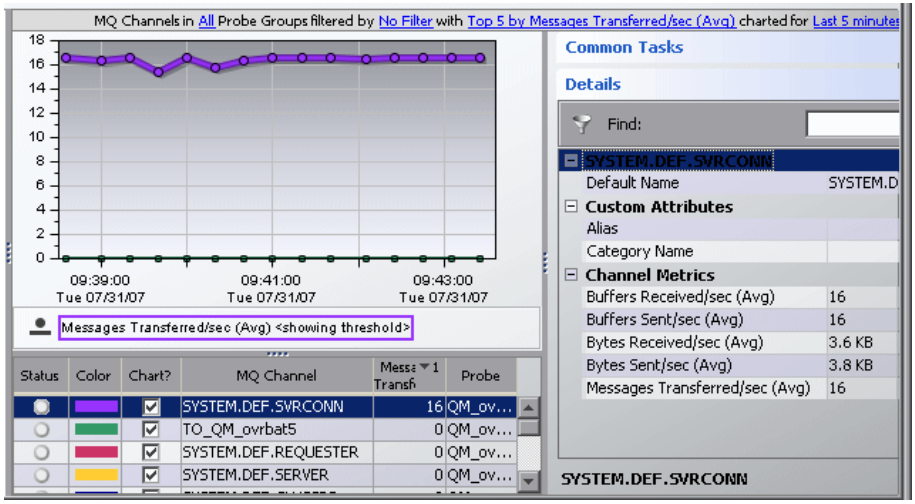
Description of the MQ Views

The Diagnostics MQ view group contains the following views. See the *HP Diagnostics Installation and Configuration Guide* chapter on "Installing the Diagnostics Collector" for details on configuring the MQ collector.

MQ Channels

The MQ Channels view displays performance metrics for the channels in your MQ environment.

An example of an MQ Channels view is shown below.



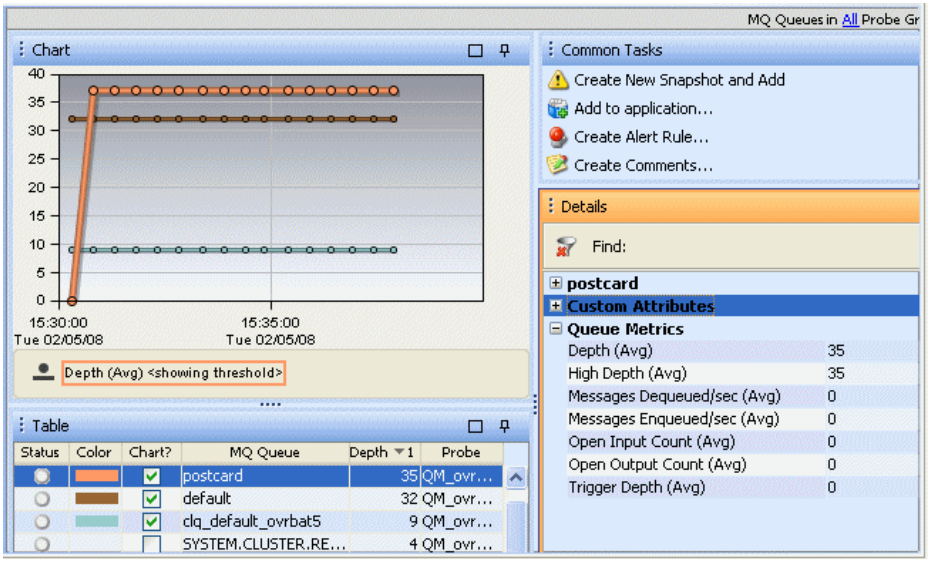
The MQ Channels view displays the following default metrics:

- **Buffers Received/sec.** The rate of buffers received by this channel.
- **Buffers Sent/sec.** The rate of buffers sent from this channel.
- **Bytes Received/sec.** The rate of bytes received by this channel.
- **Bytes Sent/sec.** The rate of bytes sent from this channel.
- **Messages Transferred/sec.** The rate of messages transferred in this channel.

MQ Queues

The MQ Queues view displays performance metrics for the queues in your MQ environment. You can specify which types of queues are included in the metrics. By default, all types of queues are included. For more information about how to specify the queues included in the metrics, refer to the *HP Diagnostics Installation and Configuration Guide*.

An example of an MQ Queues view is shown below.



The MQ Queues view displays the following default metrics:

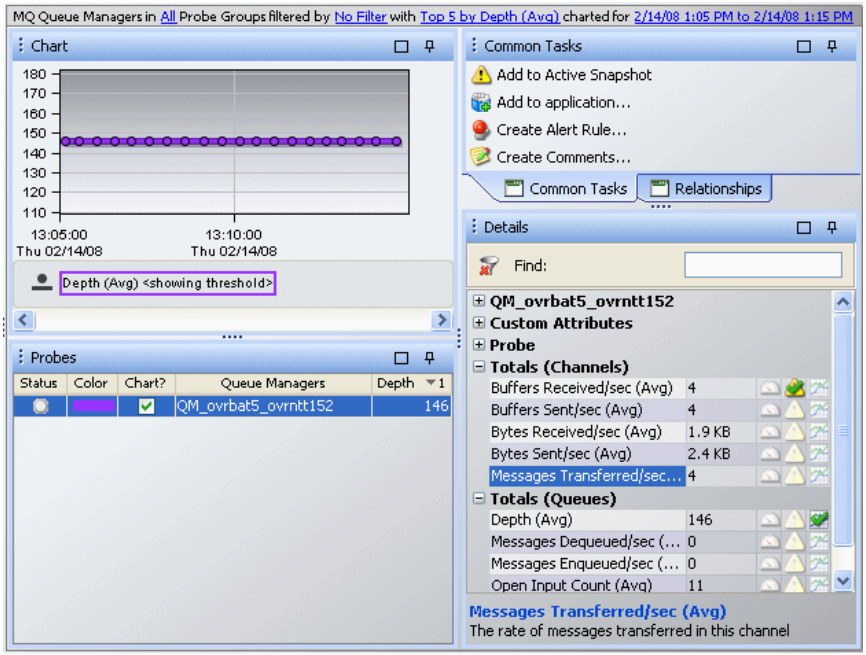
- **Depth.** The current depth of the queue.
- **High Depth.** The maximum number of messages on the queue during the last interval. (The interval is the time between queries. It can be configured in mq.properties. The default is 20 seconds.)
- **Messages Dequeued/sec.** The number of messages dequeued per second.
- **Messages Enqueued/sec.** The number of messages enqueued per second.
- **Open Input Count.** The total number of handles that are currently valid for removing messages from the queue.
- **Open Output Count.** The total number of handles that are currently valid for adding messages to the queue.

➤ **Trigger Depth.** The number of messages that have to be on the queue before a trigger message is written when trigger type is set to MQC.MQTT_DEPTH.

Queue Managers

The Queue Managers view displays metrics for all monitored MQ Queue Managers.

An example of the Queue Managers view is shown below.



The metrics on this screen are an aggregation of the queue and channel metrics shown in the MQ Channels and MQ Queues view. They are grouped in two categories: **Total (Channels)** and **Total (Queues)**.

For example, under the **Total (Channels)** category, you can see a metric called **Messages Transferred/sec**. This is the total rate of messages transferred for the entire MQ environment (that is, for all the channels that are shown under the MQ Channels view).

For example, under the **Total (Queues)** category, you can see a metric called **Depth**. This is the total depth for all the queues. It is obtained by adding up the individual queue depths of the queues shown under the MQ Queues view.

34

BEA WebLogic Views

The **BEA WebLogic** view group displays performance data for BEA WebLogic application servers.

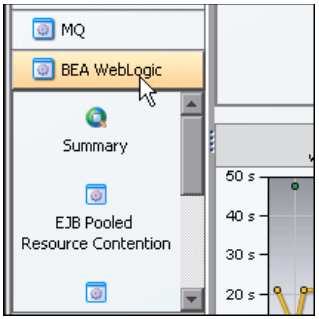
This chapter includes:

Accessing the BEA WebLogic Views

You can access the BEA WebLogic views from the View bar in the Diagnostics views.

To access the BEA WebLogic views:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **BEA WebLogic** to open the BEA WebLogic view group.



If the BEA WebLogic view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **BEA WebLogic** and click **OK**. The BEA WebLogic view group is displayed in the view bar.

3. Select the appropriate view from the BEA WebLogic view group.

Description of the BEA WebLogic Views

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 21, "Status View." For more information about the Server Requests view, see "Aggregate Requests and Server Requests Views" on page 311.

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.

35

IBM WebSphere Views

The **IBM WebSphere** view group displays performance data for IBM WebSphere application servers.

This chapter includes:

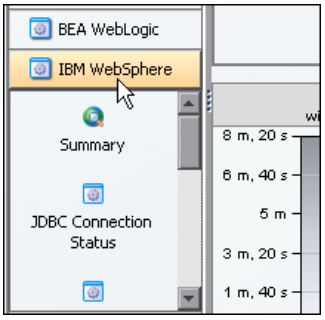
Accessing the IBM WebSphere Views

You can access the IBM WebSphere views from the View bar in the Diagnostics views.

To access the IBM WebSphere views:

1 In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.

2 In the View bar, click **IBM WebSphere** to open the IBM WebSphere view group.



If the IBM WebSphere view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **IBM WebSphere** and click **OK**. The IBM WebSphere view group is displayed in the view bar.

3 Select the appropriate view from the IBM WebSphere view group.

Description of the IBM WebSphere Views

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 21, "Status View." For more information about the Server Requests view, see "Aggregate Requests and Server Requests Views" on page 311.

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.

36

CICS Views

The **CICS** view group displays performance data for CICS (Customer Information Control System) transaction servers. CICS runs primarily on IBM mainframe systems under z/OS.

CICS Transaction Gateway is a Java connector that handles communication between a WebSphere application server and a CICS system.

HP 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.

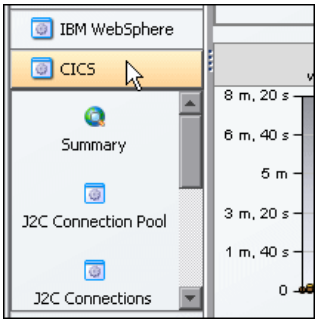
This chapter includes:

Accessing the CICS Views

You can access the CICS views from the View bar in the Diagnostics views.

To access the CICS views:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **CICS** to open the CICS view group.



If the CICS view group is hidden in your application, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **CICS** and click **OK**. The CICS view group is displayed in the view bar.

3. Select the appropriate view from the CICS view group.

Description of the CICS Views

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 "Aggregate Requests and Server Requests Views" on page 311.

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.

37

External Monitors Views

The **External Monitors** view group provides views of data coming from external sources such as SiteScope monitors.

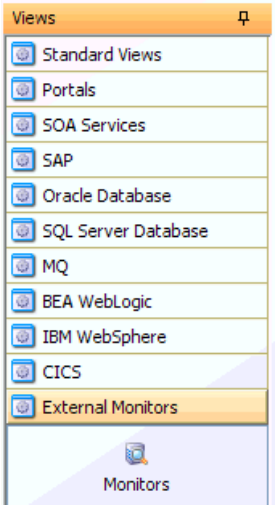
This chapter includes:

Accessing the External Monitors Views

You can access the External Monitors views from the View bar in the Diagnostics views.

To access the External Monitors views:

1. In the navigation pane of the Diagnostics Applications window, double-click an application name (for example, **Entire Enterprise**). The Diagnostics views open.
2. In the View bar, click **External Monitors** to open the External Monitors view group.



If the External Monitors view group is hidden, right-click inside the View bar and select **Open a View Group**. In the Open a View Group dialog box, select **External Monitors** and click **OK**. The External Monitors view group is displayed in the view bar.

3. Select the Monitors view.

Description of the External Monitors Views

The External Monitors view group contains a single view labeled **Monitors**. The Monitors view currently presents data gathered by SiteScope monitors.

HP SiteScope is an agentless monitoring solution designed to ensure the availability and performance of distributed IT infrastructures—for example, servers, operating systems, network devices, network services, applications, and application components. SiteScope can monitor utilization, response time, usage, and resource availability of a variety of host types and application platforms.

Within SiteScope, you create your monitoring structure, configure the Diagnostics integration, and select objects for the integration. SiteScope then forwards data to HP Diagnostics for display in the Diagnostics UI.

SiteScope can forward data to HP Diagnostics that allows the user to see a more complete view of the application servers that are monitored by Diagnostics and/or provide insight into the infrastructure components that these application servers are deployed into. For example, integrating data from the SNMP monitor can help determine problems with the infrastructure that the application server runs on.

SiteScope sends data to Diagnostics from the monitor's counters, along with the name of the monitor, the name of the system that the monitor is targeting, the name of the SiteScope server and the hierarchy of groups that the monitor is defined in.

The Diagnostics integration with SiteScope is configured in SiteScope (see the "Integration Preferences" section of the SiteScope documentation for details).

When configuring SiteScope for integration with Diagnostics it will be important to ensure the metrics sent to Diagnostics conform to the allowable units in Diagnostics. The allowable units within Diagnostics are as follows:

Time Units	Size Units	Numerical Counter	Contribution	Load
microseconds milliseconds seconds minutes hours days	bytes kilobytes megabytes gigabytes	count	percent fraction_percent - (A value in the range 0.0-1.0 which is multiplied by 100 by the UI)	load

If an invalid unit is sent to Diagnostics from SiteScope, it will be defaulted to the count unit.

After configuring the SiteScope integration with Diagnostics (see the SiteScope documentation), and as monitors are associated with the integration, SiteScope generates a file **/SiteScope/conf/integration/data_integration_uom.xml** that controls the mappings of SiteScope monitors to Diagnostics metrics. SiteScope units are captured form the monitored source and will sometimes need to be mapped to the appropriate Diagnostics metric unit. Modify the xml file as needed.

For example:

```
<monitor type="CPU">
      <counter units="%" name="utilization"/>
      <counter units="%" name="utilization cpu # 1"/>
      <counter units="%" name="utilization cpu # 2"/>
      <counter units="%" name="utilization cpu # 3"/>
      <counter units="%" name="utilization cpu # 4"/>
</monitor>
```

Should be modified as follows:

```
<monitor type="CPU">
```

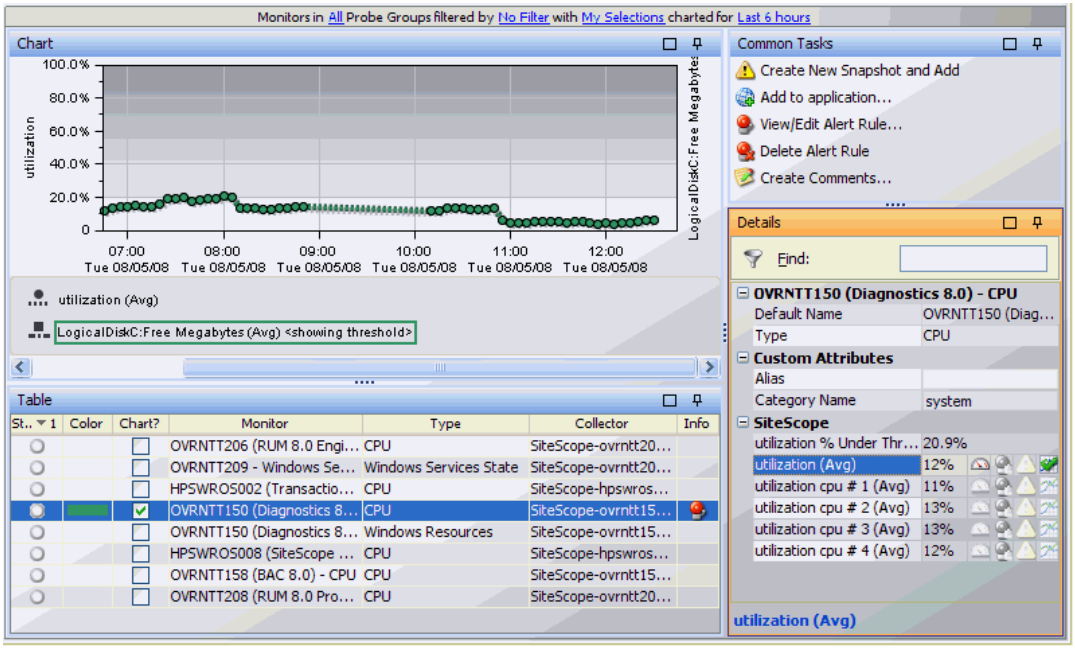
```
    <counter units="percent " name="utilization"/>
    <counter units="percent " name="utilization cpu # 1"/>
    <counter units="percent " name="utilization cpu # 2"/>
    <counter units="percent " name="utilization cpu # 3"/>
    <counter units="percent " name="utilization cpu # 4"/>
</monitor>
```

Sometimes SiteScope is unable to determine an appropriate unit of measure, and sets the units to "unknown". For any counters with a unit of "unknown", SiteScope will not forward the data to Diagnostics. You must modify the **data_integration_uom.xml** file and assign an appropriate unit to the counter before data will flow to Diagnostics for that counter.

Once configured, SiteScope data flows to the Diagnostics servers configured to receive the data after the monitors finish running.

In the Monitors view the SiteScope data is presented in the form of graphs. This SiteScope data can help in understanding of the root cause of the problems identified by the probes (for example, slow method execution) by correlating between slow performance of the application (based on data from the probes) and various metrics provided by SiteScope monitors.

Important: When you first view SiteScope data in the Monitors view, by default the monitor's status is gray and no data is graphed. In order to see a status (red, yellow, green) you will need to set a threshold on a metric (in the details pane). In order to see data in the graph you will need to select a metric to be charted (in the details pane).



To display graph data and see monitor status in this view:

1. Check the Chart? column for a monitor in the graph entity table.
2. Select a metric in the details pane and check the icon to **Start Charting this Metric**.
3. To see monitor status, select a metric in the details pane and set a threshold. You'll note when setting a threshold in this view that there aren't any default threshold values for the SiteScope metrics. In order to set a threshold, in the Set Threshold dialog box, select **Use custom threshold** and define the threshold you want set for the SiteScope metric.

Table columns in the Monitors view include:

Monitor. The name of the SiteScope monitor configured by the SiteScope administrator.

Type. The name of the metric type configured by the SiteScope administrator.

Collector. The SiteScope server displayed as SiteScope-<SiteScope_server_name>.

You can set threshold, metric alert rules for the SiteScope metrics and add SiteScope metrics to snapshots as you would other Diagnostics metrics.

Common tasks you can select for monitors include: Add to application, Create/View/Edit/Delete (entity status) alert rule, Create/View/Edit/Delete comments. You can select Add to Snapshot, which is useful if you want to view SiteScope data combined with probe data.

You will also see the monitors listed under probe groups in views such as the Server Summary view. Monitors are associated with Diagnostics probe groups whose names are derived from the names of the groups and sub-groups in which the monitor is defined in SiteScope. The group hierarchy is represented as a path. For example, a monitor in the "Windows" sub-group of the "Servers" group will be in the "Servers/Windows" probe group in Diagnostics.

Note: You may see some of the same metrics in both the Monitors view (displaying SiteScope data) and Diagnostics views. The difference between the SiteScope and Diagnostics view of the same metrics has to do with the frequency of the measurements and aggregation.

Part V

Using the Diagnostics Profiler for Java

38

Using the Java Diagnostics Profiler

The Diagnostics Profiler for Java is installed with the Java probe. The Profiler runs in a separate UI and provides near real-time data, enabling you to pinpoint application performance bottlenecks.

This chapter includes:

Note: The Diagnostics Profiler for Java 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 five concurrent threads.

For more information about licensing, refer to the *HP Diagnostics Installation and Configuration Guide*.

If you installed the probe from the HP Web site and you want to use it with a Diagnostics Server, contact HP Support.

Accessing the Diagnostics Profiler for Java

Once you have installed and configured the Java Probe and you have started the application that is being monitored, you can access the Java Diagnostics Profiler from your browser and view Diagnostics data. You can also access the Java Diagnostics Profiler by drilling down from the views of the HP Diagnostics user interface.

To view Diagnostics data from the Java Diagnostics Profiler:

1 In your browser, go to the Java 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 *HP Diagnostics Installation and Configuration Guide*.

To drill down to the Diagnostics Profiler from HP Diagnostics:

➤ From any Status screen in HP Diagnostics, right-click the probe entity and select **View Profiler for <probe name>** from the menu.

➤ Alternatively, in the standard Probes view in HP 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.

Diagnostics Profiler for Java Processing

This section describes the way in which the Java Probe monitors your application and how this data is displayed in the Java Diagnostics Profiler.

Monitoring Method Latency and Call Stacks

The Diagnostics Probe for Java (Java 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 Java 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 Java 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 Java Probe continues to monitor your application, capture method metrics, and capture call stacks.

For more information on monitoring method latency with the Java Diagnostics Profiler, see Chapter 39, “Analyzing Method Latency with Java Diagnostics Profiler.”

Monitoring Application Memory

The Java Diagnostics Profiler allows you to monitor your application's memory usage using one of the following methods:

- Light Weight Memory Diagnostics
- Heap Breakdown

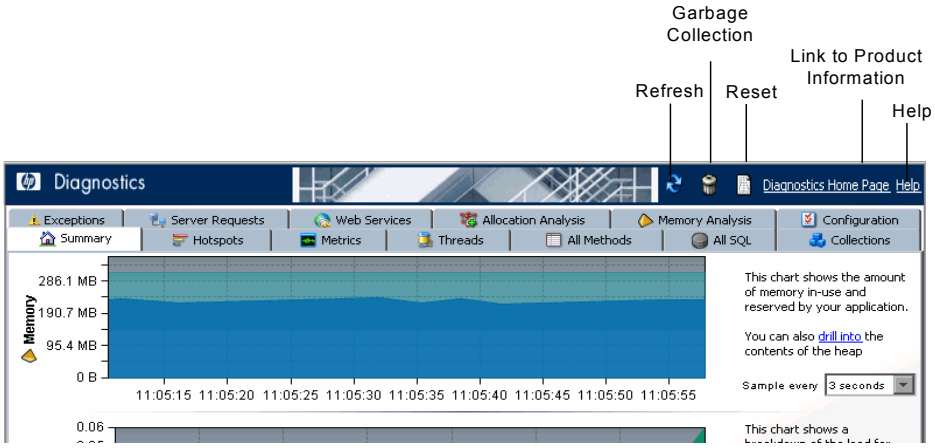
Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you 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 570.

For more information on Heap Breakdown, see "Analyzing Memory Using the Heap Breakdown Tab" on page 583.

Common Java Diagnostics Profiler Tab Navigation and Display Controls

This section describes the features and controls that are common to the tabs of the Java 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 Java 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.

