Quest™

Foglight™ for Application Servers 5.9.11

User Guide

Legend

■ **WARNING: A WARNING icon indicates a potential for property damage, personal injury, or death.**

! **CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.**

i **IMPORTANT NOTE**, **NOTE**, **TIP**, **MOBILE**, or **VIDEO:** An information icon indicates supporting information.

Foglight for Application Servers User Guide
Updated - April 2017
Software Version - 5.9.11

Contents

Monitoring Application Servers

This *Foglight for Application Servers User Guide* provides instructions and conceptual information about how to monitor and display .NET and Java EE Technologies metrics, and JMX metrics from JMX-enabled systems.

In addition, this guide provides information about the Application Servers Administration and Monitor dashboards and views. This guide is intended for any user who is interested in monitoring .NET or Java EE systems, or JMX data with Foglight for Microsoft .NET, Foglight for Java EE Technologies, or Foglight for JMX. Guides that are common to these three products are called the *Foglight for Application Servers* guides.

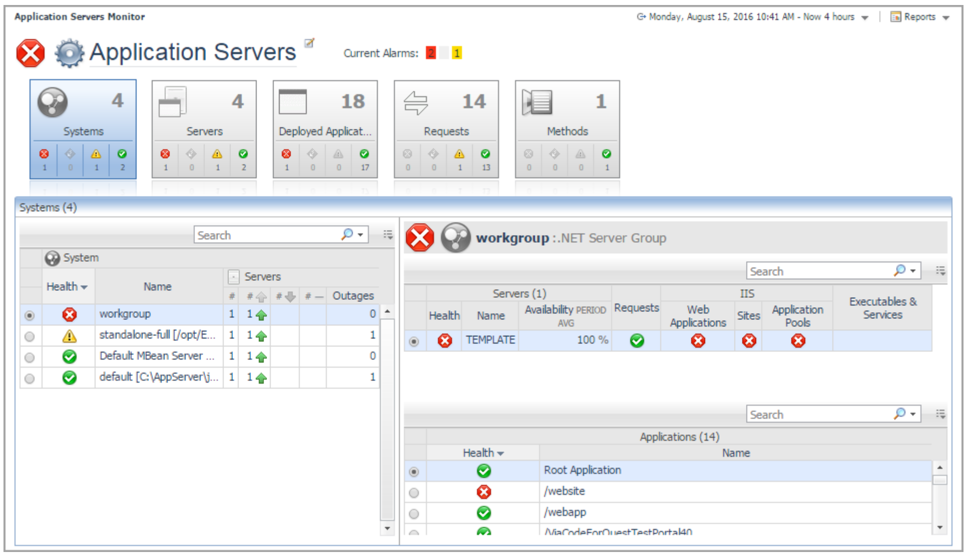
The Application Servers Monitor dashboard provides a centralized location for monitoring your .NET, Java EE Technologies, and JMX systems, servers, deployed applications, requests, and methods. From this dashboard, you can view the health status of your monitored components, and investigate any alarms that occur.

NOTE: If you do not see the Application Servers Monitor dashboard listed in the navigation panel, inform your Foglight System Administrator. Your Administrator can add your user account to one of the following role groups: .NET Administrator, .NET Read-only, Java EE Administrator, or Java EE Read-only User, JMX Administrator, or JMX Read-only User.
You must have one of these roles to be able to see the Application Servers Monitor dashboard.

To view the Application Servers Monitor dashboard:

- On the navigation panel, under Dashboards, click **Application Servers > Monitor**.

Figure 1. The Application Servers Monitor dashboard opens.


Application Servers Monitor
Monday, August 15, 2016 10:41 AM - Now 4 hours
Reports
Application Servers
Current Alarms: 2 1
Search

	Systems	Servers	Deployed Applicat...	Requests	Methods
Total	4	4	18	14	1
⊗	1	1	1	0	0
◈	0	0	0	0	0
⚠	1	1	0	1	0
✓	2	2	17	13	1

Systems (4)

System		Servers				
Health	Name	#	# ↑	# ↓	# —	Outages
⊗	workgroup	1	1 ↑			0
⚠	standalone-full [/opt/E...	1	1 ↑			1
✓	Default MBean Server ...	1	1 ↑			0
✓	default [C:\AppServer\j...	1	1 ↑			1

workgroup :.NET Server Group

Servers (1)				IIS			
Health	Name	Availability PERIOD AVG	Requests	Web Applications	Sites	Application Pools	Executables & Services
⊗	TEMPLATE	100 %	✓	⊗	⊗	⊗	

Applications (14)	
Health	Name
✓	Root Application
⊗	/website
✓	/webapp
✓	/ViaCodeForQuestTestPortal40

NOTE: The tiles and views available on this dashboard depend on the products you have installed. For example, the Deployed Applications, Requests, and Methods tiles are only available when you have installed Foglight for Java EE Technologies or Foglight for Microsoft .NET.

The Application Servers Monitor dashboard displays services as defined by the default Application Servers service category. The Application Servers category contains all Java EE, JMX, and .NET services.

If you want to view only a specific subset of services (such as only *All .NET Systems)*, you can select the service.

To select the service to display:

1 On the navigation panel, under Dashboards, click **Application Servers > Monitor**.

2 On the **Application Servers Monitor** dashboard, click the edit icon .

3 In the **Service Selection** dialog box that opens, select the name of the service you want to display on the dashboard.

   TIP: You may have to expand a service (click the '+' sign) to locate the service you want.

4 Click **OK**.

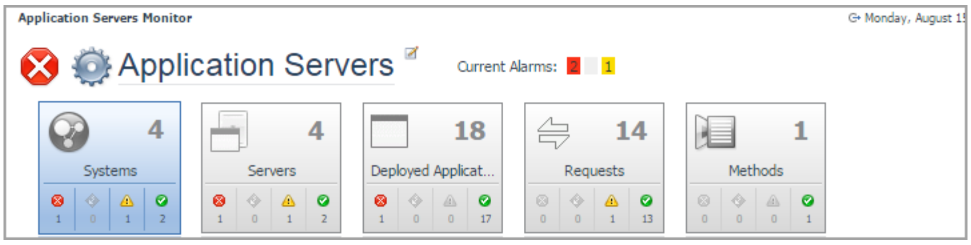
   The dialog box closes and the dashboard refreshes, displaying the selected service.

You can select a different service to view at any time using the same procedure. You can also create your own services.

Using the Application Servers Monitor dashboard

The Application Servers Monitor dashboard organizes your .NET and Java EE Technologies monitoring data into views that are divided into five categories: Systems, Servers, Deployed Applications, Requests, and Methods. These categories are presented as tiles on the dashboard.

Figure 2. The Application Servers Monitor dashboard.

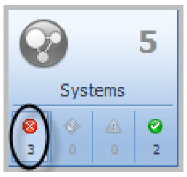


Click a tile to access the views for that category. For more information, see:

- Monitoring Systems on page 12
- Monitoring Servers on page 18
- Monitoring Deployed Applications on page 24
- Monitoring Requests on page 28
- Monitoring Methods on page 51

If you want to view only the components that have triggered alarms for that category, click the health status icon on the tile.

Figure 3. The System tile, with the health status icon circled.



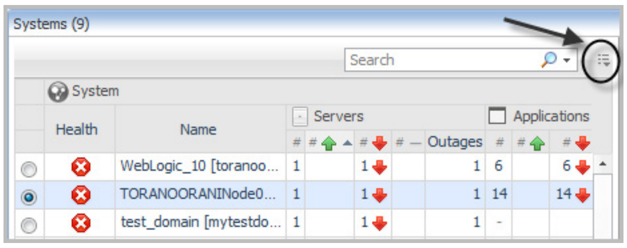
The health status icons act as a filter for the category, allowing you to focus on trouble spots within your system.

Show/Hide table columns

Some columns are not visible by default. You can use the customizer menu to show all available columns, or to reduce the number of columns shown if the information is not pertinent to your monitoring needs.

To show/hide columns:

1 Click the Customizer icon ☰ at the top-right corner of any table.



Systems (9)

Search

	System		Servers					Applications		
	Health	Name	#	# ▲	# ▼	# —	Outages	#	# ▲	# ▼
○	✖	WebLogic_10 [toranoo...	1		1 ▼		1	6		6 ▼
●	✖	TORANOORANINode0...	1		1 ▼		1	14		14 ▼
○	✖	test_domain [mytestdo...	1		1 ▼		1	-		

A list of the columns available to add to the table opens.

2 Select the columns you want to show or hide by selecting or clearing the appropriate check boxes.

3 Click **Apply**.

Reviewing alarms

On the Application Servers Monitor dashboard, click the number next to **Current Alarms:** to open the Alarm Summary dialog box that lists the details of the current alarms.

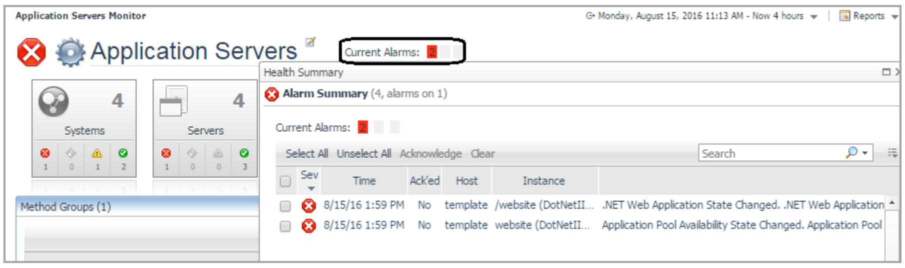


Figure 4. Alarm details on the Application Servers Monitor dashboard.

Application Servers Monitor

Monday, August 15, 2016 11:13 AM - Now 4 hours | Reports

Application Servers | Current Alarms: 2

Systems 4: 1 0 1 2

Servers 4: 1 0 0 3

Method Groups (1)

Health Summary

Alarm Summary (4, alarms on 1)

Current Alarms: 2

Select All Unselect All Acknowledge Clear Search

	Sev	Time	Ack'ed	Host	Instance	
☐	✖	8/15/16 1:59 PM	No	template	/website (DotNetII...	.NET Web Application State Changed. .NET Web Application
☐	✖	8/15/16 1:59 PM	No	template	website (DotNetII...	Application Pool Availability State Changed. Application Pool

Use these links to access more detailed information about the alarms affecting your monitored services.

To view the overall health summary:

- Click the health status icon in the Sev list.

  Or

- Click any value in the Time list.

  The Health Summary pop-up opens. From here, you can acknowledge and clear alarms, or view details of an individual alarm.

For more information about managing alarms, see Viewing, Acknowledging, and Clearing Alarms in the *Foglight User Help*.

Managing services

In Foglight, a service is defined as a grouping of one or more monitored components, such as Java Service Pages (JSPs), servlets, and Enterprise Java Beans (EJBs). Services can be nested within other services, each with their own monitored components. Application Server (Java EE and .NET) services are organized by predefined types such as: all monitored domains, WebSphere cells, OracleAS cluster topologies, JBoss partitions, Tomcat servers, or web applications.

Examining the structure of Application Servers services

The Application Servers service category includes the *All .NET Systems* service, and the *All Java Systems* service, each of which in turn can be comprised of one or more services. For example, the *All Java Systems* service could include the following services:

- WebLogic service that monitors all WebLogic domains
- WebSphere server that monitors all web applications
- OracleAS service that monitors all OracleAS cluster topologies
- JBoss service that monitors all JBoss partitions
- Tomcat service that monitors all Tomcat servers
- Java EE web applications service that monitors all Java EE web applications.

Each service has components or sub-components that are monitored by a rule.

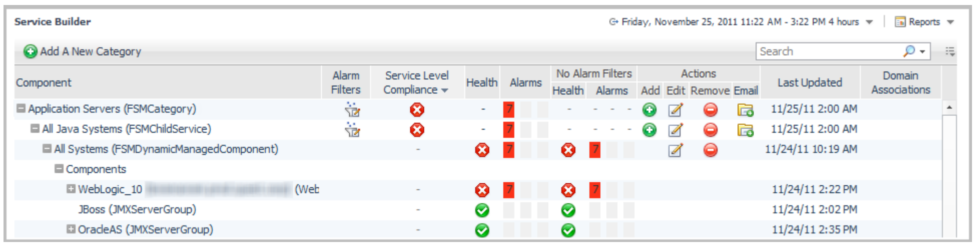
A component has its own alarms and Service Level Compliance (availability) status. The status of each component rolls up to provide an overall Service Level Compliance and alarm status for a service. From the **Service Builder** dashboard, you can explore how a service is built. You may want to examine this composition to see which components are contributing to the poor health.

To examine how a service is built:

1. On the navigation panel, under Dashboards, click **Services > Service Builder.**

   The Service Builder dashboard appears.

2. Expand the **Application Servers** service category. The grouping underneath the service category appears.



Service Builder

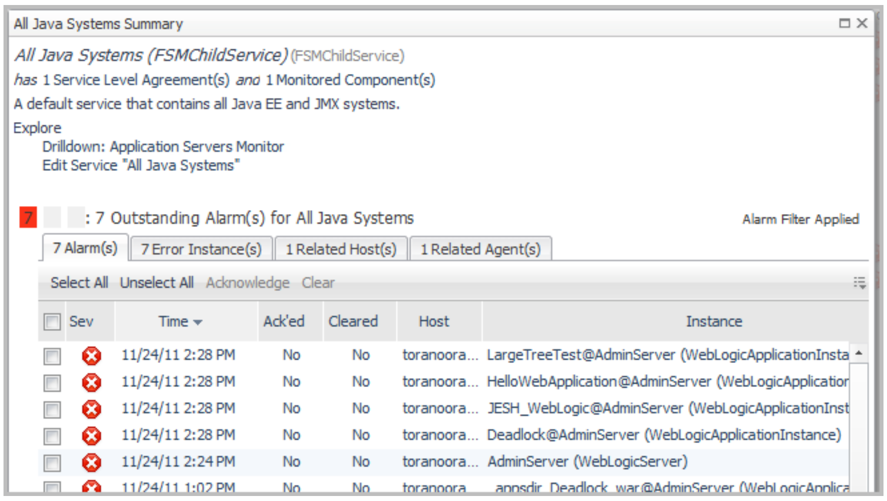
Friday, November 25, 2011 11:22 AM - 3:22 PM 4 hours | Reports

Add A New Category | Search

Component	Alarm Filters	Service Level Compliance	Health	Alarms	No Alarm Filters		Actions				Last Updated	Domain Associations
					Health	Alarms	Add	Edit	Remove	Email		
Application Servers (FSMCategory)		✗	-	7	-	- - -					11/25/11 2:00 AM	
All Java Systems (FSMChildService)		✗	-	7	-	- - -					11/25/11 2:00 AM	
All Systems (FSMDynamicManagedComponent)		-	✗	7	✗	7					11/24/11 10:19 AM	
Components												
WebLogic_10 [redacted] (Web		-	✗	7	✗	7					11/24/11 2:22 PM	
JBoss (JMXServerGroup)		-	✓		✓						11/24/11 2:02 PM	
OracleAS (JMXServerGroup)		-	✓		✓						11/24/11 2:35 PM	

To see more detailed information:

1. Select the **All Java Systems** service.


All Java Systems Summary

All Java Systems (FSMChildService) (FSMChildService)

has 1 Service Level Agreement(s) *and* 1 Monitored Component(s)

A default service that contains all Java EE and JMX systems.

Explore

Drilldown: Application Servers Monitor

Edit Service "All Java Systems"

7 : 7 Outstanding Alarm(s) for All Java Systems — Alarm Filter Applied

7 Alarm(s) | 7 Error Instance(s) | 1 Related Host(s) | 1 Related Agent(s)

Select All Unselect All Acknowledge Clear

Sev	Time	Ack'ed	Cleared	Host	Instance
⊗	11/24/11 2:28 PM	No	No	toranoora...	LargeTreeTest@AdminServer (WebLogicApplicationInsta
⊗	11/24/11 2:28 PM	No	No	toranoora...	HelloWebApplication@AdminServer (WebLogicApplicatior
⊗	11/24/11 2:28 PM	No	No	toranoora...	JESH_WebLogic@AdminServer (WebLogicApplicationInst
⊗	11/24/11 2:28 PM	No	No	toranoora...	Deadlock@AdminServer (WebLogicApplicationInstance)
⊗	11/24/11 2:24 PM	No	No	toranoora...	AdminServer (WebLogicServer)
⊗	11/24/11 1:02 PM	No	No	toranoora...	appsdir_Deadlock_war@AdminServer (WebLogicApplica

The breakdown of the Java systems service is displayed with alarm information. From here you can investigate the Service Level Agreements (SLAs), examine the monitored components, and acknowledge and clear alarms.

2 Under the **Explore** heading, click **Drilldown: Application Servers Monitor** to view the details of the service in the Application Servers Monitor dashboard.

For more information about Services and Service Level Agreements, see Monitoring Your Services in the *Foglight User Guide*.

Monitoring Systems

The Systems and Servers views of the Application Servers Monitor dashboard are common to Foglight for Microsoft .NET, Foglight for Java EE Technologies, and Foglight for JMX. This section describes the features and functionality of these two views, and the related Java Virtual Machine (JVM) view.

The Systems tile provides a high-level overview of the behavior of your systems. Use this view to review groups of monitored domains, cells, cluster topologies, and partitions. From here you can quickly see the number of servers and deployed applications being monitored, their health, availability, and more.

Select an object in the system summary table to view the components that belong to that system, such as servers and deployed applications. When you hover over a component, a dwell displays all its sub-components and any alarms associated with that system.

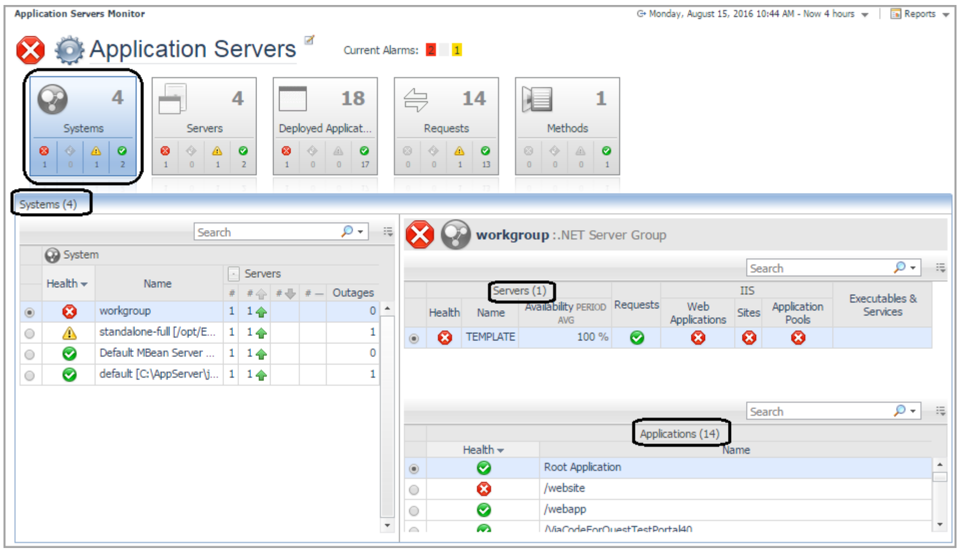
From this view, you can perform the following actions:

- Review the number of monitored systems in your infrastructure
- Search for system types
- Identify the system aggregate state
- Review system servers total and unavailable counts
- Review system deployed applications total and unavailable counts
- Collect Traces (from the Action panel > General tab)

To view the systems:

1 On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.

2 Click the **Systems** tile.



System		Servers				
Health	Name	#	# ↑	# ↓	# —	Outages
❌	workgroup	1	1			0
⚠	standalone-full [/opt/E...	1	1			1
✓	Default MBean Server ...	1	1			0
✓	default [C:\AppServer\j...	1	1			1

Health	Name	Availability PERIOD AVG	Requests	IIS Web Applications	IIS Sites	IIS Application Pools	Executables & Services
❌	TEMPLATE	100 %	✓	❌	❌	❌	

Health	Name
✓	Root Application
❌	/website
✓	/webapp
✓	/ViaCodeForQuestTestPortal40

The Systems view displays the summary of all the monitored systems.