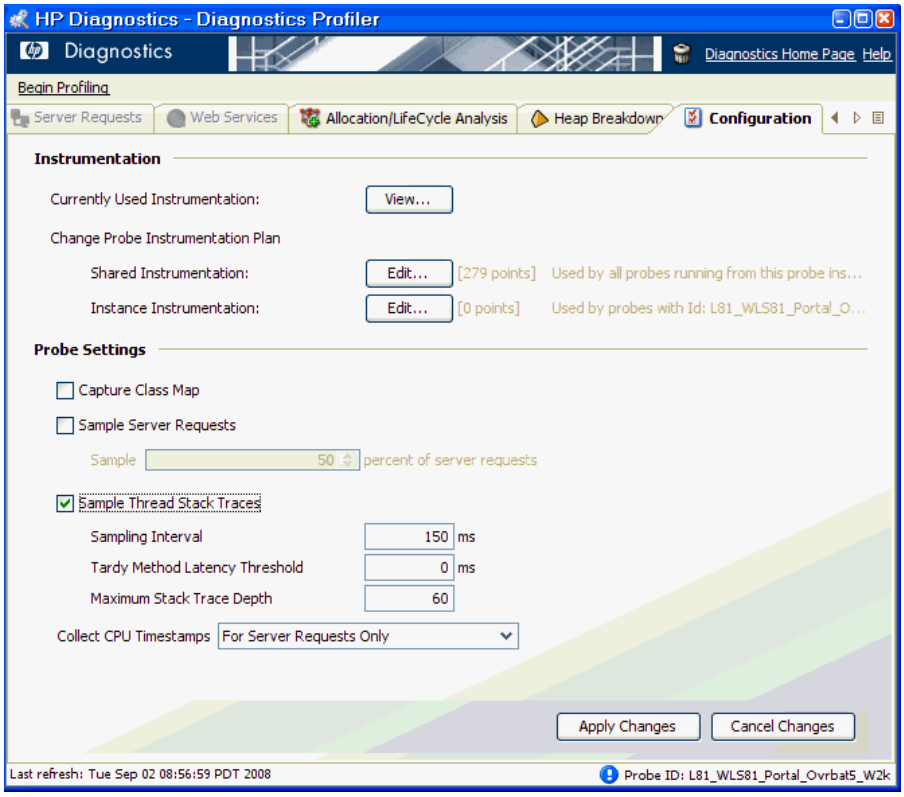
Accessing Help

When you click **Help**, on the top right hand corner of the screen, you access the on-line help manual for the Diagnostics Profiler for Java.

Configuring the Java Probe Using the Profiler

You can use the Configuration tab from within the Java Diagnostics Profiler to view and modify the instrumentation and adjust the settings for the Java Probe.

The Configuration screen is divided into two parts as shown in the following image:



The Instrumentation part of the Configuration screen allows you to view and update the instrumentation of your application.

The Probe Settings part of the Configuration screen allows you to control the overhead of the Probe and the level of detail for the performance metrics captured by adjusting the method trimming and class map capture.

Instructions for using the Instrumentation part of the Configuration screen can be found in the *HP Diagnostics Installation and Configuration Guide* chapter on Instrumenting Java Applications from the Profiler.

39

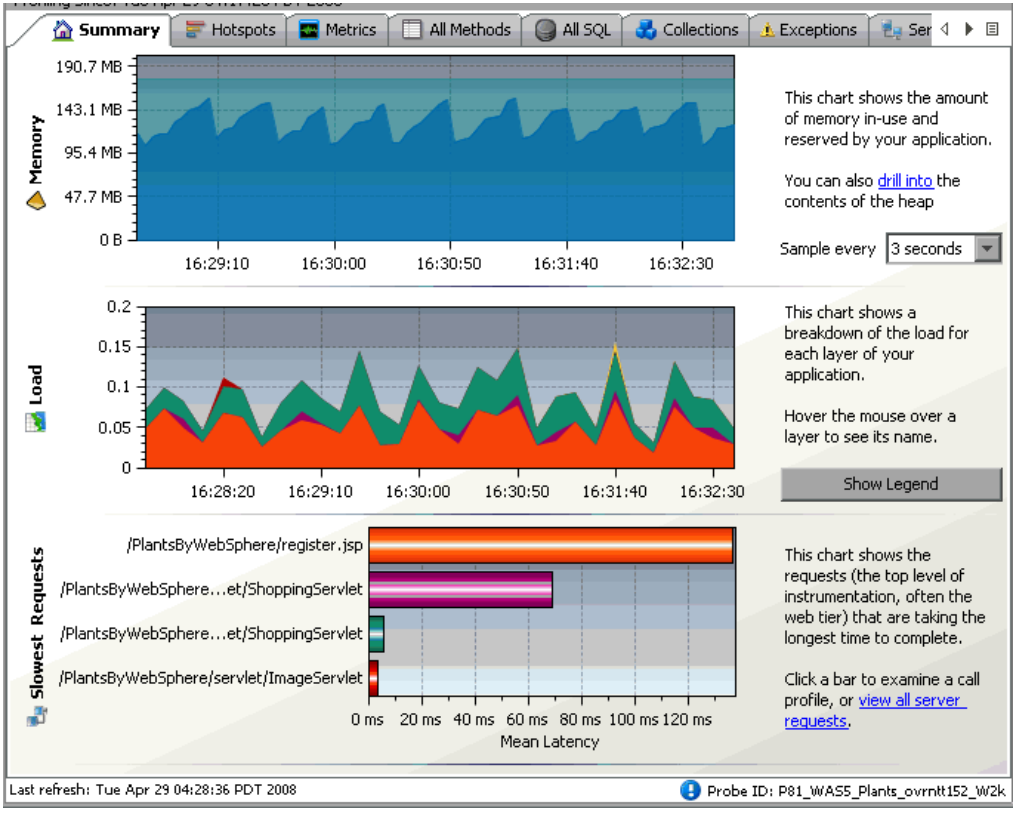
Analyzing Method Latency with Java Diagnostics Profiler

You can use the different tabs in the Java Profiler to analyze method latency for the selected application.

This chapter includes:

Analyzing Performance Using the Summary Tab

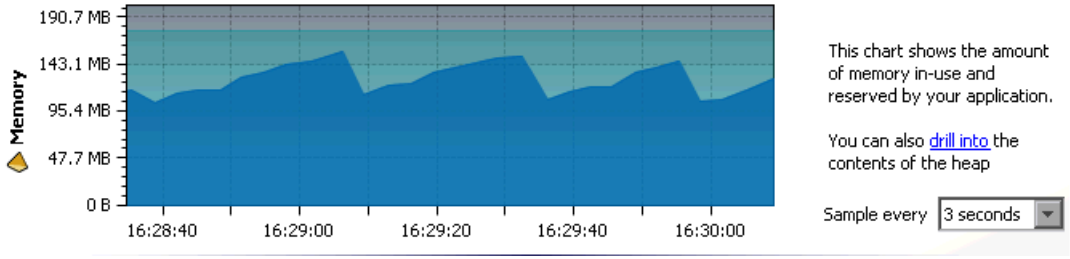
The Summary tab consists of graphs that display information about the memory in use and reserved by your application, the load for each layer of your application and the slowest requests made to your application server.



The Summary tab is divided into the following sections.

Memory Graph

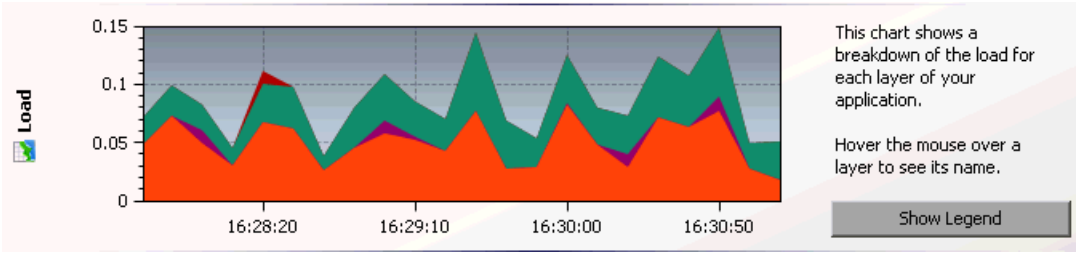
The Memory graph shows the amount of memory allocated in your application and the amount of memory (JVM heap size) reserved by your application.



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

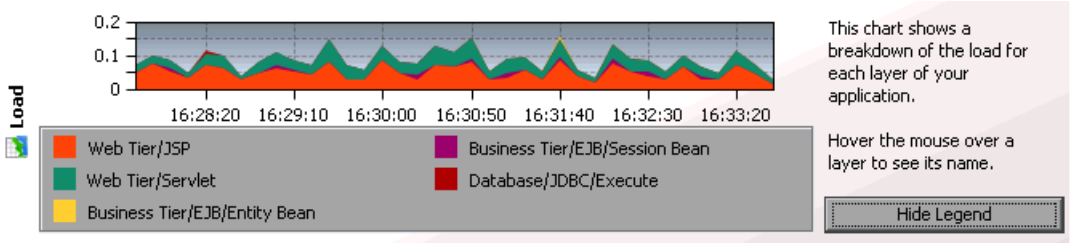
The Load graph shows the breakdown of the load for each layer of your application.



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 Java Diagnostics Profiler displays the layers on one level and does not split them into sublayers.

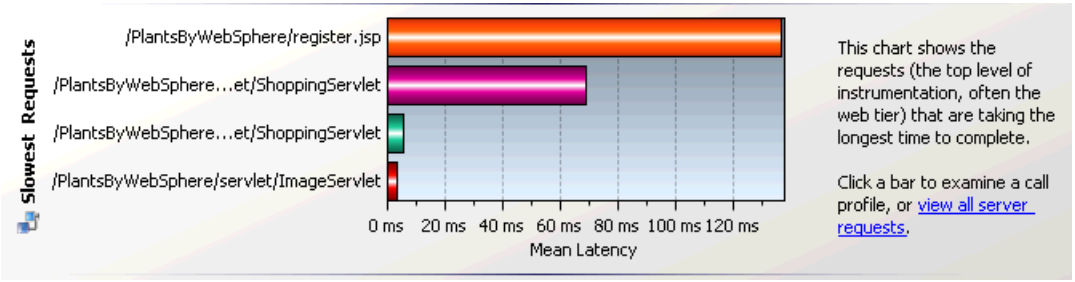
You can see the name of each layer by holding your pointer over various points on the graph to view the tooltip.

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 535.

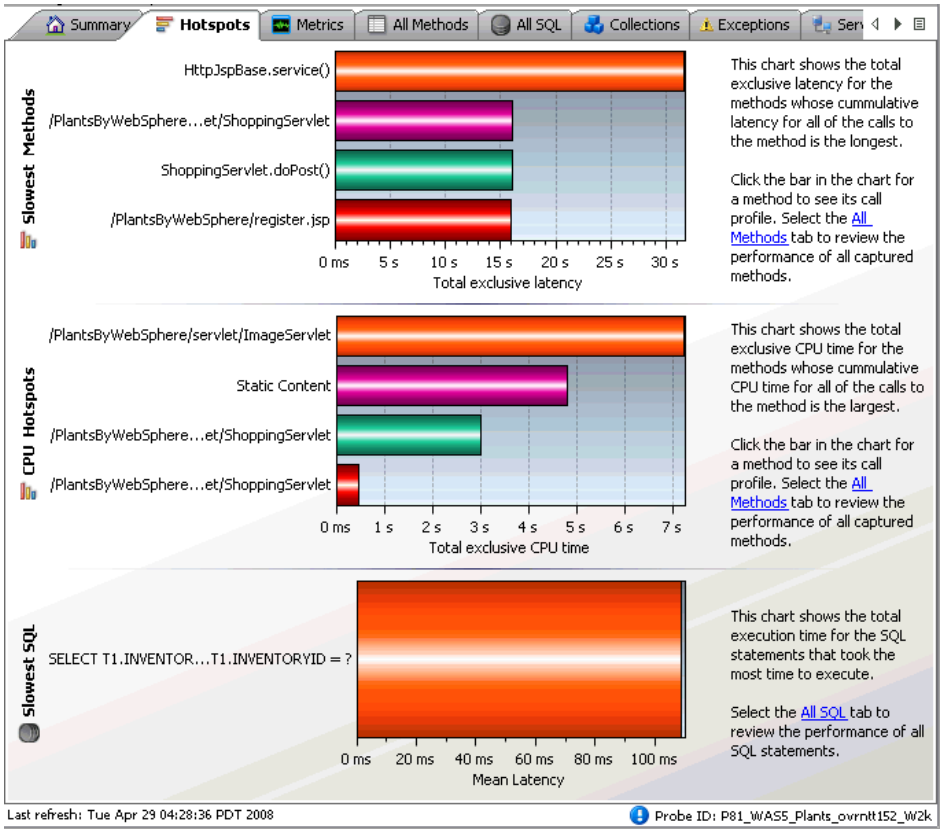
Information Pane

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 probe ID.

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.



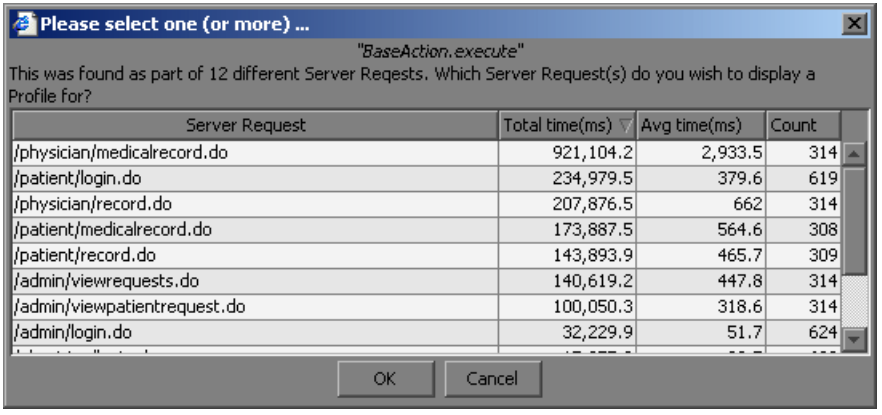
Note: You can view the details for a graphed metric by holding your pointer over the bar for the metric and viewing the tooltip.

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 555.

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 555.

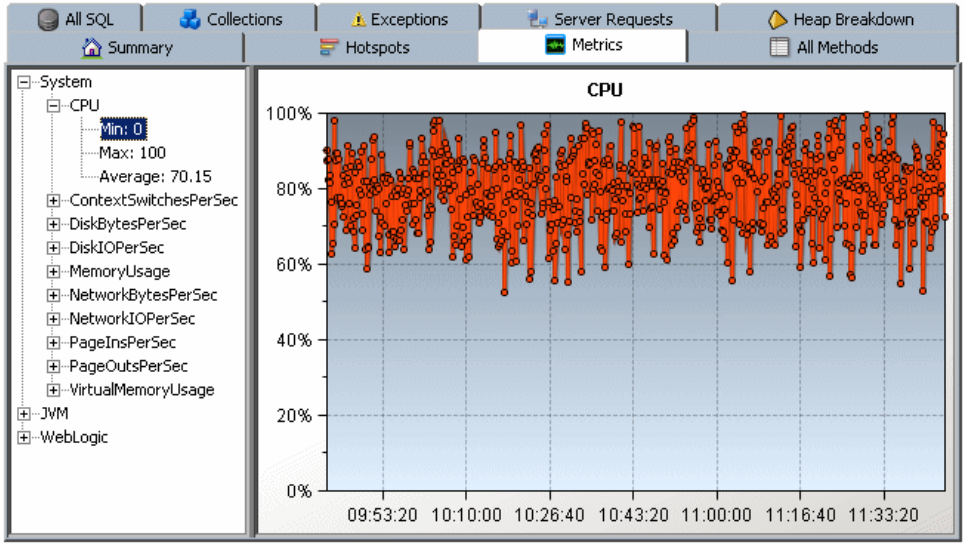
Slowest SQL Graph

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 548.

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 HP Software 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.

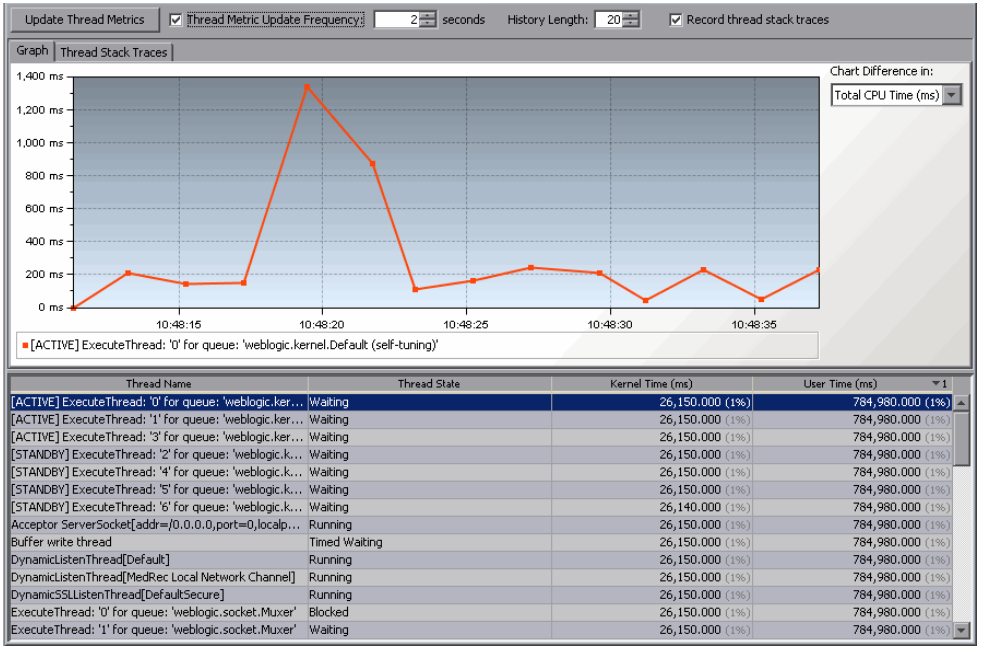
Note: This Threads tab is only displayed when the application is running on JDK 1.5 and above.

Understanding the Threads Tab

The Threads tab is divided into four parts:

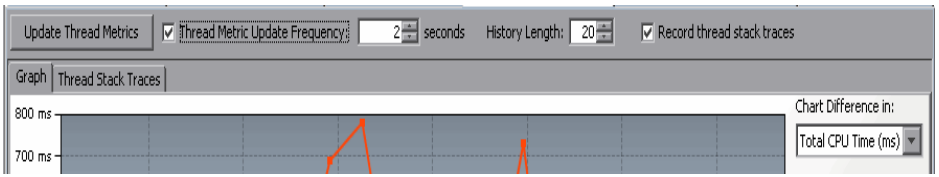
- **Controls.** Where you indicate how much, how often, and what kind of data is captured and displayed for the thread processing in your application.
- **Thread Table.** Where the metrics for each thread are listed.
- **Graph tab.** Where the metrics for the threads selected from the threads table are 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 Threads 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:

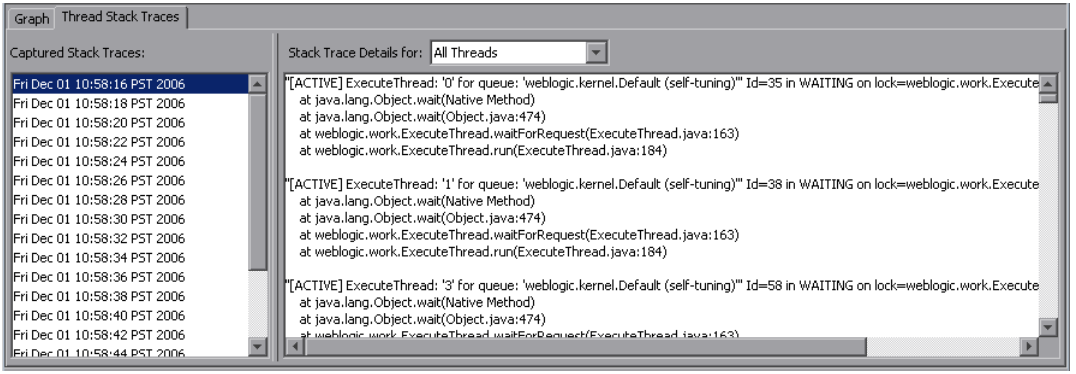
- Select the row in the thread table that contains the thread whose metrics you would like to remove from the chart using **Ctrl-Click.**

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 displays 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.

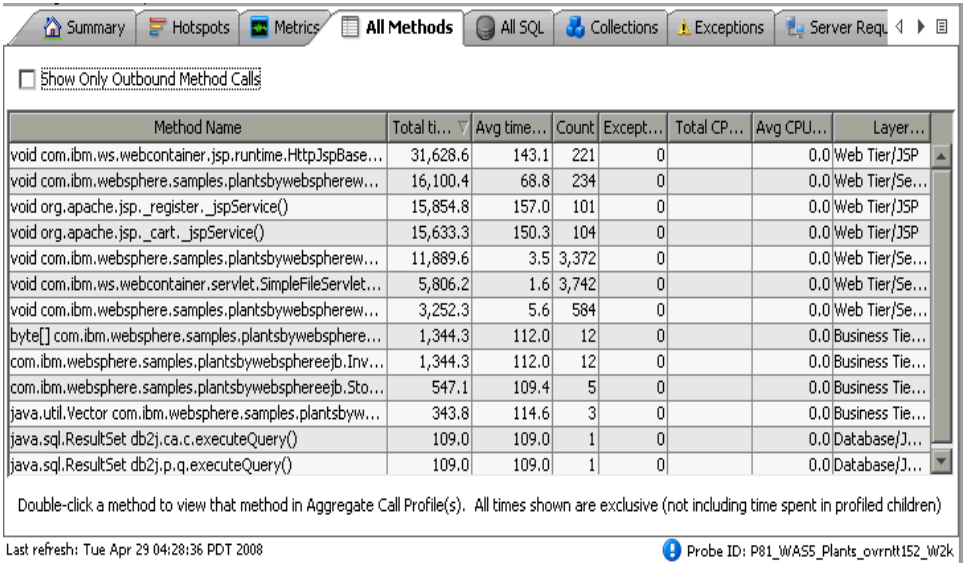
- The Captured Stack Traces list contains a list of the times when stack trace captures occurred.

- 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 542.

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.



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.
- **Avg CPU.** The CPU time that the method used during an average invocation.

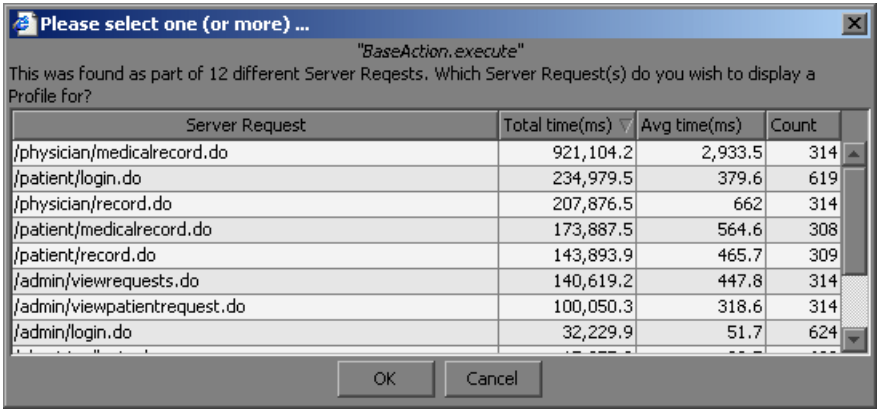
  If CPU time metrics are not being displayed, CPU Timestamp collection for methods can be configured (see "Collection of CPU Time Metrics" on page 141 for details).
- **Layer.** The Layer associated with this method according to the instrumentation in the **auto_detect points** file. The layers are displayed on one level and there is no distinction made between layers and sub-layers.

Note: All CPU times shown are exclusive (not including time spent in profiled children).

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 see "Analyzing Performance Using the Call Profile Window" on page 555.