Systems

The Systems table lists all the systems running in your monitored environment. This table can also list the total number of, and availability for, monitored servers and applications in your infrastructure, depending on the columns shown.

Use this table to quickly review the overall system health, and to determine if an error is occurring at the server or application level. Colored icons indicate the health status of the components: green check marks or upward arrows indicate a healthy state, while the red X or downward error indicates an alarm condition. A yellow caution triangle (not shown) indicates that the system is in a warning state and may become unhealthy.

Select a system from the list to populate the System details view with data about the servers, applications, domains, cells, cluster topologies, or partitions in the selected system.

For example, the health of a WebLogic_10 system is poor (✖), because one server and six applications are unavailable (⬇). Click the status icon for an affected component (system, server, or application) to obtain information about the source. For more information, see Reviewing the health of your systems on page 13.

System details

The system details view displays details of the system selected in the System table. Depending on the type of system selected, this view may be divided into two tables: the Servers table (upper right), and the Applications table (lower right). For some systems, the Applications table is not applicable and is therefore hidden.

To review the properties for the selected system, click **Properties** (at the top right corner of the Systems Details view). The information available in this pop-up depends on the type of system selected.

Servers table

The Servers table lists the monitored servers for the selected domain, and provides you with at-a-glance health information for the server and any monitored components. Click any entry to access several different drill-down views associated with the components.

The columns in this table dependentd on the cartridges you have installed and the components you are monitoring. For example, for a system monitored by the JMX Agent, the Servers table includes MBeans and JVM status columns, and the Applications table is not available.

Applications table

NOTE: This table is not applicable to Foglight for JMX.

The Applications table lists all the applications running in a particular domain, cell, cluster topology, or partition and the status of related components such as EJBs, Web Applications, Web Services, and Resource Adapters. Click any entry to access drill-down views that provide detailed information about the selected component.

For more information, see Reviewing the health of your systems on page 13.

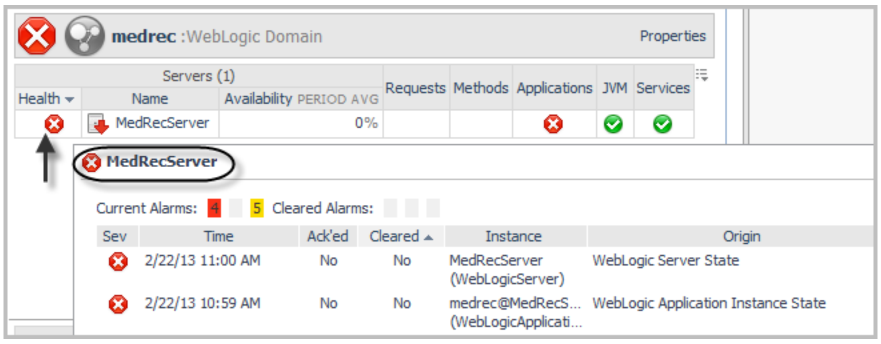
Reviewing the health of your systems

You can obtain at-a-glance information about the health of a domain, cell, cluster topology, partition, or server group, and quickly assess the health of components and sub-components, from the Systems view of the Application Servers Monitor dashboard.

To view the health of a system:

- Hover over any Health icon in the System table of the Systems view to display the Health Summary information for that system, domain, or application.

Figure 5. Health summary information.



medrec :WebLogic Domain   Properties

Servers (1)			Requests	Methods	Applications	JVM	Services
Health	Name	Availability PERIOD AVG					
✖	MedRecServer	0%			✖	✓	✓

MedRecServer

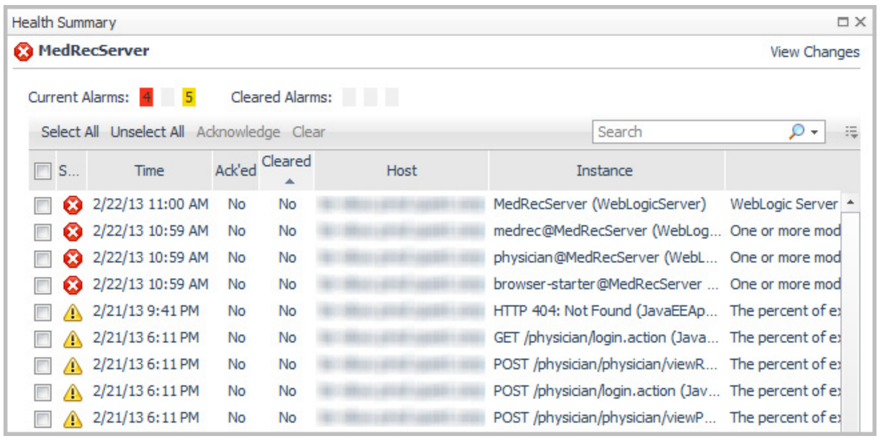
Current Alarms: 4 5 Cleared Alarms:

Sev	Time	Ack'ed	Cleared	Instance	Origin
✖	2/22/13 11:00 AM	No	No	MedRecServer (WebLogicServer)	WebLogic Server State
✖	2/22/13 10:59 AM	No	No	medrec@MedRecS... (WebLogicApplicati...	WebLogic Application Instance State

The table lists all the alarms made for this system.

- Click any health icon to open the Health Summary dialog box, which lists the details of any alarms.

Figure 6. Health summary dialog box.



Health Summary

MedRecServer   View Changes

Current Alarms: 4 5   Cleared Alarms:

Select All Unselect All Acknowledge Clear   Search

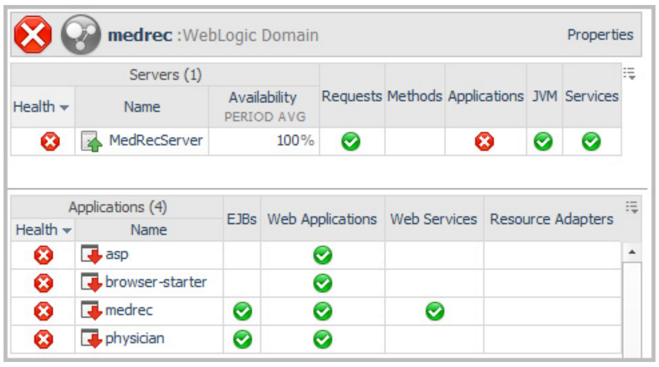
S...	Time	Ack'ed	Cleared	Host	Instance	
✖	2/22/13 11:00 AM	No	No	[redacted]	MedRecServer (WebLogicServer)	WebLogic Server
✖	2/22/13 10:59 AM	No	No	[redacted]	medrec@MedRecServer (WebLog...	One or more mod
✖	2/22/13 10:59 AM	No	No	[redacted]	physician@MedRecServer (WebL...	One or more mod
✖	2/22/13 10:59 AM	No	No	[redacted]	browser-starter@MedRecServer ...	One or more mod
⚠	2/21/13 9:41 PM	No	No	[redacted]	HTTP 404: Not Found (JavaEEAp...	The percent of e:
⚠	2/21/13 6:11 PM	No	No	[redacted]	GET /physician/login.action (Java...	The percent of e:
⚠	2/21/13 6:11 PM	No	No	[redacted]	POST /physician/physician/viewR...	The percent of e:
⚠	2/21/13 6:11 PM	No	No	[redacted]	POST /physician/login.action (Jav...	The percent of e:
⚠	2/21/13 6:11 PM	No	No	[redacted]	POST /physician/physician/viewP...	The percent of e:

In this dialog box, select an alarm to view the full details, or to acknowledge or clear the alarm. Click any value related to an alarm to see more information. For example, click the agent name in the Agent column to review the agent health and related information.

For more information about alarms, see the *Foglight User Help*.

Example: Investigating application health from the Systems view

Figure 7. The medrec WebLogic Domain contains four applications, all of which are in poor health.



medrec :WebLogic Domain — Properties

Servers (1)

Health	Name	Availability PERIOD AVG	Requests	Methods	Applications	JVM	Services
✖	MedRecServer	100%	✓		✖	✓	✓

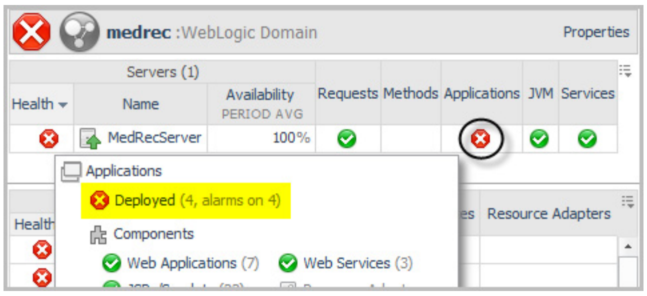
Applications (4)

Health	Name	EJBs	Web Applications	Web Services	Resource Adapters
✖	asp		✓		
✖	browser-starter		✓		
✖	medrec	✓	✓	✓	
✖	physician	✓	✓		

To investigate:

1. In the **Applications** column of the Servers table, click the health status icon.

   A pop-up opens, displaying the health status of the applications broken down into categories (Deployed, Components, and EJBs), and sub-components (such as Web Applications and Web Services).

   Applications

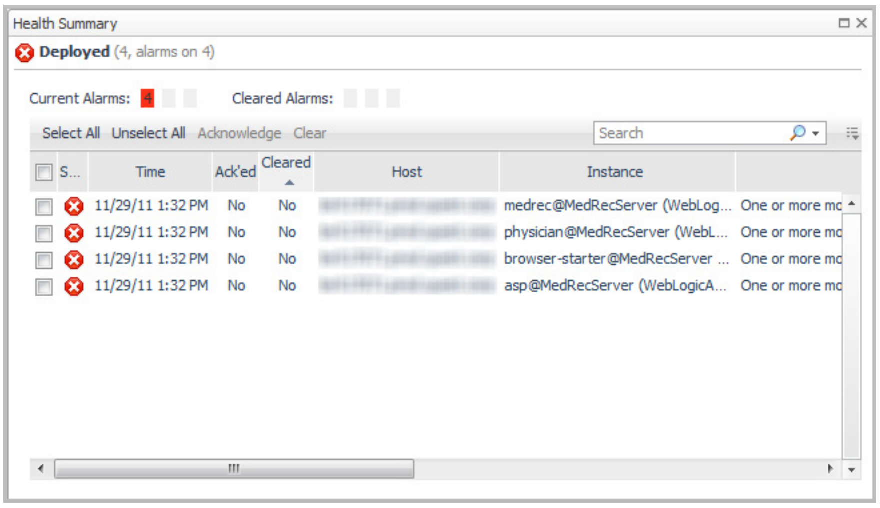
   ✖ Deployed (4, alarms on 4)

   Components

   ✓ Web Applications (7) ✓ Web Services (3)

   The status icon indicates that the problem is in the Deployed category.

2. Click the **Deployed** health status icon.

   Health Summary

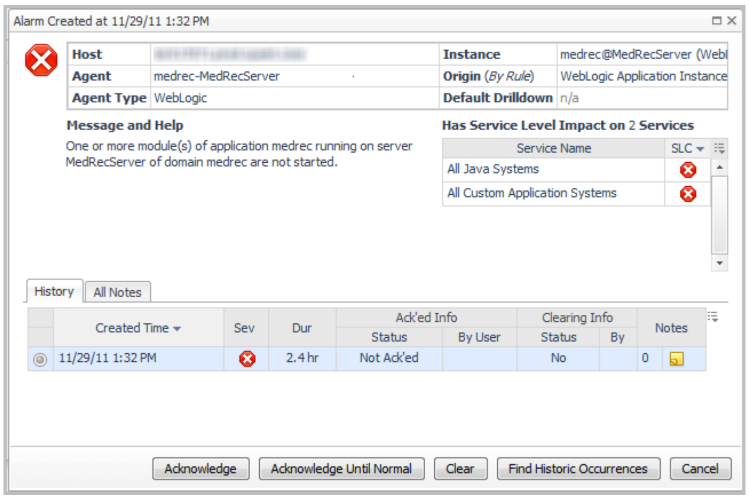
   ✖ **Deployed** (4, alarms on 4)

   Current Alarms: 4 Cleared Alarms:

   Select All Unselect All Acknowledge Clear Search

	S...	Time	Ack'ed	Cleared	Host	Instance	
☐	✖	11/29/11 1:32 PM	No	No	[redacted]	medrec@MedRecServer (WebLog...	One or more mc
☐	✖	11/29/11 1:32 PM	No	No	[redacted]	physician@MedRecServer (WebL...	One or more mc
☐	✖	11/29/11 1:32 PM	No	No	[redacted]	browser-starter@MedRecServer ...	One or more mc
☐	✖	11/29/11 1:32 PM	No	No	[redacted]	asp@MedRecServer (WebLogicA...	One or more mc

3. In the Health Summary view, to review the details of an alarm, click the health status icon.


Alarm Created at 11/29/11 1:32 PM

Host	[redacted]	**Instance**	medrec@MedRecServer (Webl
Agent	medrec-MedRecServer	**Origin** (*By Rule*)	WebLogic Application Instance
Agent Type	WebLogic	**Default Drilldown**	n/a

Message and Help

One or more module(s) of application medrec running on server MedRecServer of domain medrec are not started.

Has Service Level Impact on 2 **Services**

Service Name	SLC
All Java Systems	✗
All Custom Application Systems	✗

History | All Notes

	Created Time	Sev	Dur	Ack'ed Info		Clearing Info		Notes
				Status	By User	Status	By	
●	11/29/11 1:32 PM	✗	2.4 hr	Not Ack'ed		No		0

Use this Alarm dialog box to review the alarm details, such as the full alarm message and its origin, and to acknowledge or clear the alarm. You can also drill down to more information about any named components (such as the host, agent, instance, or service) by clicking the component name.

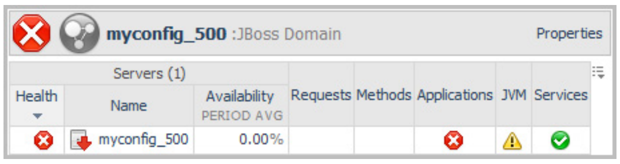
For more information about managing alarms in Foglight for Application Servers, see the *Foglight User Help*.

For more information about the rules that trigger alarms in the Application Servers Monitor dashboard, review the rules on the Foglight Rule Management dashboard (**Dashboards > Administration > Rules & Notifications > Rules**).

Investigating system component health

The health of a system depends on the health of its servers, applications, and components (where applicable). Using the System view of the Application Servers Monitor dashboard, you can drill down to investigate the health of an individual system component.

Figure 8. For example, from the Servers table in the Systems view, you can see that a JBoss Java Virtual Machine (JVM) is experiencing a problem.


myconfig_500 :JBoss Domain   Properties

Servers (1)							
Health	Name	Availability PERIOD AVG	Requests	Methods	Applications	JVM	Services
✗	myconfig_500	0.00%			✗	⚠	✓

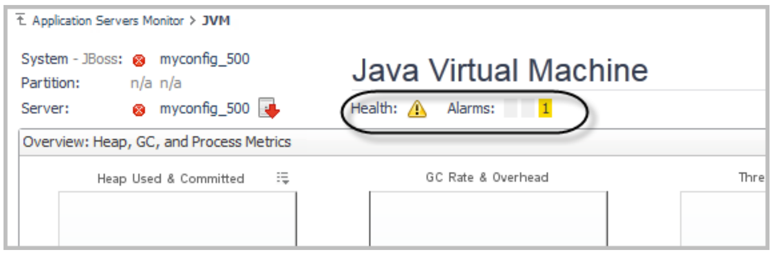
To investigate:

1 Click the health status icon in the JVM column. A pop-up opens.

Review the information in this pop-up. For example, check that the garbage collection rate and overhead shown in the GC Rate & Overhead chart are as expected.

2 Click the JVM name at the top of the pop-up. The Java Virtual Machine (JVM) view opens.

In the JVM view, you can see that the warning health status is due to a warning level alarm.



3 Click the health status icon for the JVM to review the alarm details.

For more information about the rules that generate alarms for the JVM, review the rules on the Foglight Rule Management dashboard (**Dashboards > Administration > Rules & Notifications > Rules >** search for “JVM”).

The JVM is just one example of a component for which there are drill-down views. Other examples include Web Applications, Web Services, EJBs, Requests, Services, Application Pools, and Sites. In all cases, you can access these drilldown views by clicking the name of an object or component in a health status pop-up, as described in the previous example.

For more information about individual views, see Application Servers Monitor Views on page 55.

3 Monitoring Servers

Use the Servers tile of the Application Servers Monitor dashboard to monitor specific application servers. From this view, you can investigate individual application servers in your internal and remote infrastructure and review the invoked operations that these application servers use.

To learn more about the supported application servers, see one of the following guides:

- *Foglight for Java EE Technologies Supported Platforms and Servers Guide*
- *Foglight for JMX Systems User Guide*
- *Foglight for Microsoft .NET Installation and Configuration Guide*

From this view, you can perform the following actions:

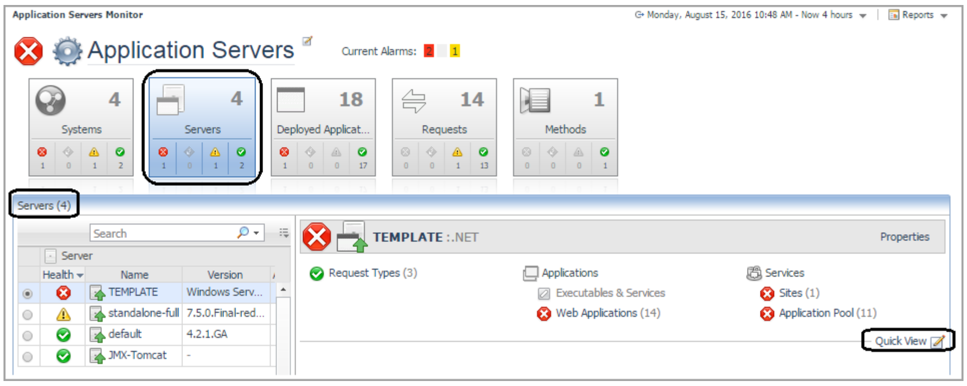
- Review the number of servers in your infrastructure
- Review the health of the servers
- Identify the application server type
- Review the server Availability (average for the timeline range specified)
- Review the server Outages
- Review JVM Heap Usage (average for the timeline range specified)
- Review JTA Rollback totals (Sum for the timeline range specified)

> **IMPORTANT:** JTA Rollback totals are available only for Foglight for Java EE Technologies.

- Review source metric charts in Quick View
- Collect Traces (from the Action panel > General tab)

To view the servers:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.
2. Click the **Servers** tile.



Servers list

The Servers table lists all the application servers running in your environment. Use this table to review the health and version of any monitored application server, as well as the availability (up time as a percentage), and any outages. Select an application server in this table to display detailed information in the Server Details view on the right.

NOTE: In addition to the default columns, there are several columns you can add to this table. For information about how to select columns, see Show/Hide table columns on page 9.

Server details

This view provides details about the server that you select in the Server table, including: Request Types, Method Groups, health of the JVM, Applications, any collected MBean components, EJBs, and Services.

NOTE: The details available in this view are relative to the type of server selected in the Servers table.

Each monitored component in the server details view has a health status icon, and each component name is a link. To view the health summary for the component, click the status icon. For more information, see Reviewing the health of your systems on page 13.

To view more details about a component, click the name link. A pop-up or drilldown view opens, providing more information. For example, click **JVM** to open the Java Virtual Machine (JVM) view.

TIP: Use the breadcrumb trail at the top of any drilldown view to return to the Application Servers Monitor dashboard.

Quick View

The Quick View option provides condensed charts that show the general state of a selected system. These charts answer questions such as: *What are the slowest three requests on server ABC*, or *what thread pools have the most waiters*? The Quick View Selection menu differs for application server type. The following table outlines the charts available for each application server.

Table 1. .NET application servers

Quick-View Chart Name	Description
Sites — Current Connections	Period average of current connections to the web site.
Web Applications — Request Rate	The number of requests run per second by the application.
Site — Request Throughput	The rate, in seconds, at which all HTTP requests have been received.
Web Application — Timed Out Requests	Count of web application requests that timed out.
Application Pool — Exception Count	Count of exceptions that occurred in the application pool.
Application Pool — % Processor Time	Percentage of processor time the application pool uses, as a period average.
Executable/Service — Exception Count	Count of exceptions that occurred in all standalone executables and Windows services applications monitored by the server.
Executable/Service — % Processor Time	Current processor time used by all monitored executable and Windows services applications.