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:

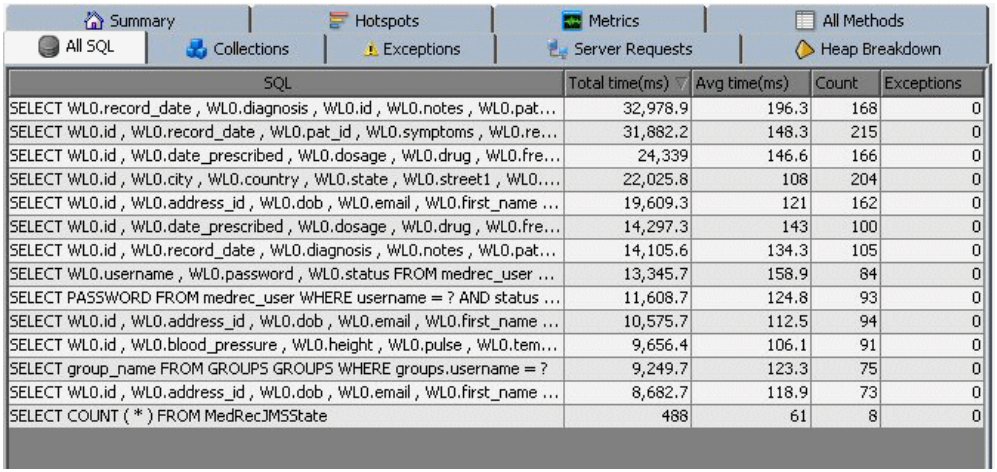


Double-click the appropriate server request row to view the call profile.

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 WL0.record_date , WL0.diagnosis , WL0.id , WL0.notes , WL0.pat...	32,978.9	196.3	168	0
SELECT WL0.id , WL0.record_date , WL0.pat_id , WL0.symptoms , WL0.re...	31,882.2	148.3	215	0
SELECT WL0.id , WL0.date_prescribed , WL0.dosage , WL0.drug , WL0.fre...	24,339	146.6	166	0
SELECT WL0.id , WL0.city , WL0.country , WL0.state , WL0.street1 , WL0....	22,025.8	108	204	0
SELECT WL0.id , WL0.address_id , WL0.dob , WL0.email , WL0.first_name ...	19,609.3	121	162	0
SELECT WL0.id , WL0.date_prescribed , WL0.dosage , WL0.drug , WL0.fre...	14,297.3	143	100	0
SELECT WL0.id , WL0.record_date , WL0.diagnosis , WL0.notes , WL0.pat...	14,105.6	134.3	105	0
SELECT WL0.username , WL0.password , WL0.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 WL0.id , WL0.address_id , WL0.dob , WL0.email , WL0.first_name ...	10,575.7	112.5	94	0
SELECT WL0.id , WL0.blood_pressure , WL0.height , WL0.pulse , WL0.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 WL0.id , WL0.address_id , WL0.dob , WL0.email , WL0.first_name ...	8,682.7	118.9	73	0
SELECT COUNT (*) FROM MedRecJMSState	488	61	8	0

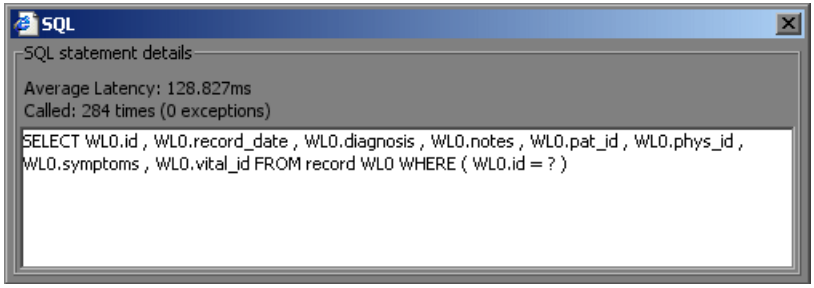
Understanding the All SQL Tab

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

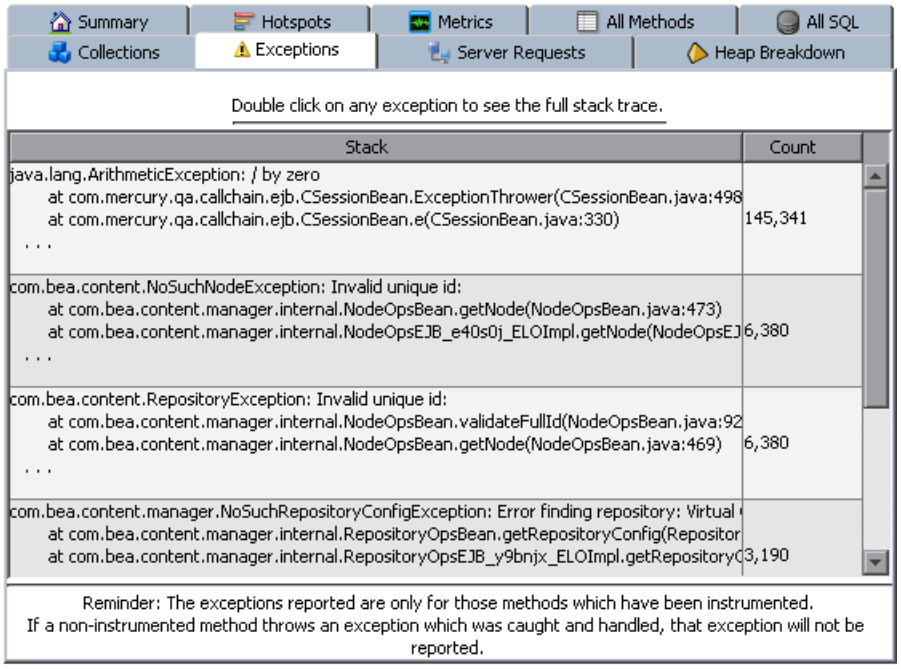
To view the SQL statement details, double-click on the relevant statement. The SQL statement details dialog box opens, displaying all the information shown in the SQL table for each statement.



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.



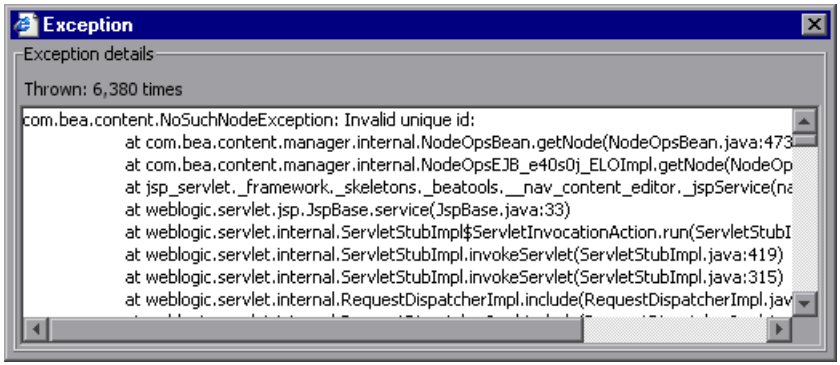
Note: If a non-instrumented method throws an exception which was caught and handled, that exception will not be reported.

Understanding the Exceptions Tab

The Exceptions tab displays the Exceptions table which contains the following columns:

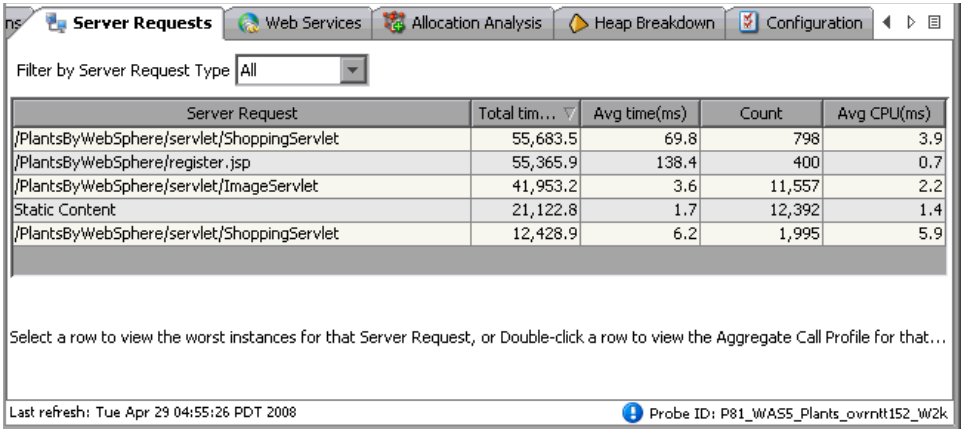
- **Stack.** Shows the first three lines of the exception stack trace.
- **Count.** The number of times the exception was generated.

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.



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 555.

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 Customer Support.

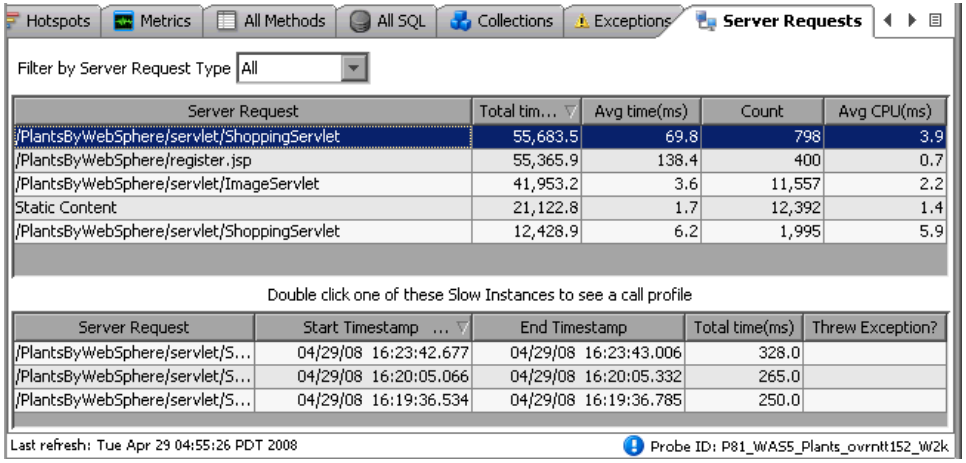
- **Total Time.** The total latency of all invocations of the server request.
- **Average Time.** The Average latency of all invocations of the server request.
- **Count.** The number of times this server request was invoked.
- **Avg CPU.** The CPU time that the method used during an average invocation.

  If CPU time metrics are not being displayed, CPU Timestamp collection for methods can be configured (see "Collection of CPU Time Metrics" on page 141 for details).
- **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 see "Analyzing Performance Using the Call Profile Window" on page 555.



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 (accessed from the Server Requests tab) displays a graphical representation of the method call stack for a selected server request. The depicted server request can be an aggregation of all of the calls made to the selected server request or a single instance of the server request depending on the server request on which you drilled down to open the call profile window. The metrics depicted in the graphical representation of the call stack are also depicted in the Call Tree Table on the same tab.

Types of Call Profile Windows

There are two types of call profile windows that are displayed depending on the how you navigated to the tab:

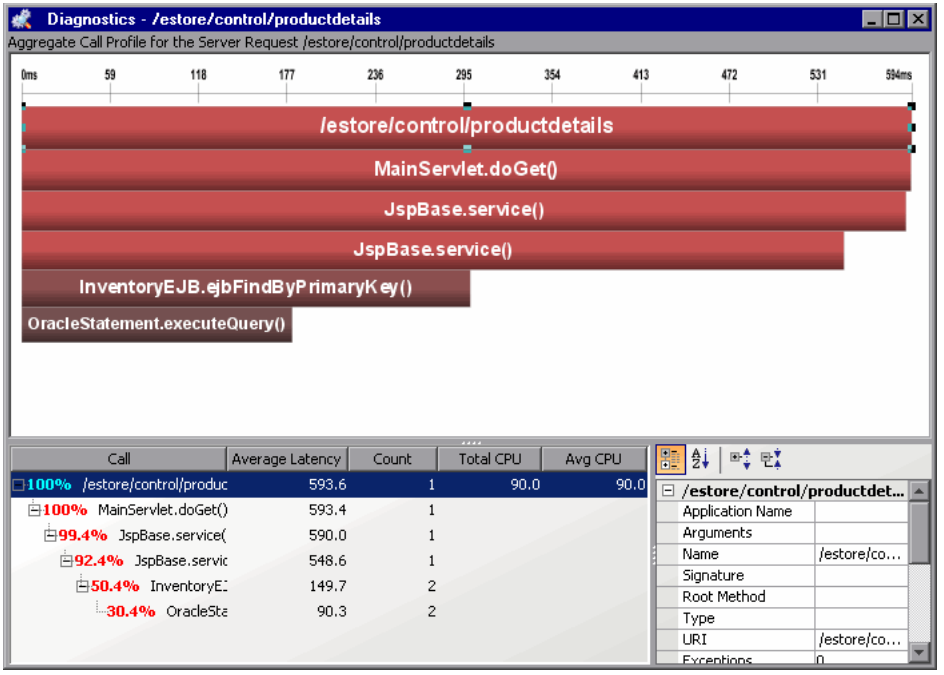
- **The Instance Call Profile window** displays the method calls that were made during the processing of the server request on which you drilled down.
- The **Aggregate Profile window** displays an aggregation of all of the method calls that were made during the processing of all of the server requests that were the same as the one on which you drilled down.

Description of Call Profile Window

The Call Profile Window is made up of three areas:

- Call Profile Graph graph
- Call Tree Table
- Details Pane

The following image is an example of an Aggregate Call Profile Window showing all three of these areas:

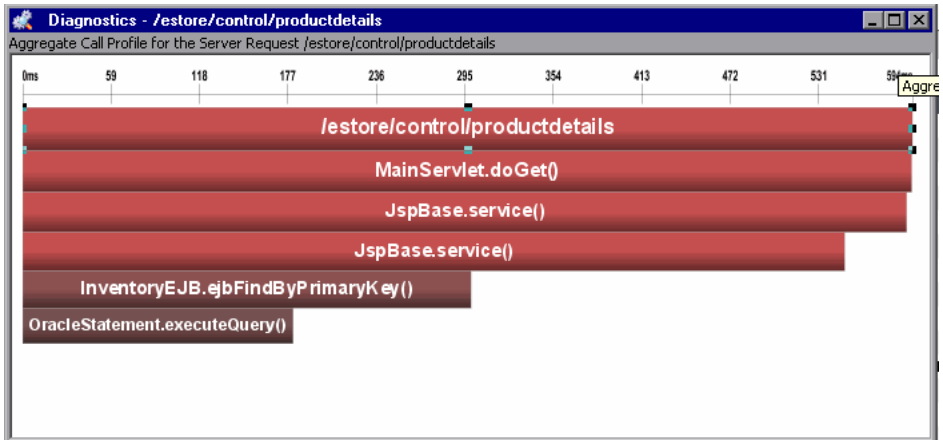


When you click a call box in the Call Profile graph, the corresponding row is selected in the Call Tree table and the metrics for the selected call are displayed in the Details pane. When you click on a row in the Call Tree table the corresponding call box in the Call Profile graph is selected and the 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.

Thread Call Stack Trace Sampling

When asynchronous thread sampling is enabled you can see additional nodes added into the call profile graph by sampling. These nodes are distinguished by their different (fuzzy) shading to emphasize lack of data about the represented method start and end times. See the *HP Diagnostics Installation and Configuration Guide* chapter on Advanced Java Probe and Application Server Configuration for more information on how to configure thread sampling. Also refer to the *HP Diagnostics User's Guide* Call Profile View chapter under the section on Asynchronous Thread Call Stack Sampling.

Instrumenting a Sampled Method Dynamically

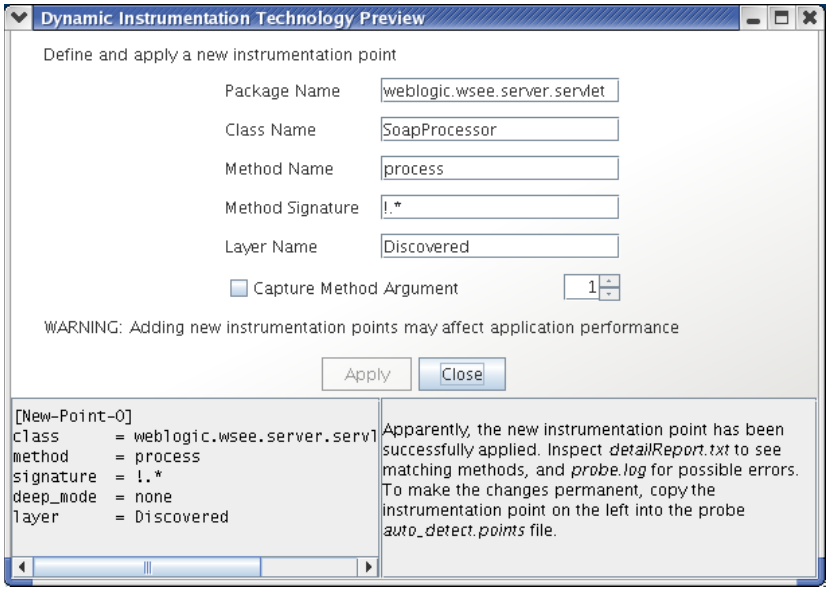
Non-instrumented methods displayed in the Call Profile when Thread Stack Trace Sampling is enabled give you an insight into the call hierarchy and latencies of these methods. But you may want to identify one of these methods to instrument in order to get additional detail information.

Dynamic instrumentation is Java bytecode instrumentation performed during the application execution *after* the respective class has been first loaded by the Java Virtual Machine. Instrumentation is temporary, for the current Java process. If you want to permanently instrument this method you must add the point you created to the instrumentation points file.

Important: Dynamic instrumentation (the **Instrument** menu item) is ONLY available for sampled data and you need to access the Call Profile from the Diagnostics Profiler for Java. It is NOT available when accessing the Call Profile from an instance tree icon in the main Diagnostics UI.

From the standalone Java Profiler UI, select the Server Requests tab and open the Call Profile window. Select the sampling (fuzzy) node in the Call Profile window and right-click to select **Instrument**.

The Dynamic Instrumentation Technology Preview window is displayed with values corresponding to the selected method.



You can change the package, class, and method name, and provide a method signature, if known, to narrow down the scope of the instrumentation. Since sampling does not reveal method signatures, by default all methods with matching names will be instrumented.

You can also chose to capture one of the method arguments, but you should only do this if you know that the 'toString()' method can be safely called on the selected argument. An incompatible argument may cause instrumentation errors, or even runtime errors.

Note: The classes belonging to the Diagnostics Java Probe or the Java runtime cannot be instrumented.

Click **Apply** after making the changes you want and the Java Probe automatically creates a new point definition and tries to apply the instrumentation dynamically. The bottom part of the dialog window contains the result of this operation: the new instrumentation point definition is placed on the left side, while the result of instrumentation is located on the right.

Once the instrumentation is successful, you should copy and paste the instrumentation point to save it because when you refresh the Call Profile, the Dynamic Instrumentation window with the details on the instrumentation point you created is no longer available.

When you refresh the Call Profile view, the dynamically instrumented method will be displayed as a solid node because it is now instrumented. Instrumentation is temporary for the current Java process.

If you want to permanently instrument this method you must add the point you created to the instrumentation points file.

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 Tootips

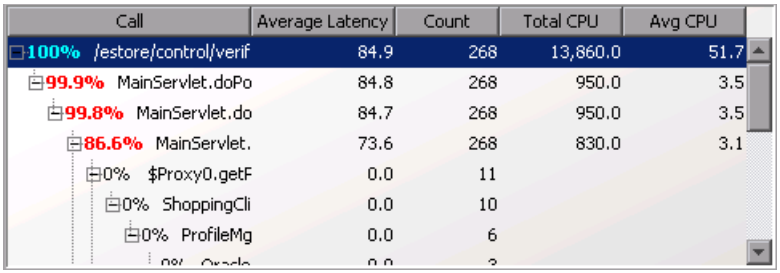
If the duration of a call is very short or if the call appears further down in the call stack, the size of the call box can cause the name of the method that the call box represents to become too small to read. You can view the name of the method along with other details for a selected method by holding 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
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.



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.

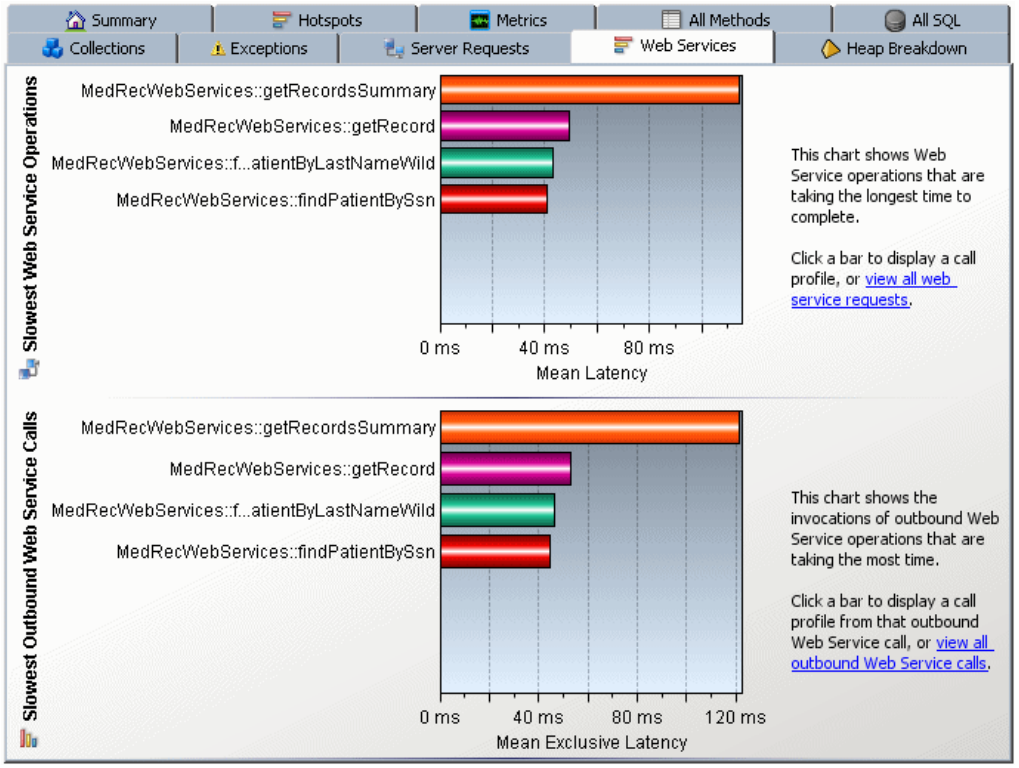
⊟ **/estore/control/cart**	
Application Name	
Arguments	
Name	/estore/control/cart
Signature	
Root Method	
Type	
URI	/estore/control/cart
Exceptions	0
Count	1
Timeouts	0
⊟ **Latency**	
Total CPU	80.0 ms
Exclusive Total Latency	0.1 ms
Max. Latency	258.4 ms
Min. Latency	258.4 ms
Standard Deviation	0.0 µs
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**.

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 Java 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 555.

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 552.

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 Java 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 555.

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 545.

Using the Configuration Tab

The Configuration tab in the Java 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 "Configuring the Java Probe Using the Profiler" on page 529

40

Analyzing Memory with Java Diagnostics Profiler

You can analyze memory problems for the selected application using the memory diagnostics metrics displayed in the Java Profiler.