Table 2. JBoss application servers

Quick-View Chart Name	Description
Slowest Response Times	Request with longest response times on average.
Longest Wait Times	Queues with the longest wait times on average.
Most Active Topics	Topics with largest incoming message rate on average.
Highest Pool Usage %	EJB pools with highest usage percentage on average.
Slowest Service Times	JSPs/Servlets with the lowest service times on average.

Table 3. OracleAS application servers

Quick-View Chart	Description
Slowest Response Times	Requests with longest response times on average.
Longest Queues	Thread pools with largest number of requests in their thread pools on average.
Longest Total Wait Times	JSBC connection pools with the longest total time all requests spent waiting for a connection.
Slowest Service Times	JSPs/Servlets with the longest request times on average.

Table 4. Tomcat application servers

Quick-View Chart	Description
Requests — Slowest Response Times (ms)	Slowest response time for each request traced.
JVM — Heap Usage	Capacity and Used memory.
JSP/Servlets — Error Count (c)	JSP/Servlets with the highest number of errors.
JSP/Servlets — Service Time (ms)	JSP/Servlets using the highest time on average.
Thread Pools — Highest Pool Usage	Thread pools which are being used most of the time.

Table 5. WebLogic application servers

Quick-View Chart Name	Description
Requests — Slowest Response Times (ms)	Request with the longest response time on average.
JVM — Heap Usage	Capacity and Used memory.
JVM — Garbage Collector	Garbage collector rate and overhead.
Work Managers — Most Waiters	Work managers with the highest number of waiters in their queues on average.
EJBs — Highest Cache Miss %	EJBs with the highest cache miss percentage on average.
EJBs — Highest Pool Miss %	EJBs with the highest pool miss percentage on average.
JSPs/Servlets — Most Invocations	JSPs/Servlets with the most invocations.
Execute Queues — Most Pending Requests	Run queues with the highest pending request count in their queues on average.
Local JMS Servers — Most Pending Messages	JMS servers with the highest pending message count on average.
JDBC Data Source — Most Connection Waiters	JDBC data sources with the highest number of threads waiting for a connection on average.

Table 6. WebSphere application servers

Quick-View Chart Name	Description
Requests — Slowest Response Time (ms)	Requests with longest response time on average.
JVM — Heap Usage	Capacity and Used memory.
Thread Pools — Highest Pool Usage %	Thread pools with highest utilization on average.
JDBC Data Source — Most Threads Waiting	JDBC data sources with highest number of threads waiting for a collection.
JSPs/Servlets — Most Popular	JSPs/Servlets accessed (requested) the most.
JCA Pools — Most Waiters	JCA Pools with highest number of threads waiting on average.
EJBs — Lowest Hit Ratio %	EJB pools with lowest hit ratios.

Adding charts to the Quick View

To add or remove charts in Quick View:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.
2. Click the **Servers** tile.
3. Click **Quick View**, or the edit icon beside the text.

   The Quick View Selection dialog box opens.
4. Select any of the source metrics listed in the dialog box.
5. Click **OK** to close the dialog box. The Quick View refreshes.

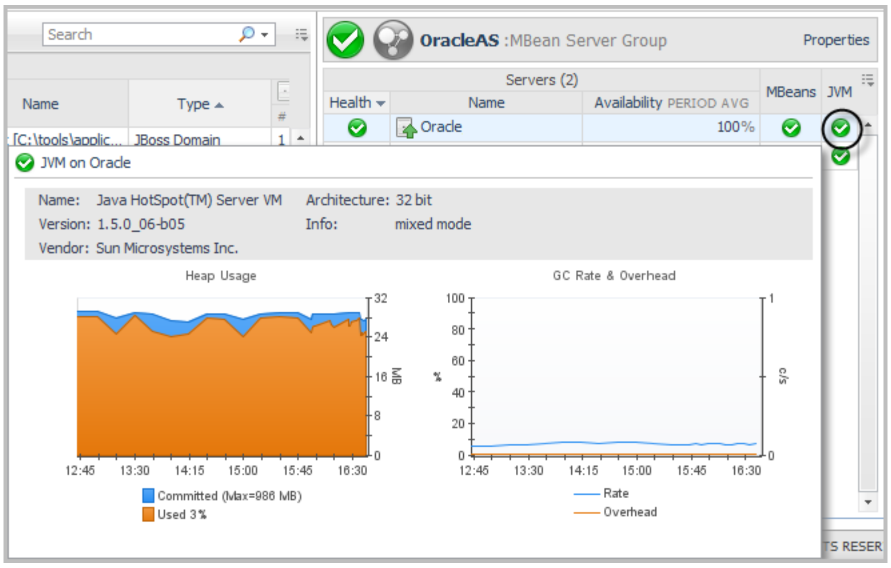
Viewing JVM data for systems or servers

You can access Java Virtual Machine (JVM) details for an MBean Server from either the Systems view or the Servers view of the Application Servers Monitor dashboard.

To access JVM data from the Systems view:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.
2. Click the **Systems** tile.
3. Select a system from the System table on the left.
4. In the system details view on the right, click the health status icon in the JVM column.

   A pop-up displays Heap Usage and GC Rate & Overhead charts, and basic JVM information.



Name: Java HotSpot(TM) Server VM
Version: 1.5.0_06-b05
Vendor: Sun Microsystems Inc.
Architecture: 32 bit
Info: mixed mode

5 Click the JVM Name or the Heap Usage chart to open the Java Virtual Machine (JVM) view.

> **TIP:** Use the breadcrumb trail at the top of the view to return to the Application Servers Monitor dashboard.

To access JVM data from the Servers tile:

1 On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.

2 Click the **Servers** tile.

3 Select a server in the Server table on the left.

4 In the server details view on the right, do one of the following:

- Hover your mouse over the JVM link in the server details area to open a dwell that displays the Heap Usage and GC Rate & Overhead charts. Click the **Heap Usage** chart in this dwell to open the JVM view.
- Click **JVM** in the server details view. The JVM view opens.
- Click **Quick View**, located just below the server details view, and select **JVM — Heap Usage** from the dialog box that opens. Click **OK**. The JVM Heap Usage chart appears in the Quick View area.

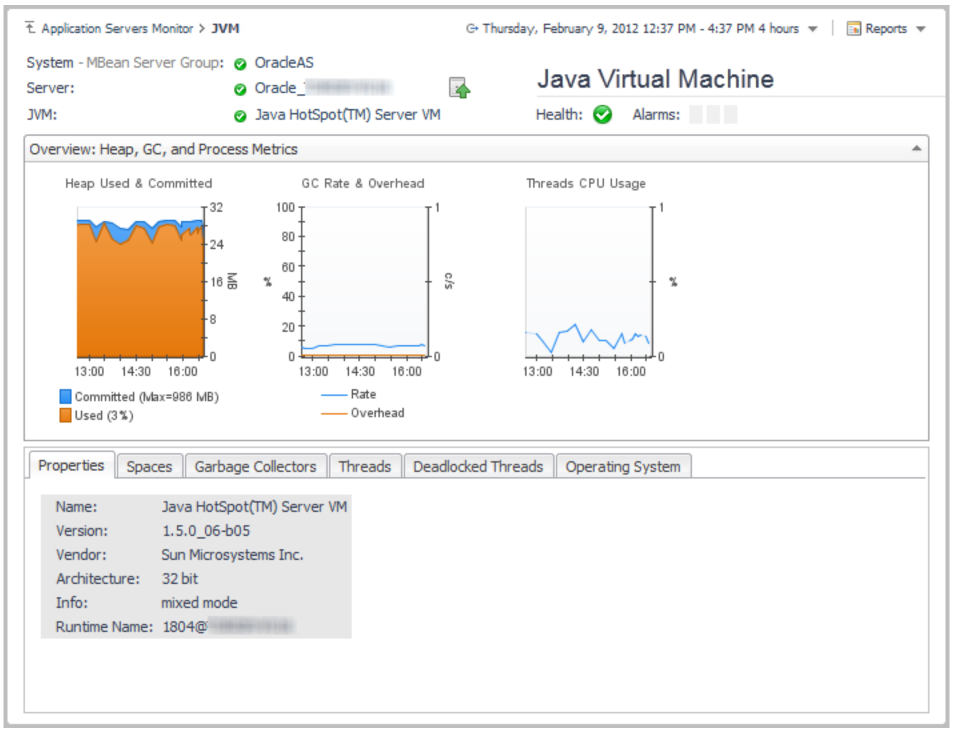
Java Virtual Machine (JVM) view

The top portion of this view lists the System, Server, and Health information. Click any status icon to review the health of that component. For more information about the rules that determine the Java Virtual Machine (JVM) health status, review the rules on the Foglight Rule Management dashboard (**Dashboards > Administration > Rules & Notifications > Rule Management** > search for “JVM”).

The charts in the middle section give you a quick view of the JVM Heap Used and Committed, Garbage Collection (GC), and Process Metrics, including:

- The amount of memory used and available
- The number of garbage collection occurrences per second (rate) and the amount of time spent performing garbage collections compared to the interval that Foglight is reporting (overhead)
- The amount of CPU the threads are using

Figure 9. Charts available in the JVM view.



Application Servers Monitor > JVM

Thursday, February 9, 2012 12:37 PM - 4:37 PM 4 hours

Reports

System - MBean Server Group: OracleAS

Server: Oracle_

JVM: Java HotSpot(TM) Server VM

Java Virtual Machine

Health: ✓ Alarms:

Overview: Heap, GC, and Process Metrics

Properties | Spaces | Garbage Collectors | Threads | Deadlocked Threads | Operating System

Name:	Java HotSpot(TM) Server VM
Version:	1.5.0_06-b05
Vendor:	Sun Microsystems Inc.
Architecture:	32 bit
Info:	mixed mode
Runtime Name:	1804@

In addition to viewing the heap usage of your JVM, it is important to understand how frequently garbage collection is run (GC Rate) and the percentage of time your JVM is spending on garbage collection (GC Overhead).

To see more details about the JVM, click one of the tabs in the JVM view below the charts. For more information, see JVM view on page 55.