This chapter includes:

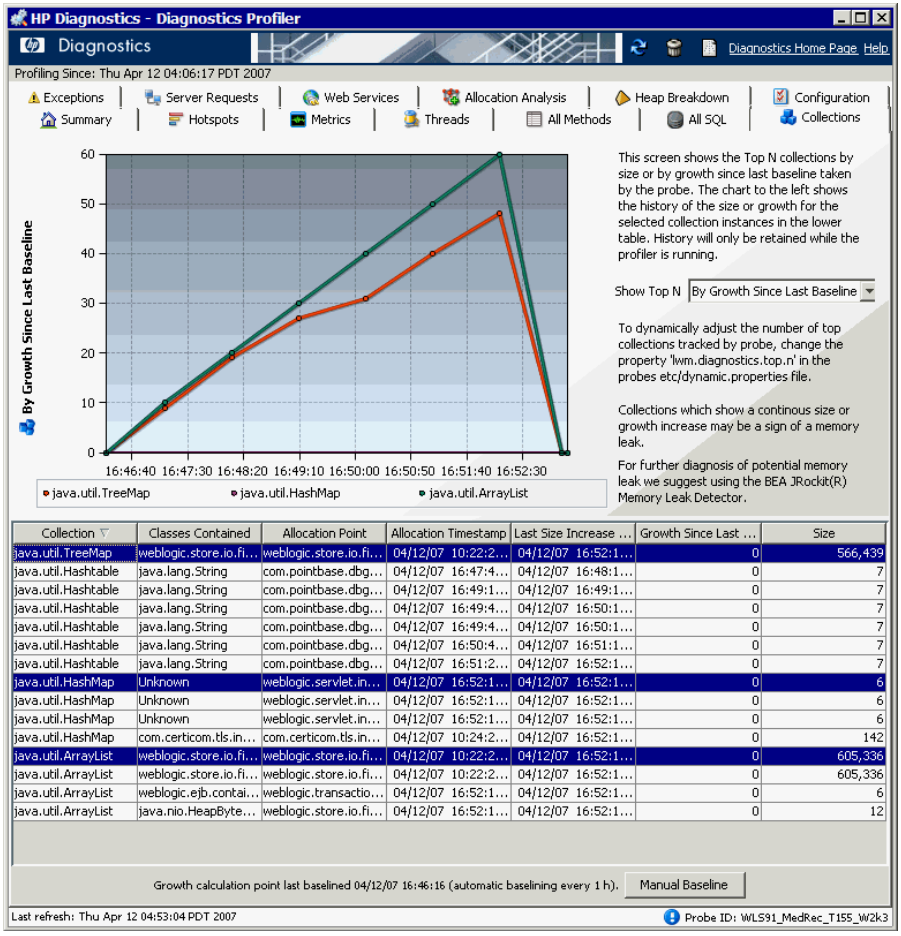
Analyzing Memory Using the Collections Tab

The Java Diagnostics Profiler monitors your applications' memory usage with 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 corresponding table. The table lists the collections, information about the allocation of the collections, and the metrics for their 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. The Profiler updates the baseline periodically. If you want, you can update it manually.



The Collections tab is divided into two parts—the collections table and the collections graph.

Collections Table

The collection table lists the collections. The collections are sorted by either the amount of growth since the last baseline, or by the size of the collection, depending on the selection 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 you make 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 the Java Probe does 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:

- Turn on the LWMD capture in the **dynamic.properties** file by setting the **lwm.diagnostics.capture** equal to **true**:

  lwm.diagnostics.capture=true.

➤ Activate the LWMD point in the **auto_detect.points** file by setting **active** equal to **true**:

```
[Light-Weight Memory Diagnostics]
keyword = lwmd
scope = !only\.in\.this\.Class\..*,!or\.in\.this\.Class\..*
active=true
```

It is very important to limit the scope of the LWMD instrumentation to a particular package to reduce overhead. The syntax starts with an exclamation point (!) to indicate that a regular expression follows.

Controlling the Charted Metrics and Table Sort Order

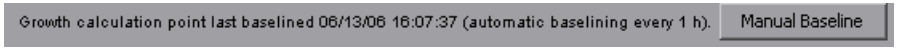
Your selection from the **Show Top N** box controls the metrics that are charted in the collections graph as well as the sort order of the rows in the collections tables.

When you choose **By Size,** the collection table is sorted in descending order by collection size. 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. 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 figure:



The Profiler automatically sets a new baseline at preset periodic intervals. You can also set a new baseline manually.

To set a new baseline manually, 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: You do not 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.baseline.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 and Object Lifecycle - Allocation/ Lifecycle Analysis Tab

The Allocation/Lifecycle Analysis tab can be used for:

- **Allocation Analysis.** Use the information displayed to investigate a memory leak that you have observed in the Heap Breakdown tab by examining the allocation and de-allocation of objects while the leak is happening. See the following topics related to allocation analysis:
  - "Understanding the Allocation/Lifecycle Analysis Tab" on page 575
  - "Enabling the Allocation Capture" on page 578

- "Analyzing Object Allocations Using the Allocation/Lifecycle Analysis Tab" on page 579

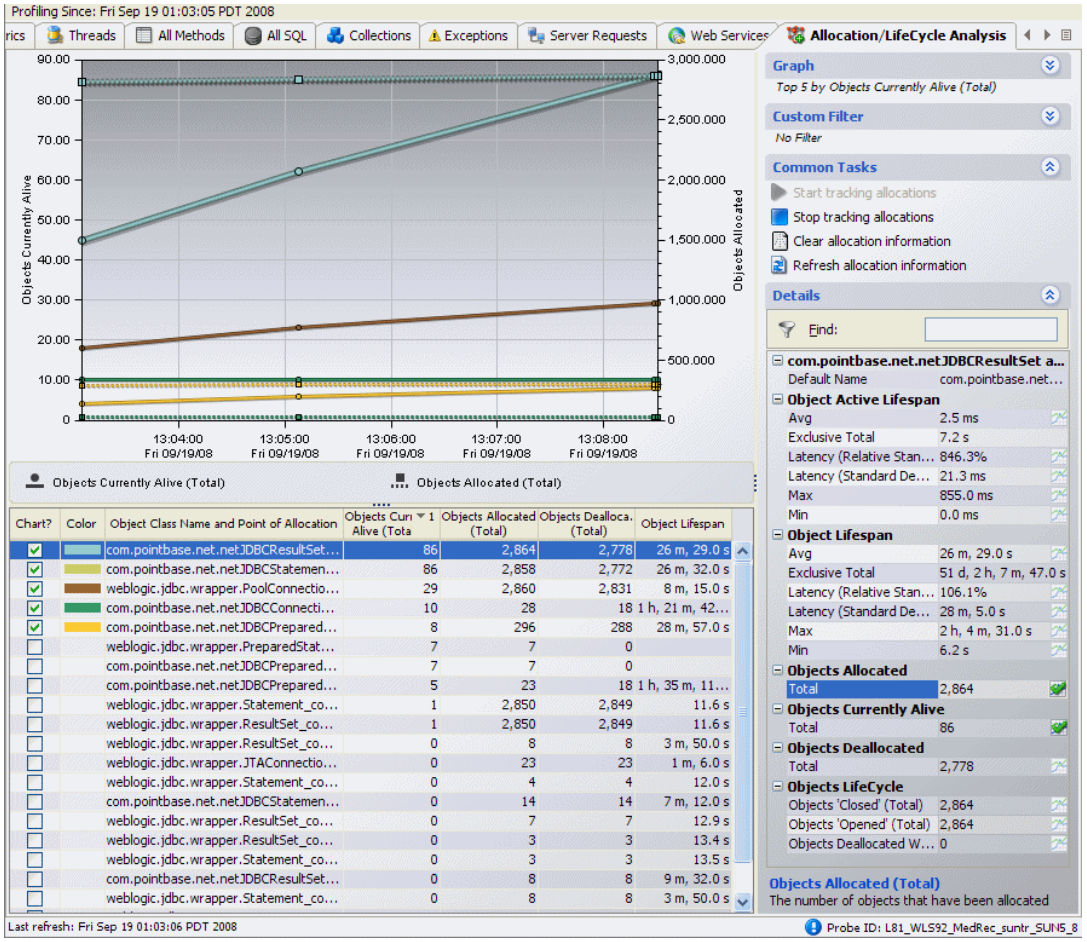
- **Lifecycle Analysis.** Use the information displayed to monitor object lifecycles. This feature can be used for resource monitoring of certain database resources. See the following topics related to object lifecycle analysis:
  - "Object Lifecycle Monitoring" on page 580

Understanding the Allocation/Lifecycle Analysis Tab

The Allocation/Lifecycle 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 figure shows an example of the Allocation/Lifecycle Analysis tab:



The Allocation/Lifecycle Analysis tab is similar to the views with a detail layout in the Diagnostics views. Instead of appearing in the view title, the view filters appear in view filter menus, along the side of the graph. The Common Tasks menu controls the tracking of the allocations as well as the refreshing of the information that is displayed for the entire view.

Allocation/Lifecycle Analysis Table

The allocation/lifecycle analysis graph-entity table lists the objects that have been allocated since you started tracking allocations. You can customize this table 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. By default, the table is sorted in order by Objects Currently Alive. It 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 clicking on the box in this column manually. Or you can let the Profiler select the objects to chart dynamically, using the criteria that you specify in the Graph filter.
- **Color.** Indicates the color that the Profiler uses to chart the metrics for the allocated object. No color is shown for metrics that are not 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 de-allocated objects. If no objects have been de-allocated, this column is blank.

In the Details pane, metrics for **Objects Lifecycle** and **Object Active Lifespan** are also available. See “Object Lifecycle Monitoring” on page 580 for more information.

Allocation/Lifecycle Analysis Graph

The allocation/lifecycle analysis graph charts the metrics that you selected from the details table for each of the objects selected in the allocation/ lifecycle 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.

Enabling the Allocation Capture

The Allocation/Lifecycle Analysis tab cannot display allocation objects or their metrics until allocation capture has been enabled for the probe. By default, allocation capture is disabled, so the Java Probe does not impose the additional overhead on its host when you are not going to use memory diagnostics metrics. If you suspect that you may have a memory issue with the way 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 instruments the **Class.newInstance**, **Constructor.newInstance**, and **Object.clone** methods. The reflection instrumentation tracks all classes that are created.

2 Restart the monitored application, so the probe restarts and can apply the updated instrumentation.

Analyzing Object Allocations Using the Allocation/ Lifecycle Analysis Tab

After 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 procedure describes 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 578.
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 de-allocated. 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 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 after you stop tracking. The instances of the objects that were already allocated continue to be tracked as they are de-allocated, so the metrics on the tab can be refreshed with accurate counts of the objects that are alive or de-allocated, as well as with accurate object lifespans.
5. Select the **Refresh Allocation Information** menu option to update the tab with the current metrics for the allocated objects.

   Each time 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.

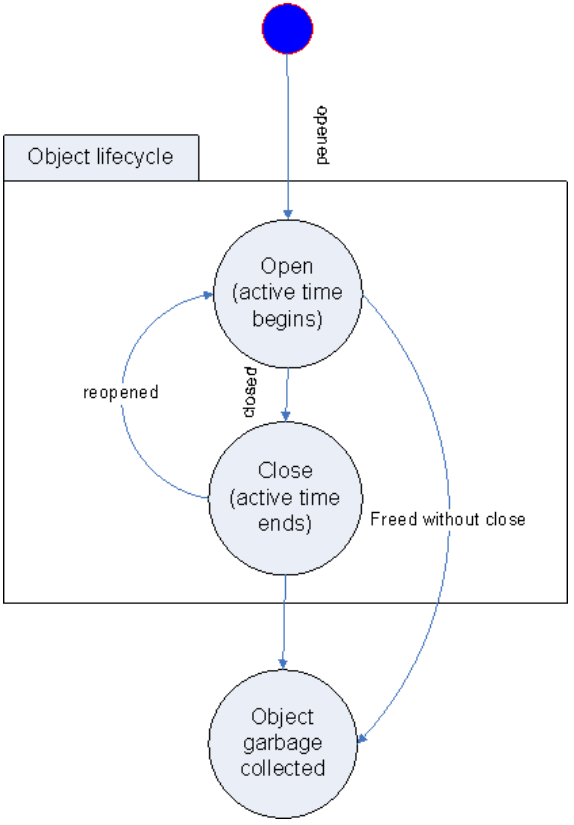
   You should repeat this step as your application continues to run, so you can see what happens to the allocated objects over time.

6 If you want to run your experiment again, select the **Clear 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.

Object Lifecycle Monitoring

Every object has a lifespan. The lifespan begins with object construction and ends with its garbage collection. You can use Allocation Analysis to monitor and analyze object lifespan (see "Analyzing Object Allocations Using the Allocation/Lifecycle Analysis Tab" on page 579).

However, some objects follow a **lifecycle** during their lifespan. For example, the objects representing database resources (like database connection or cursors) go through such a lifecycle during their lifespan. See the diagram below:



These objects are brought into an *open* state by some resource acquisition operation and then *closed* after their usage. They usually acquire their resources before entering an open state and relinquish their resources after reaching a close state. Some of these objects are designed for re-use (for example, objects based on connection pool). So these objects might be re-opened and closed multiple times during their lifespan.

You can enable object lifecycle monitoring in Diagnostics and view lifecycle information for these objects in the Allocation/Lifecycle Analysis tab.

Two Examples of These Types of Objects

- **Database connection:** An object of type java.sql.Connection represents a database connection. The connection is opened by invoking javax.sql.DataSource.getConnection() method and it is closed by invoking java.sql.Connection.close() method.
- **Database cursors:** An object of type java.sql.ResultSet represents a database cursor. The cursor is opened by invoking java.sql.Statement.executeQuery() method and is closed by invoking java.sql.ResultSet.close() method.

Types of Performance Problems with These Objects

Diagnostics allows you to monitor the object's lifecycle between its open and close states to identify the following types of performance problems.

- **Resources are not released:** This problem arises when the object is not brought into a close state. This causes the resources attached to the object to be wasted for the lifetime of the object.
- **Resources are not released in a timely manner:** This problem arises when the resources are released after unnecessarily keeping them around for quite long period of time. This can also happen if the object is not closed but the garbage collector automatically closes it during object finalization.

Enabling Object Lifecycle Monitoring

Object lifecycle monitoring in Diagnostics is not enabled by default. The resource monitoring of certain types of objects can be individually enabled in the **etc/inst.properties** file. You will need to restart the probe after making changes to this file.

For example, to enable the database cursor monitoring set mercury.enable.resourcemonitor.jdbcResultSet=true for details.conditional.properties property in the **inst.properties** file. This enables object lifecycle monitoring for all resources of this type for a single probe.

Due to higher overhead of caller side instrumentation and possibly large number of objects (resources) to be tracked, it is recommended that this feature is only enabled during development stage. It should be enabled in production environment with great caution and with a very limited 'scope'.

You specify the scope in the object lifecycle monitoring section in the **auto_detect.points** file.

Viewing Object Lifecycle Information

Object lifecycle information is available in the details pane in the Profiler Allocation/Lifecycle Analysis tab for objects enabled for monitoring.

Tips for performance analysis:

The metric **Objects 'Opened' (Total)** shows the number of objects opened during the application's lifetime.

Please note that, if an object is re-opened at multiple location (a common case for pooling), the 'opened' metrics shows the number of times the object was opened. However, the location information refers to the location of the first 'opening' of the object.

Also an object is re-opened without being 'closed' then it is assumed that the object kept itself in the 'open' state. The 'opened' counter is not incremented.

The metric **Objects 'Closed' (Total)** shows the number of objects closed during the application's lifetime. If an object is re-closed without being 'opened' again, then it is assumed that the object kept itself in the 'close' state. The 'closed' counter is not incremented.

The metric **Objects Deallocated without Close** will have a value greater than zero if the resources are not properly released.

The metric **Object Active Lifespan** will have a higher average latency if the resources are not released in timely manner. Note that this metric shows the active lifespan for only those objects that have been closed.

Differences Between Object Lifecycle and Allocation Analysis

Unlike allocation analysis, the object lifecycle feature is not managed. This means that while allocation analysis can be performed by specifically selecting the **Start tracking allocations** and **Stop tracking allocations** links in the Allocation/Lifecycle Analysis tab. Object lifecycle monitoring, if enabled, will show data since the application start-up and the data will not be cleared by the **Clear allocation information** link.

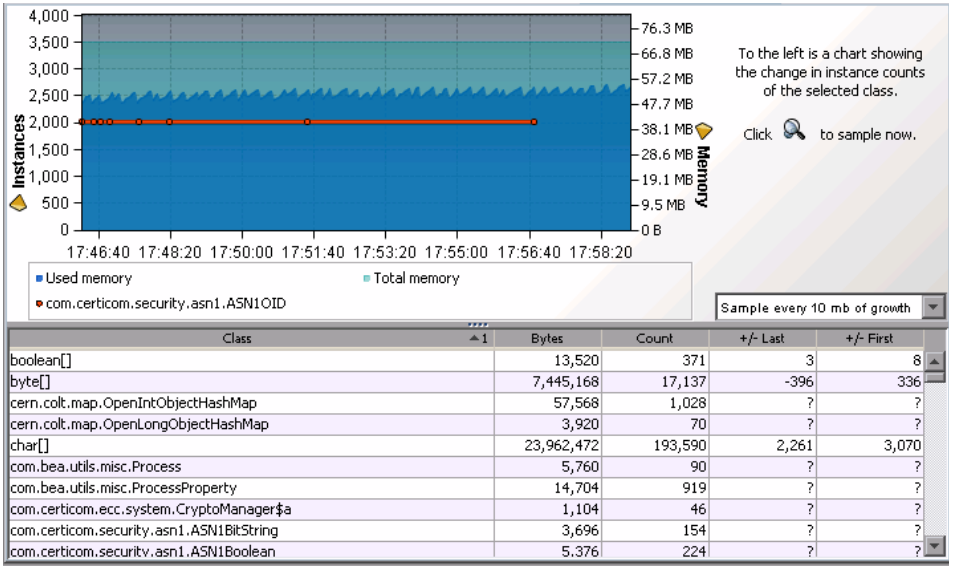
Also, unlike allocation analysis, the object lifecycle feature does not support sampling. This means that all the method calls are captured for the object's lifecycle monitoring.

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.



Class	1	Bytes	Count	+/- Last	+/- First
boolean[]		13,520	371	3	8
byte[]		7,445,168	17,137	-396	336
cern.colt.map.OpenIntObjectHashMap		57,568	1,028	?	?
cern.colt.map.OpenLongObjectHashMap		3,920	70	?	?
char[]		23,962,472	193,590	2,261	3,070
com.bea.utils.misc.Process		5,760	90	?	?
com.bea.utils.misc.ProcessProperty		14,704	919	?	?
com.certicom.ecc.system.CryptoManager$a		1,104	46	?	?
com.certicom.security.asn1.ASN1BitString		3,696	154	?	?
com.certicom.security.asn1.ASN1Boolean		5,376	224	?	?

Enabling the Heap Breakdown Tab

By default, Heap Breakdown is disabled, so the Java Probe does 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 view:

1 Add the files in the **directory <probe_install_dir>\lib\<platform>** to the OS environment path.

Note: Use the environment variable that is appropriate for your operating system. In Windows, the environment variable is **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 take a sample snapshot manually by clicking the sample button.

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.

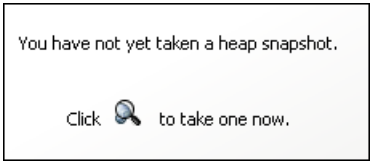
The Memory Analysis tab is made up of the Heap Breakdown view and the Reference Graph.

Enabling the Memory Analysis Tab

By default, the Memory Analysis tab is disabled, so the Java Probe does not impose additional overhead on its host when you do not need the memory diagnostics metrics.

To enable advance memory analysis and display the Heap Walker views:

1. You must have Java 1.5 or higher installed, and use the new **-agentpath:<probe_install_dir>/lib/<platform_dir>/jvmti.dll** parameter instead of **-Xrunheapdump** in the application startup script. Replace **jvmti.dll** with the appropriate library name if you run the probe on a non-Windows system.
2. Open the Profiler for the application, and click the **Memory Analysis** tab.
3. Click the icon to take a heap snapshot and open the first Heap Walker view.



Note: You cannot use Heap Walker when running your application with HotSpot 5.0 JVM with CMS enabled (the -XX:+UseConcMarkSweepGC option). Remove this option from the Java command if you plan to use Heap Walker.

When both the **-server** and the **-Xgc:parallel** options are selected, some versions of JRockit 5.0 JVM demonstrate instability. In some configurations, both options are selected by default. In such cases, specify the **-client** or the **-Xgc:gencon** option to override the default. This is a known BEA issue (CR334327) and should be resolved in future of releases of JRockit.

Understanding the Memory Analysis Tab

Data displayed in the Heap Breakdown view in the Memory Analysis Tab helps users of applications with Java 1.5 or higher to find memory leaks. The Memory Analysis Tab includes a wizard that guides you through the process of diagnosing a memory leak. See "Heap Walker Execution Steps" on page 588 below for details about the process you can use to diagnose memory leaks.

Heap Walker

Heap Walker is a memory analysis process accessible from the Memory Analysis tab. You can use it to troubleshoot Java lingering object problems that are difficult to debug or reproduce. Using object tagging and heap snapshots, Heap Walker enables you to inspect individual objects suspected of having "leaked," and to determine why they are kept alive in the Java heap. This feature targets testing (pre-deployment) environments. You can also use it in production environments.

System Requirements: Heap Walker uses the JVM Tool Interface (JVM TI), which was introduced in Java 5.0. As a result, the profiled application must run on a Java 5 VM that implements JVM TI, including the optional JVM TI capability can_tag_objects.

Sun HotSpot JVM, version 5.0, for Linux and Windows on Intel x86, and HP-UX HotSpot JVM, version 5.0, for PA-RISC, are examples of compatible JVMs. Tagging the heap, and processing the object reference graph, requires large amounts of memory. The amount of memory required depends on the size of the heap used by the application.

Heap Walker Execution Steps

The steps for using the Heap Walker are described below. The Heap Walker also contains a wizard that guides you through the process of diagnosing a memory leak.

Step 1: Establishing a Baseline

A typical large Java application allocates many objects during its initialization and warmup. Classes are loaded, thread and database connection pools are populated, and numerous caches in all components are filled. These objects typically stay alive throughout the application execution. To avoid identifying these objects as potential leaks (that is, to avoid false positives), you should let the application run under load for some time to arrive at a stable state.

The application can be placed under memory leak test after initialization has completed, and object allocation has stabilized. Clicking **Start Tracking New Objects** initiates the test operation. After that, any objects allocated by the operation will be tracked as potential leaks.

The assumption here is that the deployed Java application, if allowed to fully initialize, allocate only temporary objects for all of its operations. All temporary objects should eventually be garbage collected. While most server applications comply with this design principle, there are known exceptions to this rule. Database connections, or threads in dynamically sized thread pools, can be created at any time during the application execution without time constraints on when they should be terminated.

The Heap Walker operations may also leave a footprint on the heap. (For example, some probe classes are loaded and initialized only when you start using Heap Walker.) Footprints should not be a problem if you are aware of them. However, if you need a clear picture, it is recommended that you perform the execution twice, treating the first pass as a warm-up only. You can ignore results from the first pass.

Step 2: Exercising the Operation

The details of this step may differ, depending on whether the application is running in a production environment or in a test environment. In a test environment, the application owner can carefully stage the test load to contain only the desired operations. For example, testing can focus on newly developed code, or objects that are suspected of leaking memory based on the analysis of the logs or feedback from the IT center where the application is deployed.

It is often useful to use such an operation under test in some kind of a context. For example, if the application requires a user logon, it might be practical to wrap the tested operation by a logon and logout. It is typical for the application to hold the active session information in the heap. In this case, you can dismiss the session information only after a logout. Alternatively, you can perform the logon before new object tracking is started. In any case, you should arrange the tested operation in such a way that it leaves no permanent footprint in Java memory (adding records to a database is fine). The tested operation can be repeated several times. In the case of simple leaks, a single execution is usually enough. In a production environment, it is impossible to control the load, or to time the new object tracking by starting and stopping to catch only the desired portion of the load. You need to take this into account when analyzing the results.

Heap Walker can display the number and size of the currently tracked objects. These numbers are updated by taking a heap snapshot. You observe the numbers as they change over time. Measurements increase as the application allocates new objects. They decrease as the objects are garbage collected. After the tested operation is complete (or, in the case of a production environment, sufficient time has elapsed), you can click **Stop Tracking New Objects**. At this point, the set of tracked objects is closed. It can no longer grow.

It is normal, however, for several tracked objects to still be alive at this point. They can be present in numerous caches in the application, including the components that you do not own. It is also possible that some tracked objects require finalization. The finalizers are run periodically by the JVM, typically asynchronously to the activities controlled by the application. Objects pending finalization are considered alive, even though the application may hold no references to them.

Step 3: Flushing Application Caches

Under normal circumstances, if the application remains under load, the caches clear of all the tracked objects eventually, and the pending finalizers run eventually. The JVM also runs garbage collection periodically. This garbage collection removes the tracked object from the heap, provided they are not leaks.

You can sometimes speed up this cleaning process by forcing garbage collection. Clicking **Run Garbage Collection** makes the JVM not only run the full GC cycle, but also run the pending finalizations.

Taking heap snapshots is especially useful at this point. The observed number and the total size of tracked objects should go down over time, as the cache flushing process progresses. Ideally, these numbers should eventually reach zero, meaning that all tracked objects have been garbage collected.

However, if there is a Java memory leak in the tested operation, the numbers stabilize at some non-zero values, and no longer decrease, despite repeated garbage collections and continuous load on the application. When you decide that the tracked objects remaining on the heap should be considered a leak, it is time to capture the **object reference graph**. This action dumps all references present in the heap to a file, and starts an additional (external) process, which sorts the file. The file is used in the next steps.

Step 4: Analyzing Potential Leaks

After you capture the object reference graph, you can retrieve the list of tracked objects. In most cases, you select just one class of objects to retrieve. This can be accomplished by double-clicking the row with the selected class. It is also possible to retrieve objects for multiple classes. Simply select multiple rows (by holding down the Ctrl key), and then right-click to select **Inspect Selected Tracked Objects**.

For efficiency of operation, there is a limit on the total number of objects that can be retrieved. You can change the limit, using the selector located on the left side of the window. Retrieving a large number of objects rarely makes sense, as it is costly, and it does not necessarily increase your capability to solve the leak problem.

Step 5: Walking the Heap

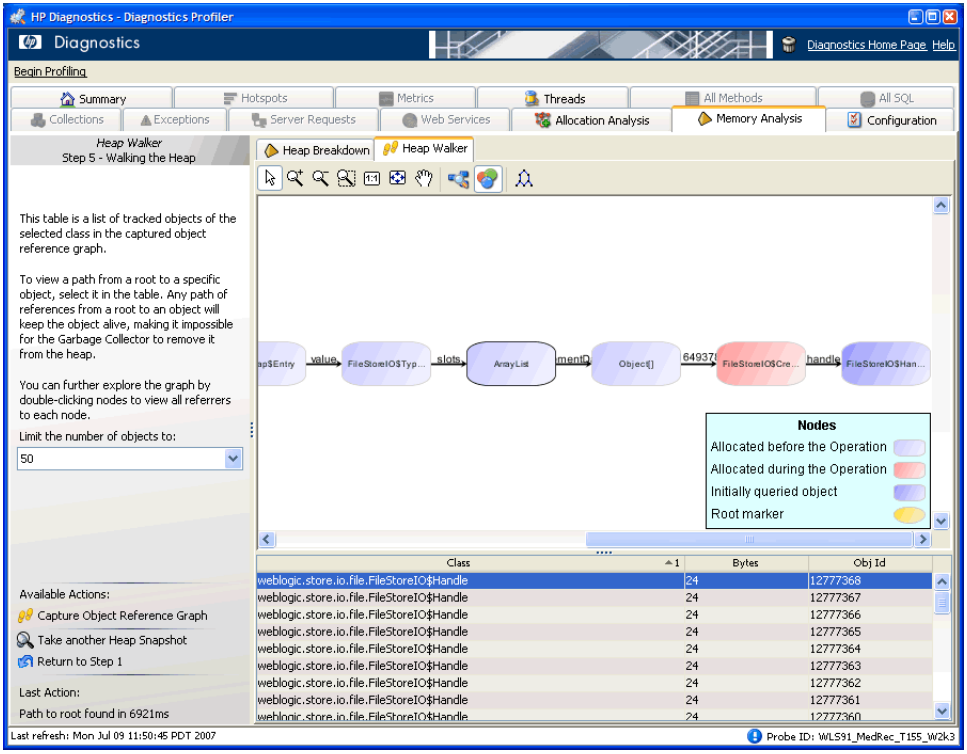
You can determine why any of the retrieved objects is alive by clicking the table row describing the object. This action displays the selected object with a chain of references that are keeping the object alive, and indicates which object is a heap root.

For any object already displayed, it is possible to show all objects directly referencing it by double-clicking the object.

As above, to keep a limit on the overhead, there is a limit selector on the left side of the screen controlling the maximum number of objects to be retrieved and displayed.

All displayed references (links between objects) are based on the captured object reference graph. Additional information, such as object type, size, or reference names are retrieved directly from the heap. Under some circumstances, the additional information cannot be retrieved because some of the objects keeping the specific object alive can cycle over time and be garbage collected. A continuously growing java.util.Vector object is a good illustration of this point, as the underlying array is replaced over time.

You may elect to capture a new reference graph to obtain a fresh view of the graph, and repeat Step 4: Analyzing Potential Leaks.



The objects are color-coded according to their age. There are three distinct object ages:

- **Baseline.** Objects allocated before new object tracking was started.
- **Tracked.** Objects allocated between new object tracking start and new object tracking stop, ostensibly by the tested operation.
- **Fresh.** Objects allocated after new object tracking was stopped.

Heap Walker Performance Characteristics

Technically, starting or stopping new object tracking, and capturing the object reference graph, uses the JVM heap tagging operations. It may require substantial execution time, which can be up to several minutes for very large heaps. The application is practically paused during this time. Do not use Heap Walker if the nature of the deployed application cannot tolerate such long pauses. If in doubt, always test Heap Walker first in a test environment.

The above steps, and in particular starting new object tracking, also make the JVM allocate extra memory generally proportional to the current heap size. This memory is allocated outside of the Java heap, but within the JVM process. You need to take special care to ensure that such memory can be allocated. Keep in mind that the JVM itself, the application code, the JIT-compiled code, and any native libraries used by the application must fit into this space as well. For 32-bit processes, there is an operating system-dependent limit on the size of the process address space (for example, 2GB for Windows on Intel x86). If almost half (or more) of the available address space is already reserved by the Java heap, the tagging operation can crash the JVM.

Heap Walker gives you the total memory usage estimate when the **Start New Object Tracking** operation is activated for the first time. The estimate is for total system memory. It is based on additional memory needed by the JVM and on memory for the object references sorting program. At this point, you have a chance to quit Heap Walker without affecting the deployed application negatively (no additional memory is allocated). Obviously, in a production environment (deployed application), it is recommended that you use Heap Walker only if the system capacity is large enough to handle the additional memory pressure. The decision whether to continue with tagging depends not only on the total amount of memory available on the system running the application, but on the impact of a possible JVM crash on the business process as well.

When using Heap Walker in a test environment, it is usually possible to scale down the load and the maximum heap size to match the system capacity. There is no direct CPU overhead on the Java application, other than actually running a Heap Walker command (indicated by the progress bar). This also includes the tracking period. That is, even though tracking start and tracking stop consume large amounts of CPU time, there is no overhead while actually tracking new objects.

However, the increased memory footprint of the JVM may cause serious sluggishness if the JVM no longer fits into main memory, and makes excessive use of the swap area. If, after having tagged the heap, you notice severe application performance degradation while none of the Heap Walker operations are running, you most likely have a swap file thrashing problem.

Part VI

Using the Diagnostics Profiler for .NET

41

Using the .NET Diagnostics Profiler

The Diagnostics Profiler for .NET is installed with the .NET probe. The Profiler runs in a separate UI and provides near real-time data, enabling you to pinpoint application performance bottlenecks.

This chapter includes:

Note: The 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.

If you installed the probe from the HP Software Web site and you want to use it with a Diagnostics Server, contact HP Software Support.

Accessing the .NET Diagnostics Profiler

The .NET Diagnostics Profiler is installed as part of the 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 HP Diagnostics screens.

Remote access to the .NET Profiler can be disabled with the profiler element in the probe_config.xml file.

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, usernames and passwords when you have the full Diagnostics product, refer to the *HP Diagnostics Installation and Configuration Guide* section on Authentication and Authorization.

To access the Diagnostics Profiler from HP Diagnostics:

➤ From any Status screen in HP Diagnostics, right-click the probe entity and select **View Profiler for <probe name>** from the menu.

➤ Alternatively, in the probe standard view screen in HP 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, ensure that you have set a default browser within your operating system.

.NET Diagnostics Profiler Inactivity Timeout

By default, the .NET Diagnostics Profiler is not started until you display the profiler UI. When you close the profiler UI, the profiler continues to run for a period of time specified by the **inactivitytimeout** attribute in **<probe_install_dir>/etc/probe_config.xml**. If you reopen the profiler UI before the profiler times out, the profiler displays the data for the time period since the profiler was started. If you reopen the profiler UI after the timeout has occurred, the profiler is restarted and only the data for the new profiler session is displayed. As long as the Profiler UI is open, the profiler session remains active. The count down for the inactivity timeout begins when you close the profiler UI.

Enabling and Disabling the .NET Diagnostics Profiler

When the .NET Probe is installed to work with a Diagnostics Server, the probe starts automatically when a Web page in the monitored application is accessed. By default the probe does not start the .NET Diagnostics Profiler until you access the Profiler UI. You may configure the probe so that the Profiler is started at the same time that the probe is started or so that the Profiler cannot be started.

Configuring the Probe to Start the Profiler Automatically

You may want to start the .NET Diagnostics Profiler at the same time that the .NET Probe is started if you are trying to understand the performance of your application when it is first invoked.

To configure the probe to automatically start the profiler:

Set the **pro** attribute in **<probe_install_dir>/etc/probe_config.xml** to true.

```
<modes enterprise="true" pro="true"/>
```

Configuring the Probe to Prevent the Profiler From Starting

You may want to prevent someone from starting the Profiler on a probe that is monitoring an application where you do not want to incur the additional overhead from the Profiler.

To configure the probe to prevent the Profiler from starting:

Set the **pro** attribute in **<probe_install_dir>/etc/probe_config.xml** to false

```
<modes enterprise="true" pro="false"/>
```

Configuring the Probe to Start the Profiler When You Access the UI

By default, the probe starts the Profiler when you bring up the Profiler UI. If you have altered the setting for the probe, you may want to reset the behavior of the probe to the default behavior.

To configure the probe to start the Profiler when you access the UI:

Set the **pro** attribute in **<probe_install_dir>/etc/probe_config.xml** to auto:

```
<modes enterprise="true" pro="auto"/>
```

Note: If you do not include the **pro** attribute, the probe defaults to the behavior for when **pro** is set to auto.

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 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. The Profiler ignores all configured trim settings, for example, latency trimming, depth trimming or throttling. For details about trim configuration refer to the section about advanced .NET Probe Configuration in the *HP Diagnostics Installation and Configuration Guide*. 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 Profiler was started/reset to the time that the user interface was started/refreshed. In the meantime the .NET Probe continues to monitor your application, capture method metrics, and capture call stacks. These changes are not sent automatically to the user interface, you must request them via the **Refresh Now** button. This is so the underlying data will not change unexpectedly while you are investigating something of interest.

For more information on monitoring method latency with the .NET Diagnostics Profiler, see Chapter 42, “Analyzing Method Latency with .NET Diagnostics Profiler.”

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

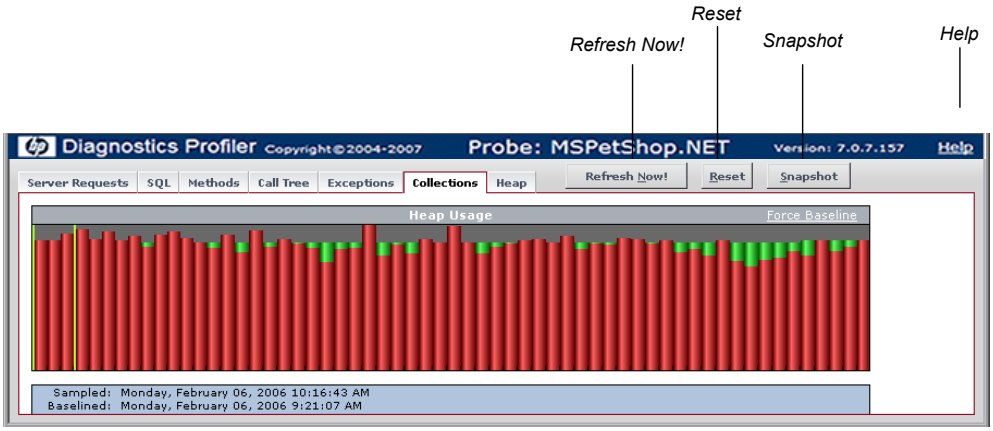
Light Weight Memory Diagnostics allows you to monitor the collections that your application has created, and to identify the largest collections and the fastest growing collections. With Heap Breakdown you can monitor the heap generation breakdown and the objects that are stored in heap. This helps you to identify objects that may be leaking.

For more information about Light Weight Memory Diagnostics, see "Analyzing Memory Using the Collections Tab" on page 626.

For more information about Heap Breakdown, see "Analyzing Memory Using the Heap Tab" on page 632.

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 Diagnostics Profiler for .NET.

42

Analyzing Method Latency with .NET Diagnostics Profiler

You can use the different tabs in the .NET Profiler to analyze method latency for the selected application.

This chapter includes:

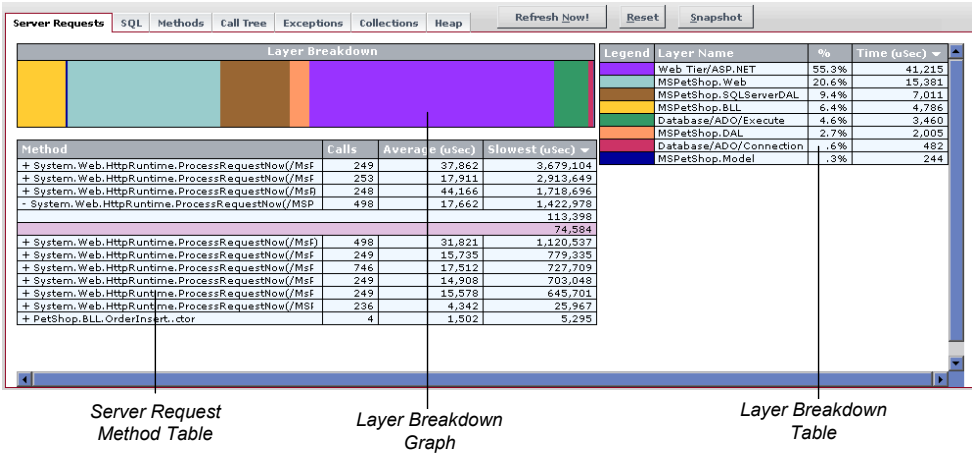
Analyzing Performance Using the Server Requests Tab

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

An example of the Server Requests tab display is shown below:



The data in the Server Requests tab is as follows:

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.

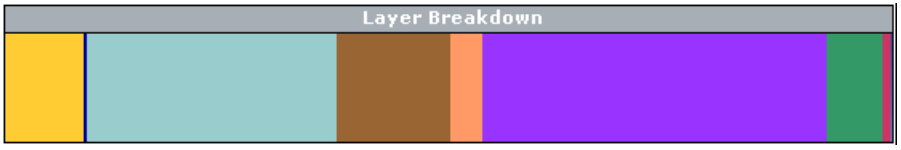
Legend	Layer Name	%	Time (µs)
	Web Tier/ASP.NET	55.3%	41,215
	MSPetShop.Web	20.6%	15,381
	MSPetShop.SQLServerDAL	9.4%	7,011
	MSPetShop.BLL	6.4%	4,786
	Database/ADO/Execute	4.6%	3,460
	MSPetShop.DAL	2.7%	2,005
	Database/ADO/Connection	.6%	482
	MSPetShop.Model	.3%	244

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 the 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 618.

Analyzing Performance Using the SQL Tab

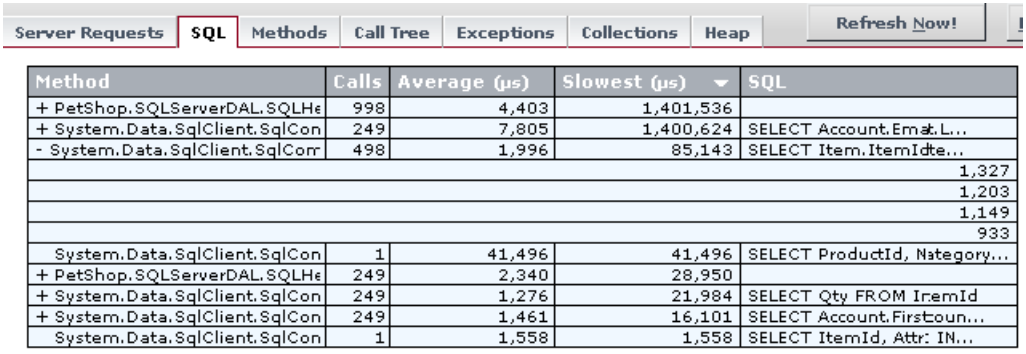
The .NET Diagnostics Profiler keeps track of all of the method calls that your application makes. The SQL tab displays the SQL methods only. The SQL methods are listed in the Method table which shows the number of times that each method was executed, along with the average latency and the slowest execution 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.

Understanding the SQL Method Table

The SQL tab contains the SQL Method table.



Method	Calls	Average (µs)	Slowest (µs)	SQL
+ PetShop.SQLServerDAL.SQLHe	998	4,403	1,401,536	
+ System.Data.SqlClient.SqlCon	249	7,805	1,400,624	SELECT Account.Emai.L...
- System.Data.SqlClient.SqlCon	498	1,996	85,143	SELECT Item.ItemId.te...
				1,327
				1,203
				1,149
				933
System.Data.SqlClient.SqlCon	1	41,496	41,496	SELECT ProductId, Nategory...
+ PetShop.SQLServerDAL.SQLHe	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 618.

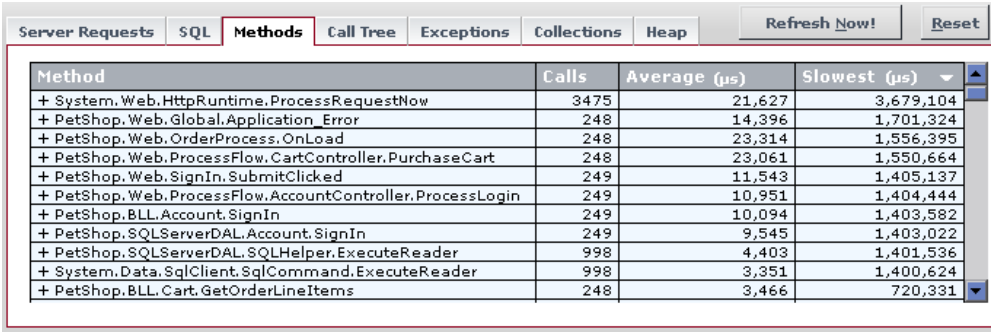
Analyzing Performance Using the Methods Tab

The .NET Diagnostics Profiler keeps track of all of the method calls that your application makes. The Method**s** 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.

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 618.

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.

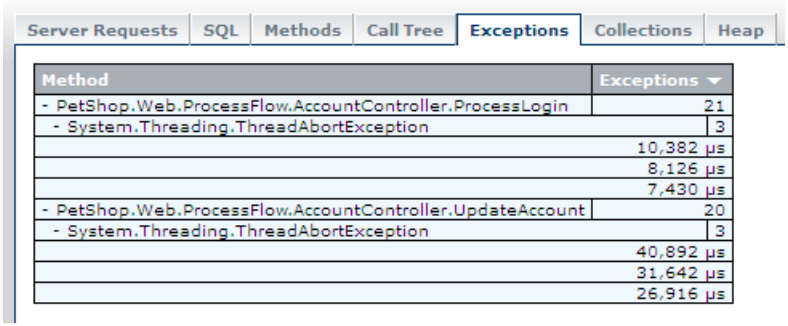
Note: Exceptions are only captured by the probe if the exception causes the termination of a method. If the instrumented method handles the exception, no exception information is gathered by the probe.

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.

Understanding the Exception Table

The Exceptions tab contains the Exception Method table.



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 table includes the following columns:

- **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 618.

Analyzing Performance Using 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.

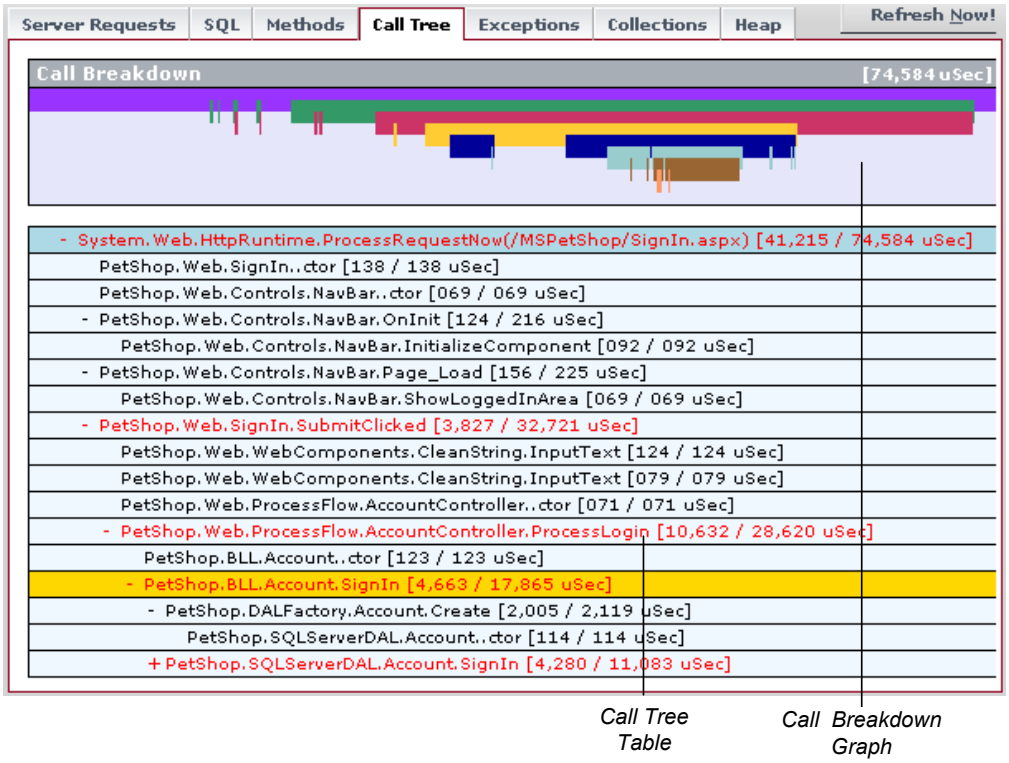
As you analyze the methods presented on the Server Requests, SQL, Exceptions, and Methods tabs, you navigate to the Call Tree tab to understand the context of the processing associated with particular instances of the method's execution. The call tree allows you to see the calling and the callee methods 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

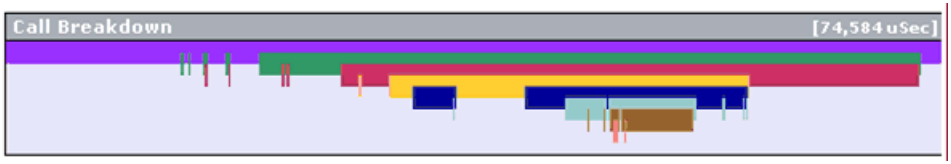
An example of the Call Tree tab display is shown below:



The Call Tree tab is divided into the following sections:

Call Breakdown Graph

The Call Breakdown graph shows the processing time that was spent at each level of the call tree hierarchy.