Monitoring Deployed Applications

NOTE: Monitoring of Deployed Applications is not applicable to Foglight for JMX.

On the Application Servers Monitor dashboard, the **Deployed Applications** tile lists all the deployed applications running in a monitored environment (domain, cell, cluster topology, server group, or partition) and the related components, such as Application Pool, EJBs, Web Applications, and the Web Services health status. From here you can drill down to explore the components in further detail.

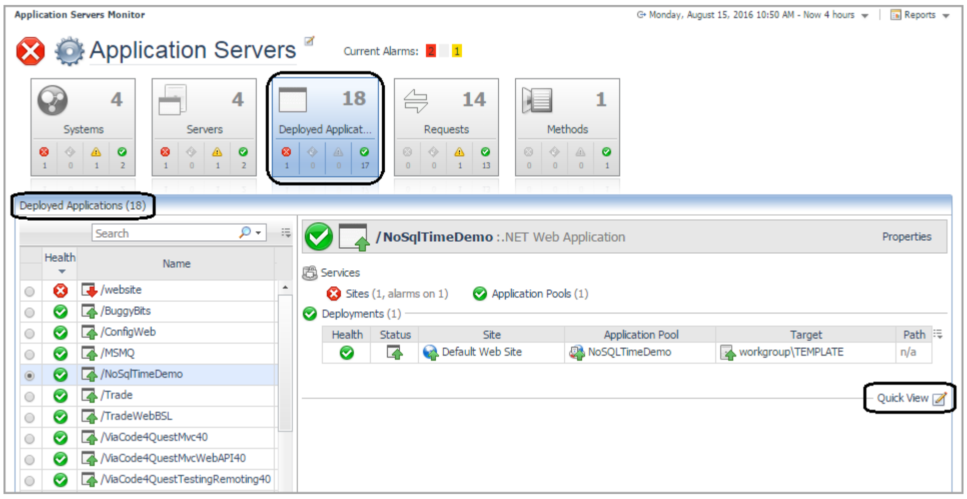
Use this view to investigate the deployed applications running in your infrastructure. Select an application in the table to display a list of the various types of components that are running in a specific application and their deployment details.

From this view, you can perform the following actions:

- Review the number of applications running in your infrastructure
- Identify the application type
- Identify the availability (average for the timeline range specified)
- Identify the application deployments (number of servers the application is deployed on)
- Select a health status icon for a component to display a dwell listing the various types of that component that are running on a specific application
- Drill down to explore the component type in further detail
- Review source metric charts in **Quick View**
- Collect Traces (from the **Action** panel > **General**)

To access the Deployed Applications view:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.
2. Click the **Deployed Applications** tile.



Health	Name
✗	/website
✓	/BuggyBits
✓	/ConfigWeb
✓	/MSMQ
✓	/NoSqlTimeDemo
✓	/Trade
✓	/TradeWebBSL
✓	/ViaCode4QuestMvc40
✓	/ViaCode4QuestMvcWebAPI40
✓	/ViaCode4QuestTestingRemoting40

Health	Status	Site	Application Pool	Target	Path
✓		Default Web Site	NoSQLTimeDemo	workgroup\TEMPLATE	n/a

Deployed Applications table

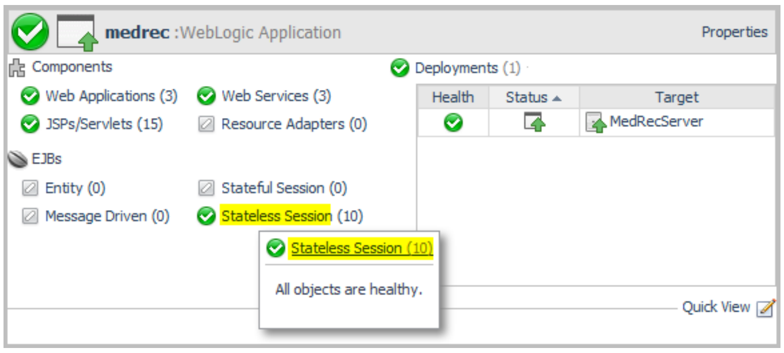
The Applications table lists all monitored applications in your environment. Click the health status icon to review the health summary for an application. By default, only the Health, Name, and Deployments columns are visible. To add the Type and System columns, see Show/Hide table columns on page 9.

Components and Deployment view

The Components and Deployment view lists the availability and health for the selected application, its components, EJBs, and deployments. Click any object link in this view to access a health summary for that object.

On some health summaries, you can drill down to more detailed views by clicking the name of an object.

Figure 10. For example, for a WebLogic Application that has Stateless Session Beans, drill down to the instances by clicking Stateless Session, and then clicking the name Stateless Session on the health summary pop-up.



TIP: Use the breadcrumb trail at the top of any drilldown view to return to the Application Servers Monitor dashboard.

Properties

You can also view the properties for the selected application by clicking the **Properties** link at the far right of the Components and Deployment view.

Table 7. The following server property information is displayed when available from the application:

Name	Description
Type	Displays the type of application.
Version	Displays the version number of the application. NOTE: n/a is displayed if the deployment descriptor element is not defined on the application server.
Is Active	An application server can have several versions of an application installed. A value of `true` indicates that the application is the active version (that is, the one that new sessions use).
Retire Time	Applications require upgrading to the latest version from time to time. This value indicates the retirement time of an application. (milliseconds)
Retire Timeout	The amount of time an older application is given to retire (seconds).
Deployment Order	The number of applications depending on this application server.

Quick View

The Quick View option provides condensed charts that show the general state of a selected system. These charts answer questions such as *what are the JSP/Servlets with the most invocations,* or *what is the average pool usage for an EJB instance.* The Quick View menu options available depend on the type of application server selected. The following table outlines the charts available for each application type.

Table 8. .NET applications:

Quick-View Chart	Description
Web Application — Most Errors During Execution	The number of errors that have occurred during the processing of a request.
Web Application — Most Timed Out Requests	The number of web application requests that timed out.
Web Application — Most Aborted Transactions	The number of web application transactions that were aborted.
Web Application — Most Requests	Total number of requests serviced since the application started.

Table 9. JBoss applications:

Quick-View Chart	Description
EJBs — Highest Pool Usage %	Average JBoss EJB Instance pool misses for a specified period.
JSP/Servlets — Slowest Service Times	JSP/Servlets with the longest request times on average.

Table 10. OracleAS applications:

Quick-View Chart	Description
JSP/Servlets — Slowest Service Times	JSP/Servlets with the longest request times on average.

Table 11. Tomcat applications

Quick-View Chart	Description
JSP/Servlets — Error Count	JSP/Servlets with the highest number of errors.
JSP/Servlets — Service Time (ms)	JSP/Servlets using the highest time on average.
Application — Cache Hit Ratio	Ratio of cache hits to cache misses.

Table 12. WebLogic applications

Quick-View Chart	Description
EJBs — Highest Pool Miss %	Average WebLogic EJB Instance pool miss percentage for a specified period.
JSP/Servlets — Most Popular	JSP/Servlets with the most invocations.

Table 13. WebSphere applications

Quick-View Chart	Description
EJBs — Lowest Hit Ratio	EJB pools with the lowest hit ratios.
JSP/Servlets — Most Popular	JSP/Servlets with the most invocations.

Adding charts to the Quick View

To add or remove charts in the Quick View:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.
2. Click the **Deployed Applications** tile.
3. Click **Quick View**, or the edit icon beside the text.

   The Quick View Selection dialog box opens.
4. Select any of the source metrics listed in the dialog box.
5. Click **OK** to close the dialog box.

   The Quick View refreshes.

5

Monitoring Requests

NOTE: Request monitoring is not applicable to Foglight for JMX.

The Application Servers Monitor dashboard includes predefined views for monitoring and investigating alarms associated with service requests made to your applications. Use the **Requests** view to investigate unique request types, such as requests made through HTTP, JMS, RMI, or JNDI, and to access full details of those request types. In addition, this interface allows you to investigate single traces in full detail or at component detail to assist you in localizing a potential trouble spot.

Service requests arrive through HTTP, or RMI and JMS calls. The requests initiate a chain of events, potentially invoking numerous EJB methods and database calls, before returning a result. The entire related chain of events is defined as a service request in Foglight. Request data appears in the Application Servers Monitor > Request view.

You can also monitor requests and generate reports on response time through this dashboard. An alarm fires when a request falls outside of its predefined conditions, and the alarm appears in the Application Servers Monitor dashboard > Requests view.

The **Requests** view shows performance information about a selected service request.

From this view, you can perform the following actions:

- Review all service requests coming from external clients, using HTTP, RMI or other communication protocols.
- Review Response Time/Call Count, Sampled Breakdowns, and Incomplete/Exceptional Exits charts.
- Collect Traces (from the Action panel > General tab)

Typical request issues that generate alarms

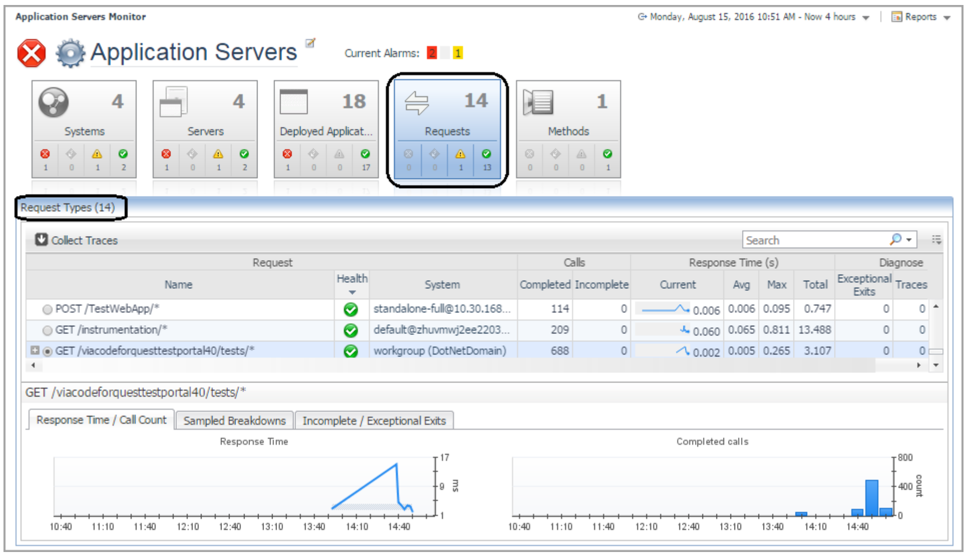
Typical problems that show up as alarms include:

- Excessive execution times
- Unacceptable number of exceptions
- Unacceptable number of incomplete invocations

To view requests:

1 On the navigation panel, under Dashboards, click **Application Servers > Monitor**. The Application Servers Monitor dashboard opens.

2 Click the **Request**s tile.



Request			Calls		Response Time (s)				Diagnose	
Name	Health	System	Completed	Incomplete	Current	Avg	Max	Total	Exceptional Exits	Traces
POST /TestWebApp/*	✓	standalone-full@10.30.168...	114	0	0.006	0.006	0.095	0.747	0	0
GET /instrumentation/*	✓	default@zhuvmwj2ee2203...	209	0	0.060	0.065	0.811	13.488	0	0
GET /viacodeforquesttestportal40/tests/*	✓	workgroup (DotNetDomain)	688	0	0.002	0.005	0.265	3.107	0	0

For more information, see:

- Requests table
- Response time baseline and threshold
- Chart area

Requests table

The Requests table lists all service requests coming from external clients, using HTTP, RMI or other communication protocols. For example, in the Medical Records sample application that ships with WebLogic® servers, the table displays HTTP Get methods for log in, log out, record requests, and administration activities.

NOTE: Some of the columns described are hidden by default. To show or hide columns, see Show/Hide table columns.

Table 14. For each service request, the Requests table displays the following information:

Data Item	Description
Name	Name of the service request.
Health	Health state of how the service requests are processing. Click the health icon to obtain alarm information.
System	System information including name and ID; and domain, cell, cluster topology, or partition type. Click the system name to display the Domain details.
Path	The context path provides the Service request method and resource name, mapping, and context path.
Completed Calls	The total number of complete calls to a request.
Incomplete Calls	The total number of incomplete calls to a request.
Incomplete %	The percentage of incomplete calls to a request.

Table 14. For each service request, the Requests table displays the following information:

Data Item	Description
Response Time (s) (Current, Average, Max, and Total (s))	Response Time baseline and threshold values are generated by IntelliProfile. For more information, see Response time baseline and threshold. • Current — a quick history bar showing Current Response Time without requiring deeper diagnostics, and value in seconds • Avg, Max, and Total — the average, maximum, and total response times for the request type, in seconds
Exceptional Exits	Number of requests that were incomplete and resulted in an exception.
Exceptional Exits %	Percentage of requests that were incomplete and resulted in an exception.
Traces	The number of single traces that have been captured. To view single trace details, click the value in the Traces column. For more information, see Managing Traces on page 33.
Actions	Opens a menu with two options: • Collect Traces — opens the Collect Traces dialog box, where you can configure single trace collection. For more information, see Managing Traces on page 33. • Generate Report — use to generate a single trace report.

To collect traces, click **Collect Traces**. The Collect Traces dialog box opens. Use it to configure single trace collection. For more information, see Managing Traces on page 33.

The request you select in the Request Types table controls what data appears in the Charts area. By default, the first item in the Request Types table is selected.

Response time baseline and threshold

The Foglight for Application Servers IntelliProfile engine generates baseline and threshold values for the Response Time of request types. IntelliProfile uses data collected during a desired time period and generates a baseline operating range based on the collected metrics. A baseline establishes expected data patterns for a given time period. Baselines are periodically evaluated during IntelliProfile learning cycles, to reflect changes in data patterns.

The baseline data for the Response Time appears in the Response Time chart, on the Response Time/Call Count tab.

i | **NOTE:** Data must be collected for at least 24 hours before the baseline is generated and displayed.

Threshold states (such as critical or fatal) reflect the degree of deviation from the baseline, and can indicate potential performance bottlenecks. Foglight for Application Servers can generate alarms when the Response Time enters a certain threshold state, based on the rule conditions that evaluate the threshold states.

i | **NOTE:** The Response Time threshold specifies only Critical and Warning bound values.

There are two rules related to Response Time for request types: **Request Response Time Baseline Rule** and **Request Response Time Threshold Rule**. These rules are disabled by default.

For more information about the rules, review the rules on the Foglight Rule Management dashboard (**Dashboards > Administration > Rules & Notifications > Rule Management >** search for “response time”).

For more information about IntelliProfile in Foglight for Application Servers, see Analyzing Activity Levels with IntelliProfile in the *Foglight Administration and Configuration Help*.

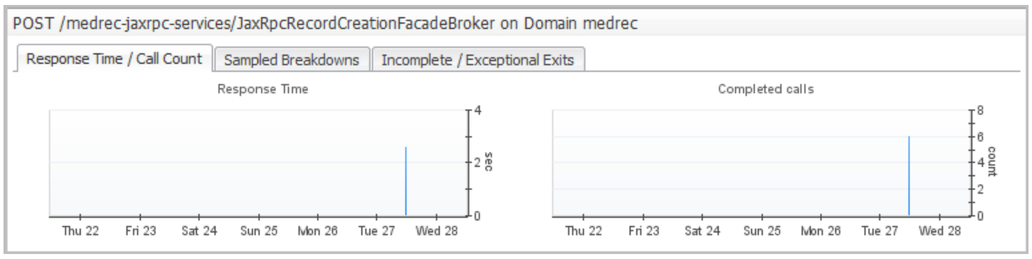
Chart area

The chart area of the Requests view contains graphs and tables of the execution-time data for the selected request over the selected time period. Use the tabs to access more information about Response Time and Call Counts, Sampled Breakdowns, and Incomplete and Exceptional Exits.

Response Time/Call Count tab

The graphs display the historical response time and call count of a request. For each selected request, the graph plots the average, maximum, and minimum values.

Figure 11. An example of the response time/call count tab.


POST /medrec-jaxrpc-services/JaxRpcRecordCreationFacadeBroker on Domain medrec
Response Time / Call Count
Sampled Breakdowns
Incomplete / Exceptional Exits
Response Time
Completed calls
sec
count
Thu 22
Fri 23
Sat 24
Sun 25
Mon 26
Tue 27
Wed 28

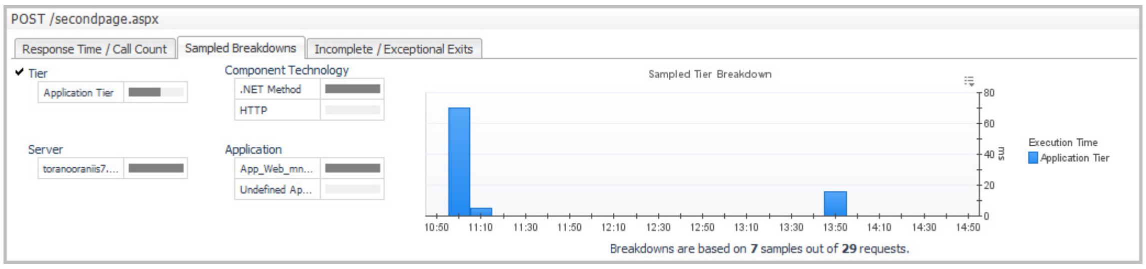
When baseline data for the Response Time is available, it appears in the Response Time chart. For more information about baselines, see Response time baseline and threshold on page 30.

Adjusting the timeline zonar updates the graphs with the data collected during that time period.

Sampled Breakdowns tab

There are four types of Sampled Breakdowns: Tier, Component Technology, Application, and Server. Click the name of the sampled breakdown type that you want to view to open its graph.

Table 15. An example of the sampled breakdown tab.



Breakdowns are based on 7 samples out of 29 requests.

The breakdown types:

- The **Sampled Breakdowns > Tier** graph shows the distribution of the response time over different tiers, such as the application servers and database. The results are displayed as a stacked graph, with a different colour representing each tier.
- The **Sampled Breakdowns > Component Technology** graph shows the execution-time data for the selected request by component technology. The response time is displayed in segments and each segment represents the time spent in each Java EE Component Technology (for example, HTTP, Servlet, EJB, JDBC, JNDI, or Custom Components). This information helps you accurately determine where to focus your application performance-tuning efforts. The results are displayed as a stacked graph, with a different color representing each component technology.
- The **Sampled Breakdown**s > **Application** graph is a stacked graph that shows the response time by the various applications that serviced the request.

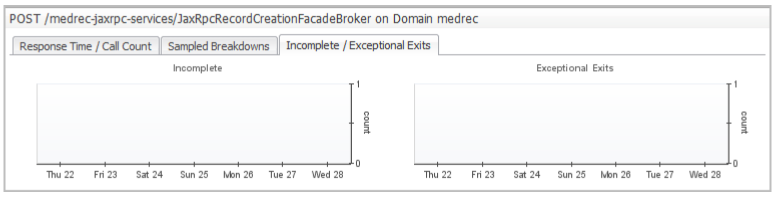
- The **Sampled Breakdowns > Server** graph is a stacked graph that shows the distribution of response time over different application servers.

These types of breakdowns are important because they show you if your application is spending more time in your application code versus your database. If a request passes between applications, the Sampled Breakdowns > Application graph shows you how much time is spent in each application. For example, if your application calls to another application for credit card validation, both applications are displayed in the Sampled Breakdowns > Application graph.

Incomplete/Exceptional Exits tab

The Incomplete Requests chart displays the count of incomplete service requests that timed out during the interval spanned by the zonar. Foglight considers a request incomplete when the request does not complete processing within the timeout specified in the *recording.config* property file.

Figure 12. An example of the incomplete/exceptional exits tab.


POST /medrec-jaxrpc-services/JaxRpcRecordCreationFacadeBroker on Domain medrec
Response Time / Call Count
Sampled Breakdowns
Incomplete / Exceptional Exits
Incomplete
Exceptional Exits
count
Thu 22
Fri 23
Sat 24
Sun 25
Mon 26
Tue 27
Wed 28

The Exceptional Exits chart displays the count of requests that completed with an uncaught exception during the interval spanned by the zonar.