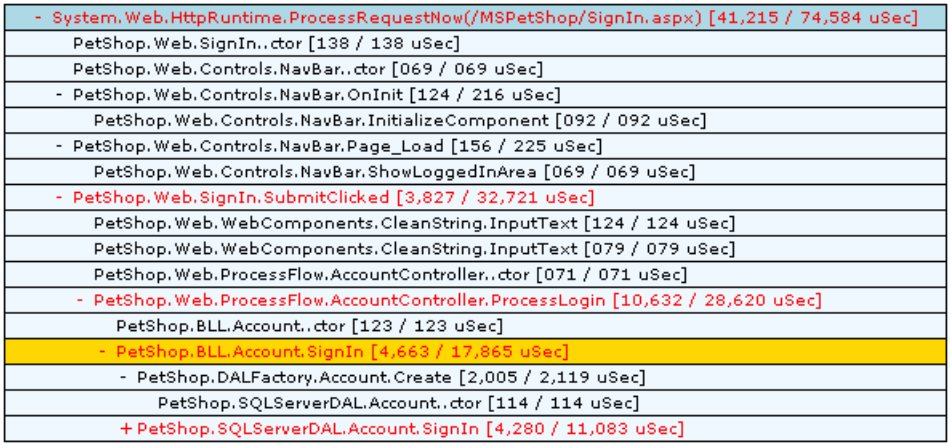
Each level in the graph represents the processing at the corresponding level in the call stack. The length of the bar is proportional to the length of time spent in performing the methods at that level of the call stack. The positions where a bar starts and stops indicates the relative time, in relationship 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.



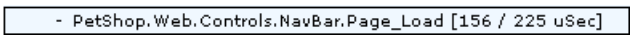
Call Tree Methods

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**.



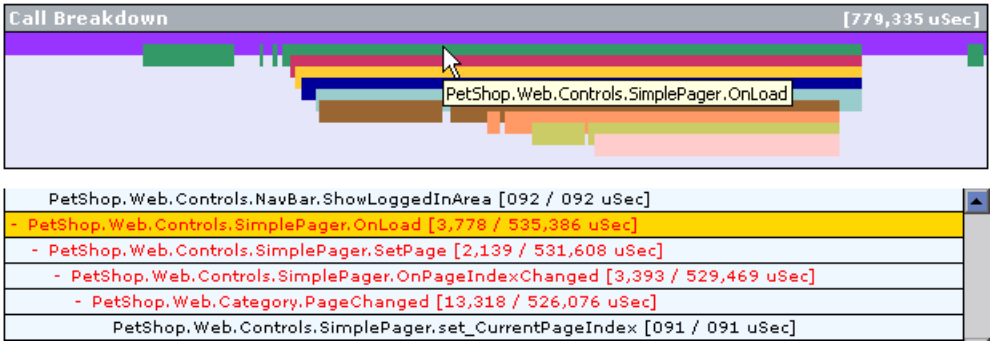
Method of Interest in the Call Tree

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:

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

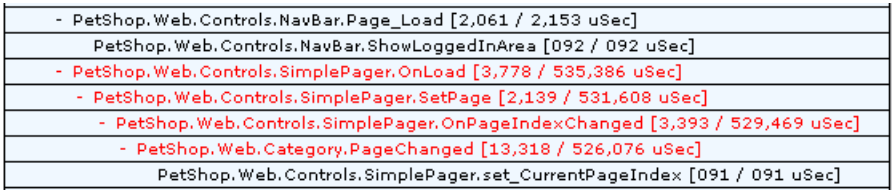
You may identify the method associated with a particular location on the Call Breakdown graph by mousing over the bars in the graph. As you slide the pointer along a bar in the graph, the Call Tree table scrolls so that the method associated with the selected location in the graph is displayed in the 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:



43

Analyzing Memory Using .NET Diagnostics Profiler

You can analyze memory problems for the selected application using the memory diagnostics metrics displayed in the .NET Profiler.

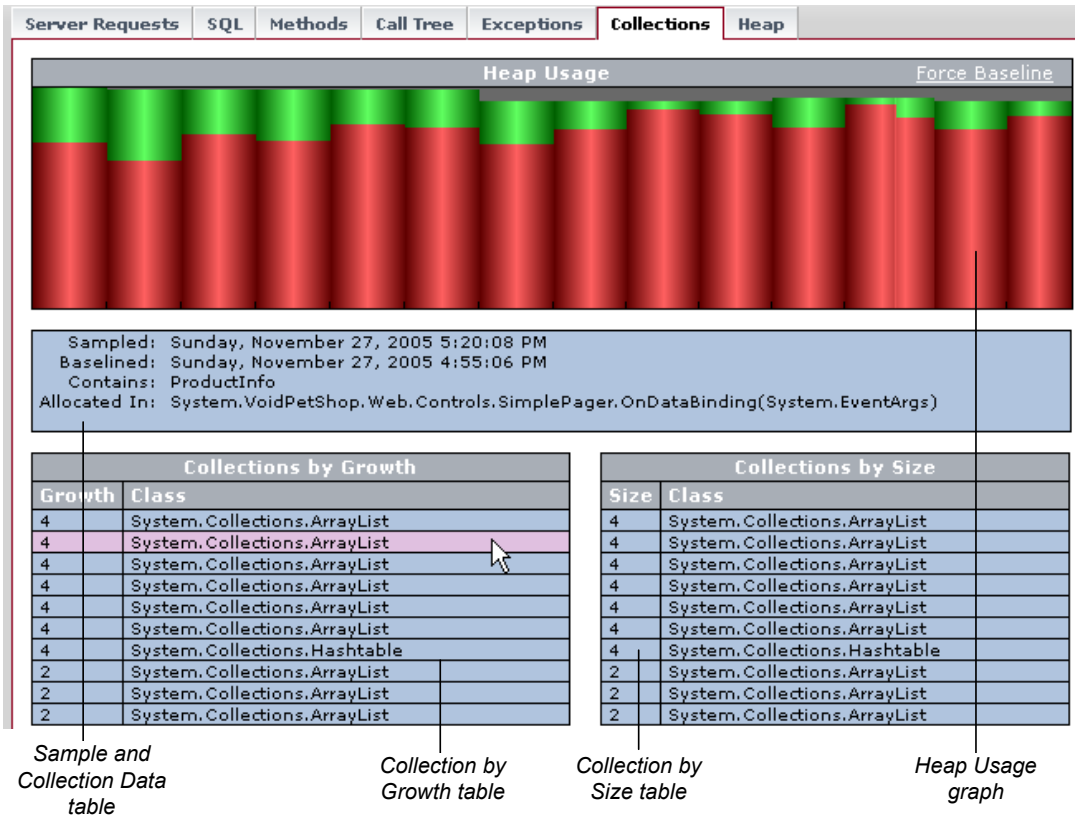
This chapter includes:

Analyzing Memory Using 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 Collections tab displays these problems, enabling identification of memory issues.

The Collections Tab at a Glance

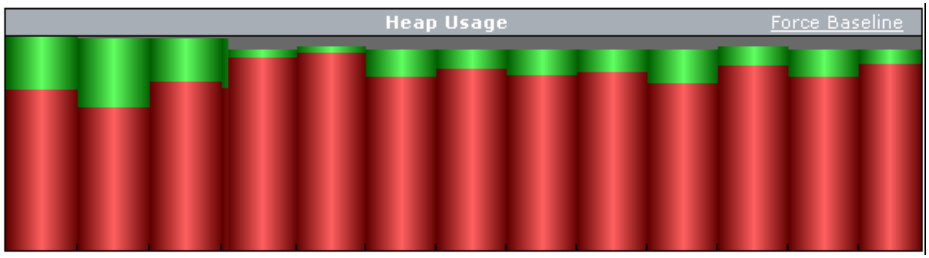
The following image is an example of the Collections tab:



The Collections display is divided into the following sections:

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:

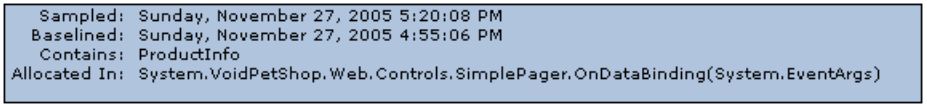
In the Heap Usage graph, hold the mouse pointer over that sample's bar.

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.



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.

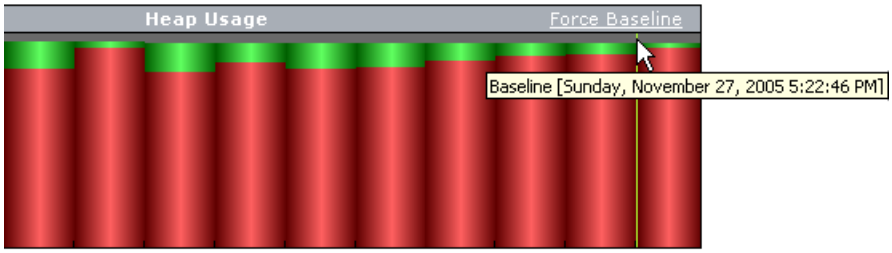
Viewing Details for a Selected Collection

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.

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>
```

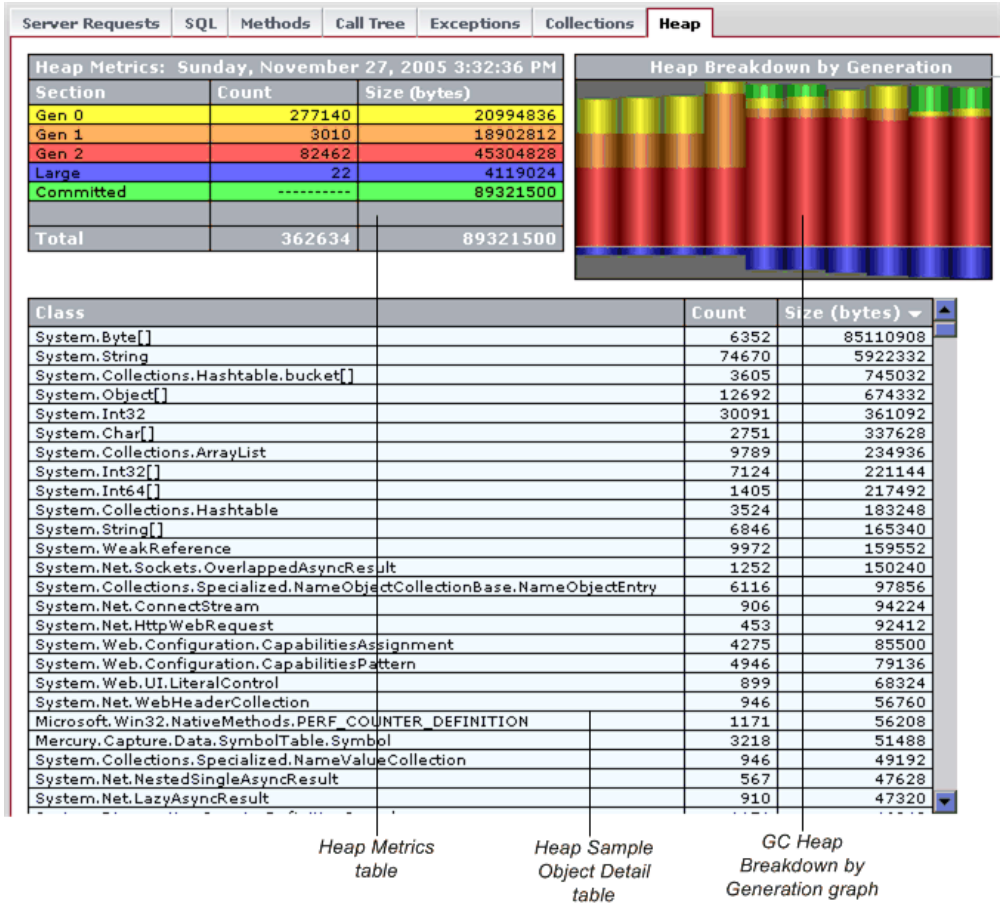
The monitorheap feature enables detailed memory tracing which is intended solely for deep drilling into memory allocation when there is reason to believe that there is a memory problem associated with an application.

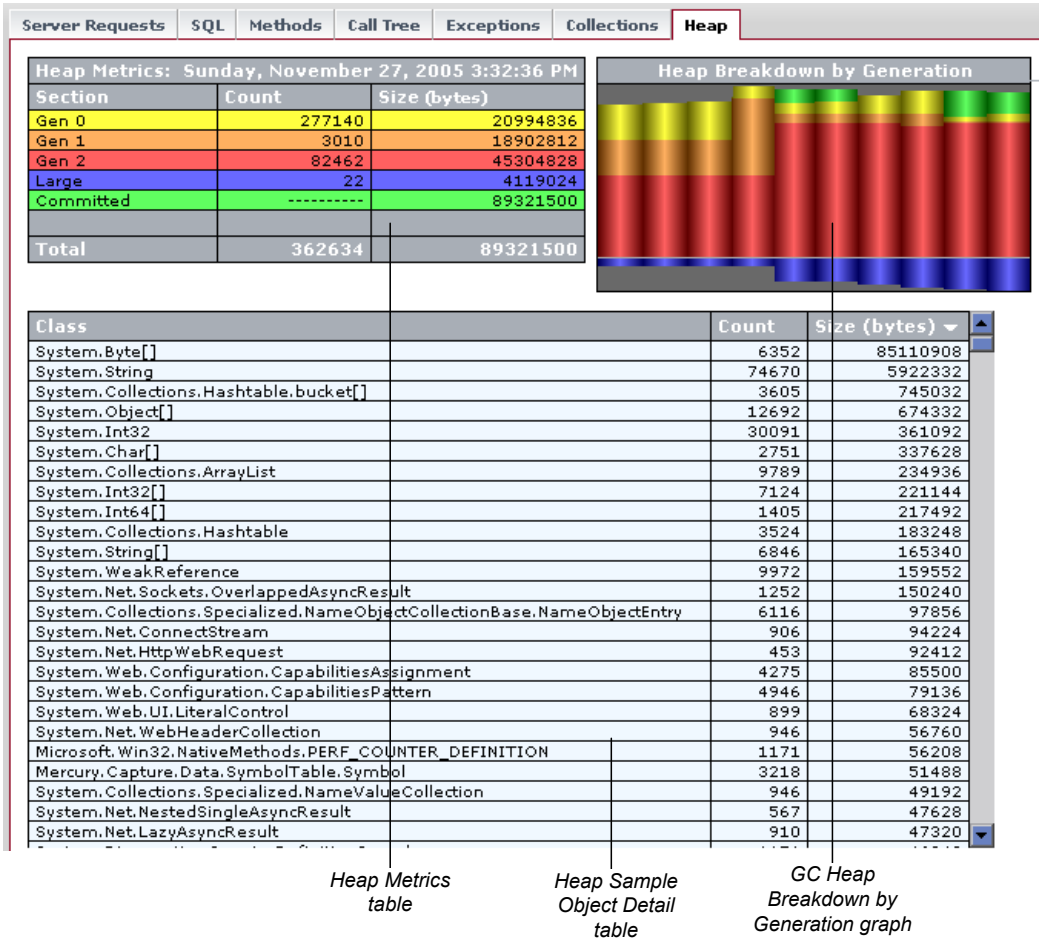
This additional deep memory tracing causes the .NET Probe to watch/track every single allocation and deallocation made for the entire hosting process and adds significant overhead compared to the normal method latency tracking. For this reason, this feature should only be used if there is a strong suspicion that there is a memory leak or some other memory allocation or cleanup issue that is adversely affecting the monitored application.

Important: This feature should always be disabled after it is used to debug the adversely affected application; it should never be a standard enabled production configuration.

The Heap Tab at a Glance

The following image is an example of the Heap tab display:

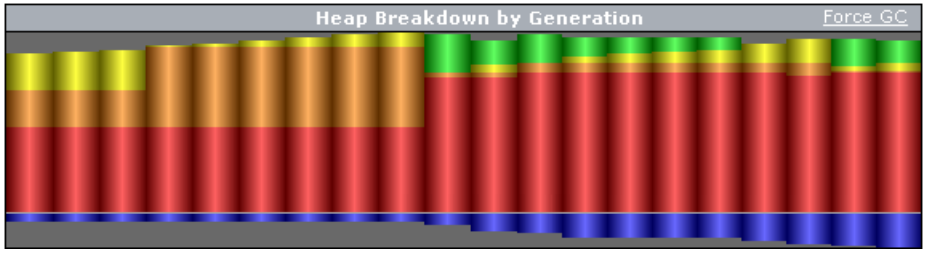




The Heap tab is divided into the following sections:

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

About Integration, Configuration and Diagnostics Terminology

44

Diagnostics Data in Other HP Software Products

Diagnostics can be integrated with other HP Software products to provide information to help you diagnose performance problems.

This chapter includes:

Viewing Diagnostics Data in Business Availability Center

Diagnostics 8.00 can be integrated with Business Availability Center 7.0 or later to provide information to help you understand and improve the performance of your applications.

Important: For SOA customers, Business Availability Center 7.50 or later is required in order to take advantage of Diagnostics Web services enhancements in the Business Availability Center integration.

Anytime a Business Availability Center system is upgraded or re-installed, a manual hard sync is needed (or a wait period of 24 hours) before Web services currently shown in Diagnostics are forwarded to Business Availability Center. To do a hard sync, go to the Diagnostics Server Administration page, select **Synchronize** and then select **Hard** for A**ll Customers**.

From within Business Availability Center, you can actively track the performance status of your applications that are being monitored by HP Diagnostics. The Diagnostics integration with Business Availability Center allows you to:

- access the HP Diagnostics screens from within Business Availability Center.
- drill down to Diagnostics data from specific Business Availability Center configuration items and reports.
- generate high level reports in Business Availability Center about the performance of applications and Business Process Monitor (BPM) transactions that are monitored by Diagnostics.

Accessing the Diagnostics Views

From Business Availability Center you can access the HP Diagnostics views 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.

Note: 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, 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 Business Availability Center's Dashboard, you can actively track the performance status of your applications that are being monitored by HP Diagnostics.

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.

  The status reflected by the Application KPI is defined by specific thresholds, which you set in the HP 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.

  In the Diagnostics View in Dashboard, you can view the status of your applications that are being monitored by Diagnostics Probes. You can view the status of an individual Diagnostics probe or of a group of Diagnostics

  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.

  Before viewing Diagnostics data for transactions included in a transaction monitor, you need to enable Diagnostics breakdown.

  After you have enabled Diagnostics breakdown, you can monitor the performance status of transactions From BPM related views in Dashboard.

Note: You can drill down to the Diagnostics screens that display Diagnostics data. For more information, see “Drilling Down to Diagnostics from Business Availability Center” on page 645.

Drilling Down to Diagnostics from Business Availability Center

In the Business Availability Center Dashboard, you can drill down from selected CIs (Configuration Items) to the Diagnostics views displaying data about that item. Also you can drill down to Diagnostics views from certain Business Availability Center reports and Business Availability Center for SOA reports.

- **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 349.

- **RUM-related views.** From selected CIs in RUM-related views, you can drill down to the Diagnostics Server Requests, which displays the performance metrics for the monitored server requests in your application. The drill down to Diagnostics displays a snapshot with details for the corresponding server requests. In the snapshot, you can select any of the instance tree icons to drill down further to a call profile.

Note: If no server requests are found when drilling down to Diagnostics, there will be no data displayed in the snapshot and an error message in a yellow box will be displayed indicating the server request was not found.

For more information about the Diagnostics Server Requests, see "Aggregate Requests and Server Requests Views" on page 311 and for more information about Diagnostics snapshots see Chapter 8, "Performing Snapshot Analysis".

- **TransactionVision views.** From TransactionVision views you can drill down to Diagnostics to get the following types of information:
  - The drill down to a Diagnostics Hosts view provides system metrics like CPU utilization and memory utilization that can help you determine if the problem is a system issue.

- The drill down to a Diagnostics snapshot of server request details can help you study the performance of these server requests over time and look at detailed instance trees (call profiles, methods, exceptions and faults) associated with the server requests to identify where the issue lies.
- From the Server Request data you can navigate to the corresponding probe entity and look at the application server metrics (for example, JMX metrics) to see if the issue lies with the application server.

- **SOA-related views.** From selected CIs in the SOA-related views, you can drill down to the Diagnostics Service Topology, which displays a topology view of monitored Web services and service connections or drill down to Diagnostics Operations view.

  The Diagnostics Service Topology, shows each instance of the Service on a probe. If the Service you drill down from is hosted on multiple probes, Service Topology will display the topology for the first service that matches the drill down parameters. You can distinguish between service instances using the Probe column in the Services Topology entity table.

  For more information about the Diagnostics SOA Services view group, see "SOA Services Views" on page 419.

- **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 281.

- **SiteScope.** If you set up the integration between SiteScope and Diagnostics (configured in SiteScope), then SiteScope sends data to Diagnostics. This SiteScope data is displayed in the External Monitors view group. See Chapter 37, "External Monitors Views".

Diagnostics Configuration that can Affect Integration with Business Availability Center

Diagnostics provides information to Business Availability Center and this information is included in several Business Availability Center reports. Some of the information from Diagnostics is configurable as described below.

- In Diagnostics, you can configure consumers in a flexible way according to:
  - IP address or IP range.
  - HTTP Header.
  - SOAP request Headers, Body or Payload. SOAP payload is the entire SOAP envelope.
  - JMS Queue name (or topic name) and JMS message header or JMS message property for SOAP over JMS web services.

IP address is used as the default consumer ID for SOAP over HTTP/S web services and inbound queue name (or topic name) is used as the default consumer ID for SOAP over JMS web services.

In Business Availability Center for SOA reports such as the **Consumer Summary Report**, consumer information can be any of these as configured in Diagnostics. You can also configure the number of consumers to be monitored by probe. See "About Consumer IDs" on page 433 and the configuring consumer IDs section of the *HP Diagnostics Installation and Configuration Guide* for more information.

Important: Business Availability Center 7.50 or later is required if you want to view consumers that have been configured as something other than the default (IP address for SOAP over HTTP/S and queue or topic name for SOAP over JMS).

- In Diagnostics, you can also configure the application server instance field in the Probe view's details pane to be something other than the probe name (application server instance name is used as the Endpoint in Business Availability Center). In Business Availability Center for SOA there is a **Server/Endpoint Summary Report** that displays application server instance as configured in Diagnostics. Note that blank web service namespaces and end_points are filled with "Unknown". See "Specifying a Name for the Application Server Instance" on page 428 for information.
- Slow Calls (over_threshold) for default web service operations can be sent to Business Availability Center if the following option is set to true in the Diagnostics **server.properties** file: threshold.violations.metric.for.web.service.include_default_violations.
- In Diagnostics, you can specify a set of SOAP faults that should not be counted towards unavailibility in Business Availability Center. Out-of-the-box, only server-side faults (Receiver/Server) are counted in determining availability in Business Availability Center. Client-side faults are excluded from availability calculations but are shown as SOAP Faults in the Business Availability Center for SOA Health report.

  By default, the following SOAP faults are excluded in the availability calculation used in Business Availability Center: VersionMismatch, MustUnderstand, DataEncodingUnknown, Sender/Client.

  In Diagnostics, you can configure the faults that should be excluded when calculating availability. You set the **soap.faults.excluded** property in the **server.properties** file and **dispatcher.properties** file for Java Probes. The .NET probe's aggregation is done on the server in the **server.properties** file so no additional changes need to be made for the .NET probes.

  This must be configured the same for the whole deployment (all Diagnostics probes and servers). See the *HP Diagnostics Installation and Configuration Guide* for more information on the property files.

- If the server request is not found when drilling down to Diagnostics it is most likely due to latency trimming on the Diagnostics side where server requests that take less than 51ms are not collected. Latency trimming is configurable on the probe. For the Java Probe this is changed by setting **minimum.fragment.latency** to 0ms in the **dispatcher.properties**. For the .NET Probe this is changed by setting **<trim><latency min="0" /></trim>** in the **probe_config.xml** file. See the *HP Diagnostics Installation and Configuration Guide*.

  There also may be no data when drilling down to Diagnostics if the application server handling a particular page is not monitored by a Diagnostics probe.

- There are a number of options when configuring the Diagnostics integration with RUM. A brief overview of the configuration is provided below, see the *HP Diagnostics Installation and Configuration Guide* for more information.

  - You can disable the RUM integration in case it is not used. For the Java probe, in **capture.properties** set the value of **rum.coloring.enabled** to false. For the .NET probe, in **probe_config.xml** configure the element to **<rum enabled="false" />**.

  - Parameter aggregation is enabled by default in Diagnostics (**auto_detect.points** file **[HttpCorrelation]** element). You can define keys if you want to see data specific for some parameters. If you turn off parameter aggregation then all URLs in RUM that contain a parameter will drill down to the same server request in Diagnostics (for example, if RUM has /url?p=a and /url?p=b the corresponding server request in Diagnostics will be /url).

  - You can configure Diagnostics to monitor post parameters in the **inst.properties** file by setting **ignore.post.parameters=false**.

Viewing Diagnostics Data from TransactionVision

Diagnostics integrates with TransactionVision views in Business Availability Center to provide additional information useful in doing triage performance problems. Business Availability Center (including TransactionVision) 8.00 can be integrated with Diagnostics 8.00.

A typical scenario would be to use TransactionVision to identify a specific instance of a business transaction that is performing poorly, and then drill down to the application server requests in Diagnostics to continue the process of identifying the problem code components.

Important: Before viewing Diagnostics data from TransactionVision, you must have installed the Java Agent and/or .NET Agent and configured the agents as both a Diagnostics probe and a TransactionVision sensor. See the *HP Diagnostics Installation and Configuration Guide* for details.

Note: To avoid a user having to login again when drilling from TransactionVision to Diagnostics, lightweight single sign-on (SSO) must be enabled in the Diagnostics Server. See the *HP Diagnostics Installation and Configuration Guide* for details on configuring single sign-on.

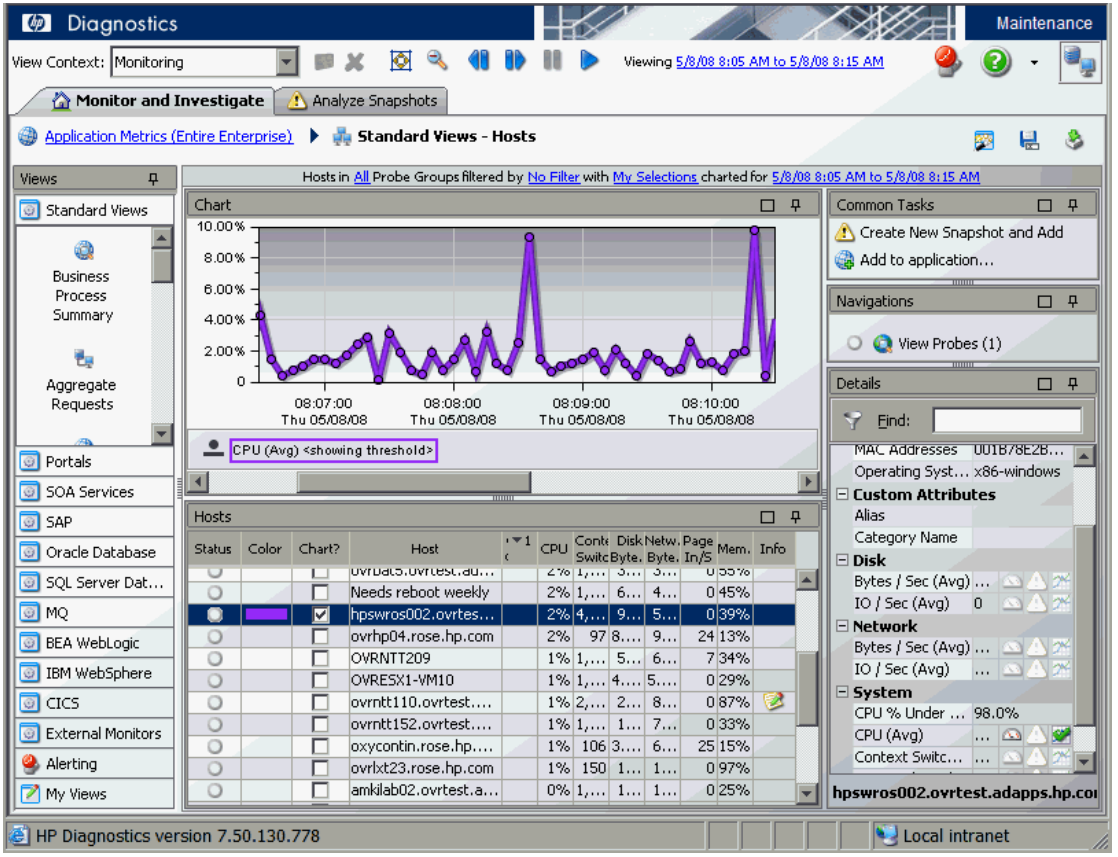
Drilling Down to the Diagnostics Hosts View from TransactionVision Views

You can drill down from a TransactionVision event to the Diagnostics Hosts view.

In TransactionVision views, select any Java event, for example, EJB event, HTTP event, JMS event (Event Analysis details display in TransactionVision).

Select the drill to Diagnostics icon located next to a TransactionVision host.

The Diagnostics UI opens and the Hosts view is displayed. The TransactionVision host is selected, data for the host is shown in the graph and metrics for the host are displayed in the details pane. This is useful because you can see system metrics and identify if the problem is a system issue.



Some of the metrics available in the details pane for the host system include: Average CPU Utilization, Average Memory Utilization, Average Disk IO in Bytes/Second, Average Network IO in Bytes/Second.

From this Hosts view in Diagnostics you can see the performance of the host system charted over various time periods. Also you can check to see if the system has recently experienced a performance degradation or if the system has consistently under performed.

Drilling Down to the Diagnostics Server Requests Data from TransactionVision Views

There are several ways to drill down to Diagnostics server requests data from TransactionVision views.

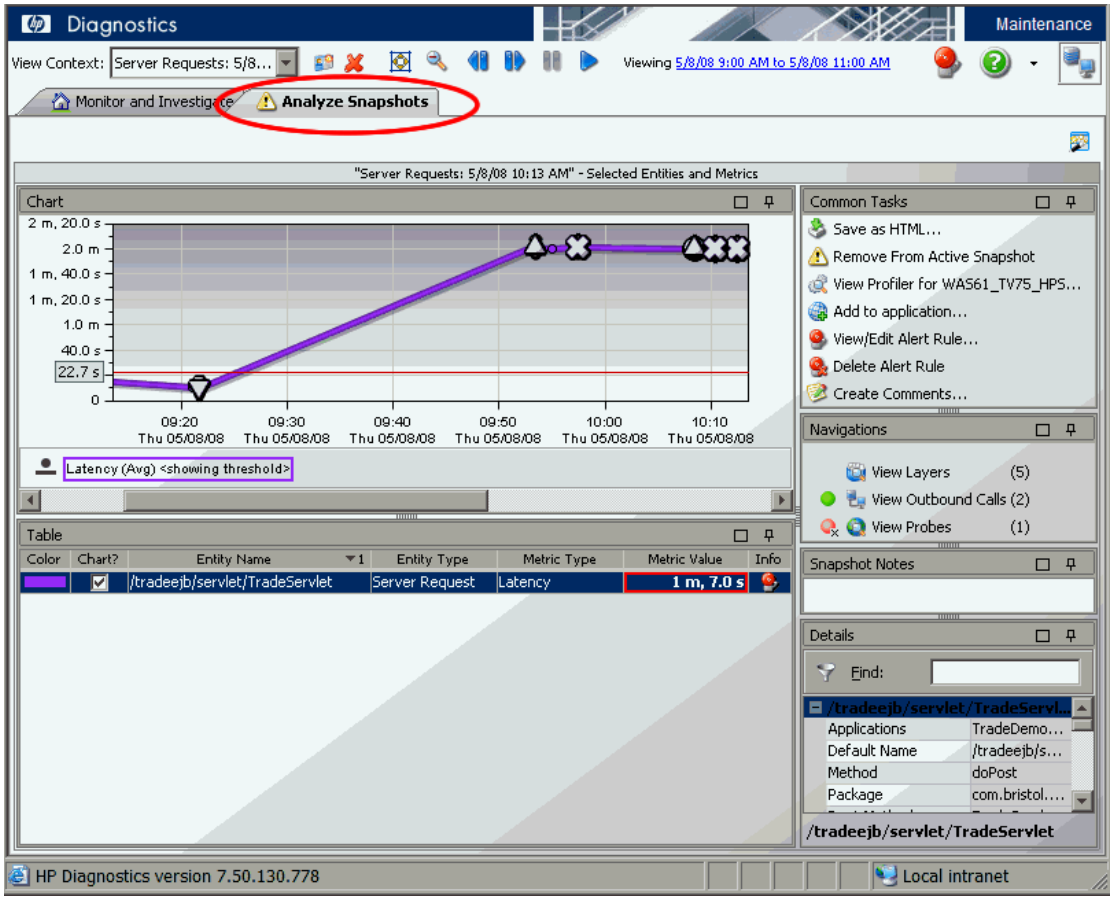
- Drill down to Diagnostics from a business transaction instance that has Java events
- Drill down to Diagnostics from an event

Drill down to Diagnostics Server Requests data from a business transaction instance

In TransactionVision views, select any business transaction instance that has Java events (Transaction Tracking details display in TransactionVision).

Select the drill to Diagnostics icon located next to the business transaction or transaction step. The drill to Diagnostics icon will be displayed whenever there is a server request associated with the business transaction or the transaction step in the Transaction Tracking detail report.

The Diagnostics UI opens and a Snapshot view of the server requests corresponding to this business transaction is displayed.



From the snapshot you can see the average latency of the server request and see how the latency of the particular business transaction step compares with it.

You can also view the average CPU time of the server request to identify if there ia a CPU bottleneck. By navigating to the corresponding probe you can view application server metrics (for example, EJB metrics, JDBC connection pool metrics, heap statistics for the application server VM).

There is an icon displayed in the graph that corresponds to the timestamp and latency of the business transaction step. If this icon coincides exactly with a Diagnostics instance tree icon, there is a high likelihood that the Diagnostics instance tree corresponds to the same instance as the business step in TransactionVision.

Diagnostics displays the data in a snapshot (see the Analyze Snapshots tab is selected in the screen shot above). In general the UI controls in the Analyze Snapshots mode works the same as the controls in the Monitor and Investigate mode. The differences are described in Chapter 8, “Performing Snapshot Analysis”.

If the server request is not found when drilling down to Diagnostics it is most likely due to latency trimming on the Diagnostics side where server requests that take less than 51ms are not collected. Latency trimming is configurable on the probe. For the Java Probe this is changed by setting **minimum.fragment.latency** to 0ms in the **dispatcher.properties**. For the .NET Probe this is changed by setting **<trim><latency min="0" /></trim>** in the **probe_config.xml** file.

Another reason you might not get a snapshot displayed is if the transaction you drilled down from in TransactionVision is a forwarded event (HTTP redirection with HTTP status codes in the 300s). This is because Diagnostics does not track forwarded requests.

Drill down to Diagnostics Server Requests data from an Event

In TransactionVision views, select any event (Event Analysis details display in TransactionVision).

Select the drill to Diagnostics icon located next to a TransactionVision URI.

The Diagnostics UI opens and a Snapshot view of the server requests corresponding to the event for this URI is displayed. The information displayed in the snapshot and how it is used was described earlier.

Viewing Diagnostics Data in LoadRunner

LoadRunner and Diagnostics are integrated products that have been designed to work together to provide information to help you understand and improve the performance of your applications.

During a LoadRunner load test scenario, you can view Diagnostics data for the whole scenario or you can drill down to HP Diagnostics data from a particular transaction. Refer to the *HP LoadRunner Controller User Guide* for details about how to access Diagnostics during a load test scenario.

LoadRunner displays monitoring versions of the Transactions view, Server Requests view, Load view and Probe views for the current run. During the load test scenario, you can navigate to other views in HP Diagnostics.

Analyzing Offline Diagnostics Data

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.

LoadRunner Analysis provides specific graphs that display Diagnostics performance metrics that were captured during the load test scenario.

After you have completed your load test scenario run, you can use LoadRunner Analysis to analyze offline Diagnostics data that was generated during the run.

There are two groups of Diagnostics graphs presented in LoadRunner 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 HP Diagnostics graphs in LoadRunner Analysis, refer to the *HP LoadRunner Analysis User Guide*.

Note: The following features and functionality in LoadRunner Analysis are different from the Diagnostics online views:

- Data in the Profiler that is not sent into the Diagnostics server will not be included in Analysis after the Performance Center and LoadRunner runs. This includes LWMD, Heap Breakdown and the Summary view.
- Oracle 10g data will not be displayed in Analysis.
- System metric and JMX data will not be displayed in Analysis.
- Instance trees will not be available in the Analysis, only aggregate trees are available after the needed drill down.

You can improve the transfer time and load time of the offline analysis files that include Diagnostics data by lowering the resolution of the .eve files.

Use the **bucket.lr.offline.duration** property and the **bucket.lr.offline.sr.duration** properties in the **server.properties** file on the Diagnostics server to increase the aggregation period.

Viewing Diagnostics Data in Performance Center

Before you can use HP 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 *HP Performance Center Administrator Guide*.

During a load test, you can drill down to HP Diagnostics data for the whole load test or for a particular transaction. HP 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 349.

After you have run your load test, you can use LoadRunner Analysis to analyze offline Diagnostics data generated during the load test.

Note: For more information about configuring Diagnostics to work with a firewall see the *HP Diagnostics Installation and Configuration Guide*. For more more information about configuring Performance Center to work with Diagnostics in a firewall environment see the Performance Center Documentation Library.

For information on viewing Diagnostics data in Performance Center see the HP Performance Center Documentation Library.

45

Diagnostics Configuration (Components) View

Much of the configuration of Diagnostics is done using property files but Server and Probe Configuration UIs are provided for some common administration tasks such as license management, setting authorizations (permissions), security, scheduling, logging and viewing lists of Diagnostics files. The configuration UI displays the **Components view**.

See the *HP Diagnostics Installation and Configuration Guide* for details on configuring and administering Diagnostics.

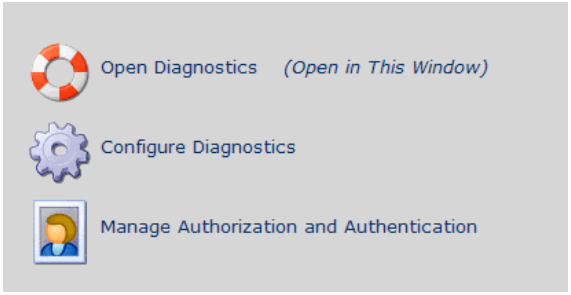
This chapter includes:

Accessing the Components View

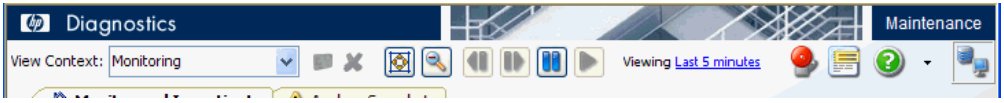
You can access these configuration UIs (Components View) from the Diagnostics Server main window and from a probe system's Admin main window.

To access the Components view from the Server:

1 From the Diagnostics Server main window you can select the **Configure Diagnostics** link to access the configuration UI for Diagnostics.



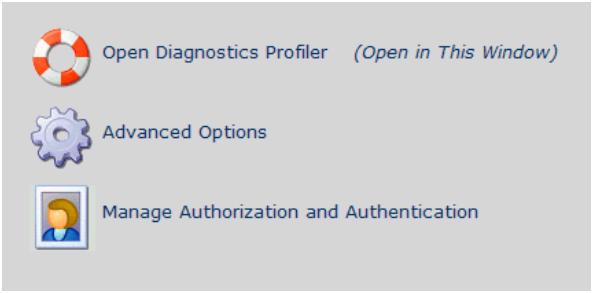
Or when you are in the Diagnostics UI you can select the **Maintenance** link in the upper right corner to access the configuration UI.



2 This displays the Components view for the Server. Note that there are basic and advanced options available in the Components view.

To access the Components view from the probe:

1 From the Diagnostics probe system you can open the Admin window and select the **Advanced Options** link to access the configuration UI for that probe system.



2 This displays the Components view for the probe system. Note that there are basic and advanced options available in the Components view.

Description of the Diagnostics Components View

The Diagnostics Components view for configuration and maintenance has a list of links that provide access to the following:

- Information about the Diagnostics deployment
- Web pages you can use to configure some aspects of Diagnostics
- Access to administration functions for Diagnostics
- Access to maintenance functions for Diagnostics

Note: Refer to the *HP Diagnostics Installation and Configuration Guide* for details on configuration and administration tasks.

46

Glossary of Terms and Metric Descriptions

A glossary of terms used in Diagnostics is provided as well as descriptions for metrics collected by Diagnostic's probes and collectors (see "Diagnostics Metrics Descriptions" on page 674).

This chapter includes:

Glossary of Terms

A

ADO (ActiveX Data Objects) – An easy-to-use, application-level interface to Microsoft's data-access paradigm, OLE DB, .NET transaction monitors use ADO as the transaction type for ADO transactions.

Alert – Indicates that a threshold has been exceeded.

Alert notification – An e-mail or SNMP sent when an entity's status becomes critical.

Alert rules – Rules that instruct Diagnostics to send an alert notification when an entity's status becomes critical.

Alias attribute – Name assigned to an entity to makes it more recognizable.

Allocation – The task of fulfilling an allocation request, which involves finding a block of unused memory of a certain size in the heap.

Alternate y-axis – Displayed when one or more metrics on a graph are measured with different units or on a different scale than those plotted on the y-axis.

Application - Packaged software that provides functionality designed to accomplish a set of related tasks. This software is the user application that is monitored, diagnosed, or both for performance problems.

Application lifecycle - The different development phases for an application, from pre-production to post-production. These phases include develop, build, test, and monitor in production.

Application server - A node on which application server software is installed. Application server software provides the execution environment for the business logic for an application program (for example, BEA WebLogic Server and IBM WebSphere Application Server).

Arguments - Any arguments specified to a method instance. SQL statements are captured as if they are method arguments.

Auto Threshold Setting - Suggesting a threshold value, based on the latency average and standard deviation, that a given percent of measurements should fall at or below.

Average latency - The average amount of time taken by an entity. This is simply the entity's total time divided by the entity's count. (This value is technically the mean.)

B

BAC - Business Availability Center.

BPM - Business Process Manager.

BPM profiles - Business Process Manager profiles.

Breadcrumb - A navigational mechanism at the top of most Diagnostics views that provides a way to determine the context for the information that is presented in the view. This mechanism provides navigation to retrace your steps in your performance analysis.

Bucket - The period of time into which data is aggregated. For example, for a given five-second “bucket,” a fragment or method would have to end in that five-second period of time for its data to be included into the bucket.

Business Process Monitor - Proactively monitoring enterprise applications in real time, and identifying performance problems before users experience them. This process includes monitoring sites from various locations, emulating end-user experience, and assessing site performance from different client perspectives. Business Process Monitor serves as one of the Business Availability Center data collectors.

C

Call profile - The visualization used to present the program trace. Diagnostics probes work by adding instrumentation before and after important methods in your application to provide a simplified program terrace that includes elapsed times.

CAM - See Composite Application Management.

CIs - Configuration items.

CMDB - Configuration Management Database. A database that contains all relevant details of each configuration item, as well as the details of the important relationships between configuration items.

Collector (Oracle 10g database, SAP ABAP) - A Diagnostics component you install to gather data from SAP ABAP systems or Oracle 10g databases.

Commander mode - Mode used by the Diagnostics Server responsible for the command and control functions between the various Diagnostics components and components of other HP Software products integrated with Diagnostics.

Common tasks - Selections in a Diagnostics view that are relevant to the data being displayed.

Composite application management - The monitoring and management of composite applications that use business logic, as well as data that may span composite components, such as Web servers, Java application servers, integration middleware, and mainframe systems.

Concise views - Diagnostics views that show a summary view of the data, with limited functionality in the graph and drill down.

Count - How many total times an entity has been invoked.

Critical (red status indicator) - Indication that the component is consistently exceeding defined thresholds.

D

Details pane - Pane that presents the metrics and custom attributes associated with the entity selected.

E

ELT - End of Logical Transaction. An event that is sent out (by LoadRunner) to Diagnostics when a Transaction instance is completed.

End Time - For a particular instance of an entity, this is the absolute date and time at which the instance completed its execution, as defined by the last point captured. If the entire entity is not instrumented, it is possible that this time is not the actual end time of the entity.

Entity - A transaction, server request, layer, or method.

EVE - File format used by Load Runner Analysis.

Event - A change in the state of an entity that is triggered by threshold violations.

F

Filters - See View filters.

Fragment Arc - A data structure used to represent the presence of a particular kind of inbound or outbound call, as captured by a probe, and possibly aggregated in the server over a given time period.

G

Garbage collection (GC) - A common name for automatic memory management. Memory can be allocated and used. GC automatically frees any chunks of memory no longer referred to.

Good (green status indicator) - Indication that the entity is performing within defined thresholds.

H

Heap - A large pool of unused memory from which memory is allocated.

Hosts - Machines that host the probes and the applications they are monitoring.

HTTP (Hypertext Transfer Protocol) - The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the Web. In the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), this is an application protocol.

HTTPS (Hypertext Transfer Protocol over Secure Socket Layer) - A Web protocol that encrypts and decrypts user page requests, as well as the pages that are returned by the Web server. (By default, HTTPS uses port 443 instead of HTTP port 80 in its interactions with the lower layer, TCP/IP.)

I

Inbound call - A remote call, specifically as recorded by the callee (the one who receives the call).

Instance trees - When capturing program traces, at five-minute intervals, Diagnostics uses an algorithm to select the most interesting instance trees for each server request.

Instrumentation Point - An instrumented location in the application code that is captured by the probe. All capture points are defined in a points file.

J

J2EE (Java 2 Platform, Enterprise Edition). A platform-independent, Java-centric environment from Sun that is used to develop, build, and deploy Web-based enterprise applications online. The J2EE platform manages the infrastructure, and supports the Web services, to enable development of secure, robust, and interoperable business applications. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications. A J2EE application is a deployable unit of J2EE functionality. This unit can be a single module or a group of modules packaged into an archive file with a J2EE application deployment descriptor.

Java probe - Diagnostics software component for monitoring applications running in Java application servers.

JAAS - Java Authentication and Authorization Service.

JVMTI - Java Virtual Machine Tool Interface. A native interface introduced by Java 5 that is a foundation of the Heap Walker implementation.

K

KPI - Key Performance Indicators (KPIs) and Application KPIs.

L

Latency - See Average Latency.

Layers - The name of a layer (a user-specified grouping of methods) or sub-layer, as assigned in the probe's "points" file. Performance metrics for the classes and methods invoked by your applications are grouped into layers and sub-layers.

LR - LoadRunner.

LWMD - Light weight memory diagnostics.

M

Magnification zooming - Mechanism that provides a magnified view of data points. No additional information is retrieved.

Max Time - The maximum amount of total time any instance of a particular entity has taken from start of the first captured point until the end of the last captured point. If the beginning or end of an entity is not captured, this time may not correspond to the actual time taken by the entity.

Memory allocation - Dynamic memory allocation is the allocation of memory storage for use in a computer program during the run time of that program. It is a way of distributing ownership of limited memory resources among many pieces of data and code. A dynamically allocated object remains allocated until it is de-allocated explicitly, either by the programmer or by a garbage collector. This type of allocation is quite different from automatic and static memory allocation. It is said that such an object has dynamic lifetime.

Message Channel - Communication between queue managers relies on a separate program, called a "channel." The Channel runs on the same host as the queue manager, and handles sending and receiving data over the network.

Message Queue - Objects that store messages in an application.

Metric - A standard of measurement. Software metrics are the statistics describing the structure or content of a program. A metric should be a real objective measurement of something such as number of bugs per lines of code. Metrics can include averages, minimum and maximum values, rates, and so on.

Min Time - The minimum amount of total time any instance of a particular entity has taken, from the start of the first captured point until the end of the last captured point. If the beginning or end of an entity is not captured, this time may not correspond to the actual time taken by the entity.

MTTR - Mean Time to Repair.

N

.NET probe - A Diagnostics component that is responsible for capturing .NET events from an application, aggregating the metrics, and sending the performance metrics to a Diagnostics Server.

NII - Non-Identifying Information. Aspects of the data model that are not used to establish the identity of the data object, but that provide additional detail and descriptions about the object.

O

Oracle 10g collector - A Diagnostics component you install and configure to gather data from an Oracle 10g database.

Outbound Calls - A remote call, specifically as recorded by the caller (the one who makes the call). Diagnostics monitors the following types of outbound calls: Web service, RMI, RFC (SAP), CICS, JMS.

P

PC - Performance Center.

Platform - Applicable to virtual machines. Indicates whether the data is coming from a Java or .NET application.

Points file (auto_detect points file) - Defines what the probe captures. Includes methods. Defines whether LWMD data, synchronization data, or both are captured. Associated with each method specification is a layer name, which is passed from the probe all the way to the user interface for reporting purposes.

Portal - A Web site that provides a single point of access to a variety of enterprise data and applications, presenting a unified and personalized view of this information. A portal is made up of portlets.

Portlet - A specialized content area that occupies a small window of a portal page. Also refers to the component in an application server representing content for this window in the portal.

Probe Arc - A data structure used to represent a set of inbound or outbound calls, as captured by a probe, and possibly aggregated in the server by source or target.

Probe groups - Part of the Diagnostics data model. You define probe groups when installing probes.

Probes - A Diagnostics component that is responsible for capturing events from an application, aggregating the metrics, and sending the performance metrics to a Diagnostics Server. The application must first be instrumented, and configured to load the probe at startup. Currently, the Diagnostics product has two probes: Java and .NET.

Profiler (Profiling) - Provides detailed diagnostics information about the application that is being monitored by a probe. This information can be viewed from within the Diagnostics UI or from the Profiler's own UI.

Q

Queue Manager - The primary component of an IBM WebSphere MQ installation. It provides a logical container for the message queue. It is also responsible for transferring data to other queue managers by using message channels.

R

Resolution zooming - Mechanism that provides a higher resolution view of metrics. Additional data is retrieved to chart metrics at a higher resolution.

RMI - Remote Method Invocation.

RUM - Real User Monitor.

S

SAP ABAP collector - A Diagnostics component you install and configure to gather data from a SAP ABAP system.

Server requests - A URI or method that is the first point captured by the probe. One example when a server request is a method and not a URI, is a cross-VM call (perhaps by using RMI).

Server request name - The name of the entry point to a server request.

Snapshot - Mechanism used by Diagnostics to capture and store the performance of your application, when you see a performance anomaly. It includes all of the metrics, but it is limited to the time period specified.

SOA - Service Oriented Architecture.

SQL statements - The complete SQL statement invoked in a database call, as captured by the probe. Depending on the probe's mode, more or less detail about the SQL statement may be captured.

Standalone mode - Using Diagnostics as a standalone product, rather than integrated with other HP software products.

Standard deviation - Measures the dispersion of the total time taken by all instances of a particular entity.

Start time - For a particular instance of an entity, the absolute date and time at which the instance started, as defined by the first point captured. If not all of the entity is instrumented, it is possible that this is not the actual start time of the entity.

Status indicator - The severity of the violation, indicated by a color highlight (none, yellow, or red) shown for the metric value in the Diagnostics displays.

Status view - A table displaying the probe groups, their probes, and the hosts for those probes. For each level in the table, you see the status compared to the thresholds set for the metrics.

Storage Threshold - The amount of space that has been allocated for the TSDB.

T

Thread (thread name) - The name the user has specified for a transaction. This is should be the exact same value the user sees when the transaction is defined (either in Load Runner, PC, or BAC).

Threshold violation - When the value of a metric exceeds the threshold value set. Used to determine status (status = Good, Warning, or Critical).

Thresholds - A value that represents an acceptable limit for a performance metric.

Tiers - As an example, a three-tier is a type of client/server architecture that consists of three well-defined and separate processes, each running on a different platform: the user interface (the client), functional modules that process data (middle tier, which is often called application servers), and a database management system that stores data required by the middle tier, and runs on a database server.

Topology - The pattern of links connecting pairs of nodes on a network or system. For enterprise business systems, nodes may represent abstract or logical entities, and links may represent communications on a service or transactional level.

Total Time - The total amount of time taken by all instances of a particular entity. This time includes any time taken by the entity's children. The time is measured from the start of the first captured point until the end of the last captured point. If the beginning or end of an entity is not captured, this time may not correspond to the actual time taken by the entity.

Transactions - A unit of work performed by an application. In particular, a unit of work that is important to measure and monitor. The response time of this unit of work is used to as a measure the health of the application. Examples include HTTP requests, middleware method invocations, and so on.

Transaction name - The name the user has specified for a transaction. This name should be the exact same value the user sees when the transaction is defined (either in Load Runner, PC, or BAC).

Transaction tracing - Monitoring the performance of a transaction, from end-to-end. A trace for an individual transaction's execution goes across a distributed environment, including the time spent in each component along the execution path.

Trend line - A line on a graph that indicates a statistical trend.

TSDB - Time Series Database.

V

View filters - Filters used to control the type and scope of information that is displayed in a Diagnostics view.

Views- Diagnostics displays of specific types of data in both tabular and graphical formats. Different views can be selected from the View Bar.

Virtual Machine - The process that hosts an application under test. When the application is being monitored, diagnosed, or both, the process also hosts the probe.

VM heap - See Heap.

W

Warning (yellow status indicator) - When the component is occasionally, but not consistently, exceeding defined thresholds.

Web service - A software system designed to support interoperable Machine-to-Machine interaction over a network. Often used to refer to clients and servers that communicate XML messages that follow the SOAP standard.

X

x-axis - The scale that runs across the bottom of a chart or graph. This horizontal axis indicates the time range and scale used to plot the metrics on the graph.

XML (Extensible Markup Language) - A set of rules for text format design that allows the structuring of data. These rules enable a computer to generate and read data, and ensure that the data structure is unambiguous. The rules avoids common pitfalls in language design: they are extensible, platform-independent, and support internationalization and localization.

Y

y-axis - The scale that runs up the side of a chart or graph. This vertical axis indicates the metric types, units, and scale for one or more of the related metrics that are charted on the graph.

Diagnostics Metrics Descriptions

The following sections give definitions for the metrics collected by Diagnostics probes and collectors.

General Performance Metric Descriptions

Metric	Definition
average latency	The average response time across all instances.
average successful_latency	The average response time across all instances of the call that did not generate a SOAP fault.
average CPU	Average CPU utilization.
average ContextSwitchesPerSec	The average number of context switches per second
average MemoryUsage	The average amount of memory in use.
average VirtualMemoryUsage	The percentage of virtual memory in use.
average NetworkIOPerSec	The average amount of network I/O per second.
average PageInsPerSec	The average number of virtual memory pages swapped in per second.
average PageOutsPerSec	The average number of virtual memory pages swapped out per second.
average DiskBytesPerSec	The average number of bytes read and written per second.
average NetworkBytesPerSec	The average number of bytes sent and received per second.

Metric	Definition
average DiskIOPerSec	The average amount of Disk I/O per second.
average HeapUsed	The average amount of heap used by the application.
average HeapFree	The average amount of heap free for use by the application.
average HeapTotal	The average total size of the heap.
average Availability	The percentage of time the application was running.
average Long_Requests	The number of incomplete Server Requests for this.
count latency	The total number of instances.
exception_count latency	The number of instances that failed because the code threw an exception.
timeout_count latency	The number of instances that failed to complete in a reasonable amount of time.
percent_contribution latency	The percentage of the time contributed to the {contrib} total time.
latency_contribution latency	The amount of time contributed to the {contrib} total time.

Metric	Definition
throughput latency	The rate of requests for the selected time period. Throughput and load latency are calculated across granularities (5m, 20m, etc) which are also aligned on 5m boundaries. For small scenario time ranges selected in the Diagnostics UI hosted in LoadRunner, the throughput/load is based on the number of granularities, which is usually larger than the time range specified in the LoadRunner UI. For example, if the scenario time is 00:00:00-00:05:18, the throughput/load is calculated based on 10m (two 5m granularities, 00:00:00-00:04:59 and 00:05:00-00:09:59).
load latency	How much work this layer performs.

System Metric Descriptions

Metric	Definition
average.ASP.NET_v1.1_Requests/Sec	The average number of ASP.NET 1.1 requests per second.
average.ASP.NET_v2.0_Requests/Sec	The average number of ASP.NET 2.0 requests per second.
average Application_Restarts	The number of application restarts that have occurred.
average GC_Collections/sec	The average number of memory garbage collections per second.
average GC_Time_Spent_in_Collections	The percentage of time spent performing garbage collections.
average HeapCommitted	The average size of the committed heap memory.

Metric	Definition
average Heap_Buffer_Space	The average size of the heap memory buffer space.
average Requests_Queued	The average number of requests queued.
average Requests/Sec	The average number of requests per second.
average Threads_Avg_Pool_Size	The average size of the application thread pools.
average Threads_Created/sec	The average number of Java threads created per second.
average Threads_Current_Count	The current number of Java threads.
average Threads_Current_Daemon_Count	The current number of Java daemon threads.
max latency	The maximum response time across all instances.
min latency	The minimum response time across all instances.
rel_std_dev latency	The relative standard deviation of response time across all instances.
std_dev latency	The standard deviation of response time across all instances.
total latency	The total latency across all instances.
total threshold_violations	The total number of threshold violations across all instances.
value_over_threshold latency	The percentage of requests over the threshold.
value_under_threshold latency	The percentage of requests under the threshold.

Metric	Definition
value_over_threshold Availability	The percentage of time the application was over the threshold.
value_under_threshold Availability	The percentage of time the application was under the threshold.
value_over_threshold CPU	The percentage of time the CPU was over the threshold.
value_under_threshold CPU	The percentage of time the CPU was under the threshold.
value_over_threshold HeapUsed	The percentage of heap memory in use over the threshold.
value_under_threshold HeapUsed	The percentage of heap memory in use under the threshold.
value_over_threshold HeapFree	The percentage of free heap memory over the threshold.
value_under_threshold HeapFree	The percentage of free heap memory under the threshold.

J2C Metric Descriptions

Metric	Definition
average J2C_Connections_Closed	The number of closed J2C connections.
average J2C_Connections_Created	The number of J2C connections created.
average J2C_Free_Pool_Size	The size of the free J2C connection pool.
average J2C_Pool_Size	The size of the J2C connection pool.
average J2C_Use_Time	The average time a J2C connection is in use.

Metric	Definition
average J2C_Wait_Time	The average time a J2C connection waits before service.
average J2C_Waiting_Thread_Count	The number of threads waiting for J2C service.

EJB Metric Descriptions

Metric	Definition
average EJB-Cache_Access_/_sec	The number of EJB cache accesses per second.
average EJB-Cache_Beans_Cached	The number of EJB beans in the cache.
average EJB-Cache_Get_Failures_/_sec	The number of EJB cache hit failures.
average EJB_Committed_Transactions_/_sec	The number of committed EJB transactions per second.
average EJB_Creates	The number of EJB creation events.
average EJB_Passive_Beans	The number of passive EJB beans in the server.
average EJB_Pools_Size	The number of beans in the EJB pool.
average EJB-Pool_Access_/_sec	The number of EJB pool accesses per second.
average EJB-Pool_Available_Beans	The number of available EJB beans.
average EJB-Pool_Beans_in_Use	The number of EJB beans in use.
average EJB-Pool_Current_Waiters	The number of EJB requests waiting on the pool.
average EJB-Pool_Get_Failures_/_sec	The number of EJB pool access failures per second.

Metric	Definition
average EJB-Pool_Get_Timeouts_/_sec	The number of EJB pool timeouts per second.
average EJB_Ready_Beans	The number of EJB beans in the ready state.
average EJB_Removes	The number of EJB remove events.
average EJB_Response_Time	The average response time of EJB beans.
average EJB_Rolled_Back_Transactions_/_sec	The number of rolled-back EJB transactions per second.
average EJB_Timed_Out_Transactions_/_sec	The number of timed-out EJB transactions per second.
average EJB_Total_Method_Calls	The total number of EJB method calls across all enterprise beans.

JMS Metric Descriptions

Metric	Definition
average JMS_Current_Bytes	The number of total bytes in the JMS server.
average JMS_Current_Connections	The number of current JMS connections.
average JMS_Current_Destinations	The number of current JMS destinations.
average JMS_Current_Messages	The number of current JMS messages.
average Depth_of_all_Queues	The average depth of all JMS queues.
average JMS_Deployed_Servers	The number of deployed JMS servers.
average JMS_Incoming_Bytes_/_sec	The number of incoming bytes passed through the JMS server per second.

Metric	Definition
average JMS_Incoming_Messages_/_sec	The number of incoming messages passed through the JMS server per second.
average JMS_Pending_Bytes	The number of pending bytes on the JMS server.
average JMS_Pending_Messages	The number of pending messages on the JMS server.

Database Metric Descriptions

Metric	Definition
average JDBC_Active_Connections	The number of active JDBC database connections.
average JDBC_Concurrent_Waiters	The number of concurrent requests waiting for a JDBC connection.
average JDBC_Connection_Closes	The number of JDBC database connections closed.
average JDBC_Connection_Creates	The number of JDBC database connections created.
average JDBC_Connections_Created_/_sec	The number of JDBC connections created per second.
average JDBC_Create_Connection_Delay	The time delay incurred when creating a new JDBC database connection.
average JDBC_Current_Capacity	The current JDBC database connection capacity.
average JDBC_Free_Pool_Size	The size of the JDBC free connections pool.
average JDBC_Leaked_Connections	The number of leaked JDBC database connections.

Metric	Definition
average JDBC_Percent_Used	The percentage of JDBC connections in use.
average JDBC_Pool_Size	The overall JDBC connection pool size.
average JDBC_Reconnect_Failures	The number of JDBC database reconnect attempts that have failed.
average JDBC_Requests_Waiting_for_Connection	The number of JDBC database requests waiting for a database connection.
average JDBC_Use_Time	The average amount of time a JDBC connection is in use.
average JDBC_Wait_Seconds_High	The maximum time a request has waited for a JDBC database connection.
average JDBC_Wait_Time	The average amount of time a JDBC connection waits before being serviced.
average JTA_Active_Transactions	The number of active JTA transactions.
average JTA_Committed_Transactions_/_sec	The number of committed JTA transactions per second.
average JTA_Rolled_Back_Transactions_/_sec	The number of rolled-back JTA transactions per second.
average JTA_Transactions_/_sec	The number of JTA transactions per second.
average Timeout/Estimated_Events_(per_minute)	The estimated number of timeout events per minute.
average Transactions/Active	The number of active transactions.
average Transactions/Committed	The number of committed transactions.

Metric	Definition
average Transactions/Rolled_Back	The number of rolled back transactions.
average Transactions/Suspended	The number of suspended transactions.
average Transactions/Timed_Out	The number of timed out transactions.
average TM_Active_Global_Txns	The number of active global transactions reported by the Transaction Manager.
average TM_Global_Txns_Comitted	The number of committed global transactions reported by the Transaction Manager.
average TM_Global_Txns_Rolled-Back	The number of rolled back global transactions reported by the Transaction Manager.

Web Application Metric Descriptions

Metric	Definition
Application_Threads/Active_Threads	The number of active threads for this application.
average Application_Threads/Current_Pool_Size	The size of the current thread pool for this application.
average Application_Threads/Current_Usage	The percentage of time this application is in use.
average Application_Threads/Initial_Pool_Size	The initial size of the application thread pool.
average Application_Threads/Maximum_Pool_Size	The maximum size of the application thread pool.
average Application_Threads/Minimum_Pool_Size	The minimum size of the application thread pool.

Metric	Definition
average Application_Threads/Queue_Capacity	The maximum size of the application request queue.
average Application_Threads/Queue_Usage	The percentage of request queue capacity currently in use.
average Application_Threads/Waiting_in_Queue	The number of requests waiting in the queue.
average Application_Threads/Waiting_to_Enqueue	The number of requests waiting to enter the queue.
average Execute_Queues_Idle_Threads	The number of idle threads in the execution queue.
average Execute_Queues_Pending_Requests	The number of pending requests in the execution queues.
average Execute_Queues_Requests_/_sec	The number of requests per second in the execution queues.
average Execute_Queues_Total_Threads	The number of total threads in the execution queues.
average HTTP/Cached_Responses	The number of responses served from the cache.
average HTTP/Requests_/_sec	The average number of HTTP requests per second.
average HTTP/Responses_/_sec	The average number of HTTP responses per second.
average HTTP/Total_Response_Time	The total HTTP response time across all instances.
average JNDI/Bound_Objects	The number of bound JNDI objects.
average JNDI/Cache_Size	The size of the JNDI cache.

Metric	Definition
average P4/Failed_Requests	The number of failed requests.
average P4/Requests_/_sec	The average number of requests per second.
average Pool_Manager/Total_Memory_Allocated	The total size of memory allocated to the pool manager.
average Pool_Manager/Total_Memory_Used	The total size of memory currently in use by the pool manager.
average Portal_Pool/Created_Objects	The average number of objects created in a portal pool.
average Portal_Pool/Evicted_Objects	The average number of objects evicted from a portal pool.
average Portal_Pool/Expired_Objects	The average number of objects expired from a portal pool.
average Portal_Pool/Installed_Pools	The number of installed portal pools.
average Portal_Pool/Max_Object_Count	The maximum number of objects allowed in a portal pool.
average Portal_Pool/Pool_Memory_Size	The amount of memory used by the portal pool.
average Portal_Pool/Pooled_Objects	The number of pooled objects in the portal pool.
average Portal_Pool/Total_Pool_Limit	The maximum amount of memory allowed for all pools.

Metric	Definition
average Portal_Pool/Total_Pool_Size	The current amount of memory in use for all pools.
average Portal_Runtime/Loaded_Applications	The number of applications currently loaded.
average Portal_Runtime/Loaded_Component_Contexts	The number of component contexts currently loaded.
average Portal_Runtime/Loaded_Components	The number of components currently loaded.
average Portal_Runtime/Loaded_Services	The number of services currently loaded.
average Security/Active_Sessions	The number of active sessions.
average Security/Invalid_Sessions	The number of invalid sessions.
average Security/Logged_Off_Sessions	The number of logged off sessions.
average Security/Timeout_Out_Sessions	The number of timed out sessions.
average Security/Total_Sessions	The total number of sessions across all instances.
average Security/Unsuccessful_Logons	The number of failed log-on attempts.
average Servlets_Average_Execution_Time	The average execution time of servlet invocations.
average Servlets_Highest_Execution_Time	The highest execution time of servlet invocations.
average Servlets_Invocations_/_sec	The average number of servlet invocations per second.

Metric	Definition
average Servlets/Current_Security_Sessions	The current number of servlet security sessions.
average Servlets/Current_Sessions	The current number of servlet sessions.
average Servlets/Requests_/_sec	The average number of servlet requests per second.
average Servlets_Response_Time	The average response time of all servlets.
average Servlets/Security_Sessions_Invalidated	The number of invalidated servlet security sessions.
average Servlets/Sessions_Invalidated	The number of invalidated servlet sessions.
average Servlets/Timed_Out_Security_Sessions	The number of timed out servlet security sessions.
average Servlets/Timed_Out_Sessions	The number of timed out servlet sessions.
average Servlets_Total_Requests	The total number of requests across all servlets.
average SM_Live_Sessions	The number of live servlet manager sessions.
average System_Threads/Active_Threads	The number of active system threads.
average System_Threads/Current_Pool_Size	The current size of the system thread pool.
average System_Threads/Current_Usage	The percentage of system threads currently in use.
average System_Threads/Initial_Pool_Size	The initial number of threads in the system thread pool.

Metric	Definition
average System_Threads/Maximum_Pool_Size	The maximum number of threads in the system thread pool.
average System_Threads/Minimum_Pool_Size	The minimum number of threads in the system thread pool.
average System_Threads/Queue_Capacity	The maximum size of the system thread queue.
average System_Threads/Queue_Usage	The percentage of the system thread queue currently in use.
average System_Threads/Waiting_in_Queue	The number of requests waiting in the system thread queue.
average System_Threads/Waiting_to_Enqueue	The number of requests waiting to enter the system thread queue.

SAP Specific Metric Descriptions

Metric	Definition
average R3ExtendedMemoryUsed	The average of the maximum extended memory used in dialog step.
average R3PrivateMemoryUsed	The average private memory in use.
average R3BytesRequested	The average number of bytes requested by the application.
average R3DatabaseRequestsToBuffer	The average number of database buffer requests.
average R3DatabaseRows	The average number of database rows returned.

Metric	Definition
average R3DatabaseCalls	The average number of database calls made.
average R3Confidence	The percentage of time latency data that is explicitly attributable to this particular RFC.

Oracle Specific Metric Descriptions

Metric	Definition
average OracleLoad	The average amount of work performed. The sum of child layers can be greater than their parent because of Oracle time measurements not being mutually exclusive.

Allocation Metric Descriptions

Metric	Definition
total objects_alive	The number of allocated objects that are currently alive.
total objects_allocated	The number of objects that have been allocated.
total objects_deallocated	The number of objects that have been de-allocated.
average object_lifespan	The average duration of an allocated objects life.

Index

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Y

Z


invent


Java™
COMPATIBLE