6

Managing Traces

With Foglight for Java EE Technologies and Foglight for Microsoft .NET, you can collect traces on requests. If you have enabled trace collection, you can review Single and Aggregated Traces, as well as Sampled Breakdowns and Overall Request Performance from the Traces view.

There are three main methods for collecting traces:

- Manually collecting single traces
- Using a rule action to collect single traces
- Using Groovy to collect single traces.

Manually collecting single traces

Single traces can be collected manually from the Application Servers Monitor > Requests > Requests table, or from the Actions panel when any of the Application Servers Monitor dashboard views are selected.

NOTE: Collecting single traces using the interface is limited to the set time range. You can collect all single traces by updating the Request Response Time Threshold rule, but it is not recommended because collecting all single traces affects performance. Instead, change the time range. For more information, see Using a rule action to collect single traces on page 34.

IMPORTANT: Users assigned the “Java EE Read-only User” or the “.NET Read-only User” role have limited access to the Application Servers Monitor dashboard. They cannot initiate single trace collection but all other information is available to them. If tracking single traces is required, update the user’s role to “Java EE Administrator” or “.NET Administrator”.For more information on groups, roles, and users, see the *Foglight Administration and Configuration Guide*.

To collect single traces:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**.
2. On the Application Servers Monitor dashboard, click the **Requests** tile.
3. In the Request Types table, click **Collect Traces.** The Collect Traces dialog box opens.
4. If you want to specify the Request Type using regular expressions, select the **Use regular expression** check box and type a string in the box below.

   For more information about writing regular expressions, see Appendix: Regular Expressions on page 80.
5. Select the number of traces that you want to collect in the **Number of Traces** list.
6. Set the time period for which you want to collect the number of traces in the **Collection Period** box. The default value is 6000 seconds.
7. Select whether you want to collect methods in full detail or component detail using the **Collection Level** list. The default value is component detail.
   - Full detail — collects single traces at full detail; all methods in the call tree are captured
   - Component detail — collects single traces at component level only; methods from a defined component technology, including custom components, are captured

IMPORTANT: If the application has been instrumented at full detail, then the full detail collection level can be used to collect everything. If the application was instrumented at component level (or less), selecting full detail has no impact. For more information, see the Managing Java EE Agent Configurations topic in the *Foglight for Java EE Technologies Installation Guide*.

8 Use the **Collect SQL Parameter** list to indicate whether you want to track SQL bind variable values. The default value is Off. For more information, see SQL tab on page 43.

9 Click the check box beside **Traces that match** to limit the trace collection to **All** or **Any**.

10 If you select **Any**, click the add icon to open the **Add criteria** menu.

Select a criteria from the list.

- Response Time — the response time of the request is over or under a specified threshold
- Exceptional Exit — the request exits due to an exception
- Incomplete — the request is incomplete
- Trace Parameter — a parameter of the request is set to a specific value

TIP: Specifying criteria to collect particular traces helps you diagnose problems faster.

11 **Optional** — Click the **Collect all parameters** check box to collect the values for all parameters. This option is enabled by default.

12 From the **Servers** list, select the servers from which you want to collect single traces.

13 Click **Collect Traces**.

The number of manual traces appears in the Traces column of the Request Types table.

Using a rule action to collect single traces

As an alternative to manually collecting single traces using the interface, you can add a RequestTracesAction to any rule to collect single traces to assist during issue diagnostics.

NOTE: This action was called JavaEETracesAction in previous versions.

For example, a request type is experiencing high response times and you have a rule that sends alarms when it reaches the threshold. You can specify the **RequestTracesAction** to collect single traces on a specific request type to monitor the issue.

To collect single traces using a rule action:

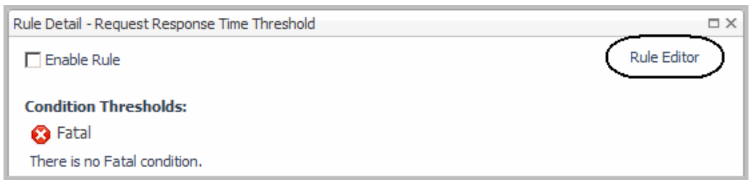
1 On the navigation panel, under Dashboards, click **Administration > Rules & Notifications > Rule Management**.

2 Click the name of a rule in the Rule Management table.

TIP: To shorten the list of rules, select the cartridge that contains the rule from the Cartridge list (for example, ApplicationServers-Java), or if you know the name of the rule, use the Search box.

3 From the menu that opens, select **View and Edit**.

4 In the Rule Detail dialog box that opens, click **Rule Editor**.


Rule Detail - Request Response Time Threshold

Enable Rule

Rule Editor

Condition Thresholds:

Fatal

There is no Fatal condition.

The Edit Rule Page opens.

5 Click the **Conditions, Alarms & Actions** tab.

6 Select a severity level (for example, Warning) and click the **Action** tab.

7 From the **Action** menu, select **RequestTracesAction**.

> **IMPORTANT:** This action was called JavaEETracesAction in previous versions.

8 Click **Add** to add the RequestTracesAction.

The default action parameters are listed, as follows:

- **Request Type** — Use this action parameter to specify unique request types.

  > **NOTE:** The Request Type action parameter is required; all others are optional.

- **Collect Parameters** — Use this action parameter to collect all parameters.
- **Collect in full detail** — Use this action parameter to indicate whether you want to collect methods in full detail or component detail (the default).
- **Count** — Use this action parameter to specify the number of traces being collected.
- **Criteria** — Use this action parameter to specify one or more criteria.

  Create an expression type Severity Level Variable to specify the criteria. Use any of the arguments listed in the AppServerCollectTraces function table to define the criteria string. For example:

  ```
  def criteria = [];
  criteria.add("responseTimeOver:10000"); // > 10 seconds
  //criteria.add("exceptional");
  //criteria.add("incomplete");
  return criteria;
  // comment: response times are in milliseconds
  ```

- **Duration** — Use this action parameter to indicate the time period to collect the number of traces.
- **Is Regular Expression** — Use this action parameter if you want to specify the Request Type using regular expressions to match the request type for a single trace collection.
  - **Match All criteria** — Use this action parameter to limit the trace collection to all or any of the following criteria:
    - The response time of the request is over a specified threshold.
    - The response time of the request is under a specified threshold.
    - The request exits due to an exception.
    - The request is incomplete, or a parameter of the request is set to a specific value.
- **The agent name IDs** — Use this action parameter to specify the agentNameIDs property of the agent that you want to collect traces on.

  Create an expression type Severity Level Variable to specify the agent. For example, to use a variable to specify an agent in Groovy:

  ```
  def agentNameID = [];
  ```

```
agentNameID.add(scope.agent.agentNameID);

return agentNameID;
```

- **Track SQL Parameters** — Use this action parameter to track SQL bind variable values. The default value is set to Off.

9 In the Value column for Request Type, click **Default** and specify the value.

Click **Change** to update the action parameter.

10 Once the action parameters have been updated, click **Save Al**l.

For more information about rules, see Getting started: view and edit rule definitions in the *Foglight Administration and Configuration Help*.

Using Groovy to collect single traces

You can also use a Groovy script to collect single traces. In Groovy, request types are specified by a regular expression, so the same set of single trace criteria can be used for a group of request types.

When you have a group of request types for which you want to collect single traces using the same criteria, set the criteria using a Groovy script called from the command line (for example, using *fglcmd*). The collected single traces are stored on the Foglight Management Server, and displayed in the Application Servers Monitor > Requests view for users to review. Use the following function to trigger the collection of requested traces.

AppServerCollectTraces function

NOTE: This function was called JEECollectTraces in previous versions. Any existing Groovy scripts using JEECollectTraces continue to work. You do not need to update them.

This function is used to collect traces in Groovy. Set all the following arguments:

Table 16. Required arguments for the AppServerCollectTraces Groovy function.

Argument	Description
`requestType: String`	Request type to collect trace.
`isRegEx: Boolean`	Request type for which to collect a trace defined as regular expression (true/false).
`tracePeriod: Long`	Duration (in seconds) of the request to trace.
`count:Integer`	The number of traces to collect.
`CollectInFullDetail:Boolean`	Collect in Full Detail (true/false). For details about instrumentation levels, see Appendix: Application Methods in the *Foglight for Java EE Technologies Installation Guide*.
`trackSQLParams:Boolean`	Track SQL Parameters (true/false).
`matchALL:Boolean`	Match all the criteria (true/false).
`criteria:List`	A list of criteria. See the next table for details.
`CollectParameters:Boolean`	Collect all trace parameters (true/false).
`agents:List`	List of agents to collect traces on. If the list is empty or null, then traces are collected on all agents.

Criteria are specified as a list of one or more strings.

Table 17. Criteria for the AppServerCollectTraces Groovy function

Format	Example
`all`	all
`none`	none
`responseTimeOver:t`	ReponseTimeOver:60000[1]
`responseTimeUnder:t`	ResponseTimeUnder:1000[1]
`exceptional`	exceptional
`incomplete`	incomplete
`traceParameterContains:k:v`	traceParameterContains:name:Bob
`traceParameterDoesNotContain:k:v`	traceParameterDoesNotContain:name:Tom
`traceParameterIs:k:v`	traceParameterIs:user:Bob
`traceParameterIsNot:k:v`	traceParameterIsNot:user:Tom
`traceParameterStartsWith:k:v`	traceParameterStartsWith:accountnumber:0
`traceParameterEndsWith:k:v`	traceParameterEndsWith:accountnumber:9

[1] Criteria are in milliseconds.

For example:

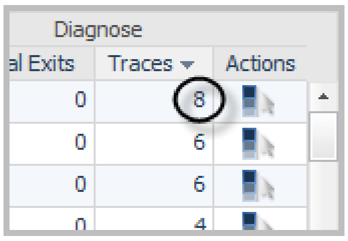
```
// Collect 5 traces in full detail for request "GET /samples/test.do" for 15 minutes
// only if response time is over 1 minute and user is Bob for the agent
def criteria = [];
criteria.add("traceParameterIs:user:Bob");
criteria.add("responseTimeOver:60000");
def agentNameIDs = [];
agentNameIDs.add(appServer.monitoringAgent.agentNameID);
// appServer is the server topology object
AppServerCollectTraces("GET /samples/test.do", false, 900, 5, false, true, true,
criteria, true, agentNameIDs);
```

Viewing traces

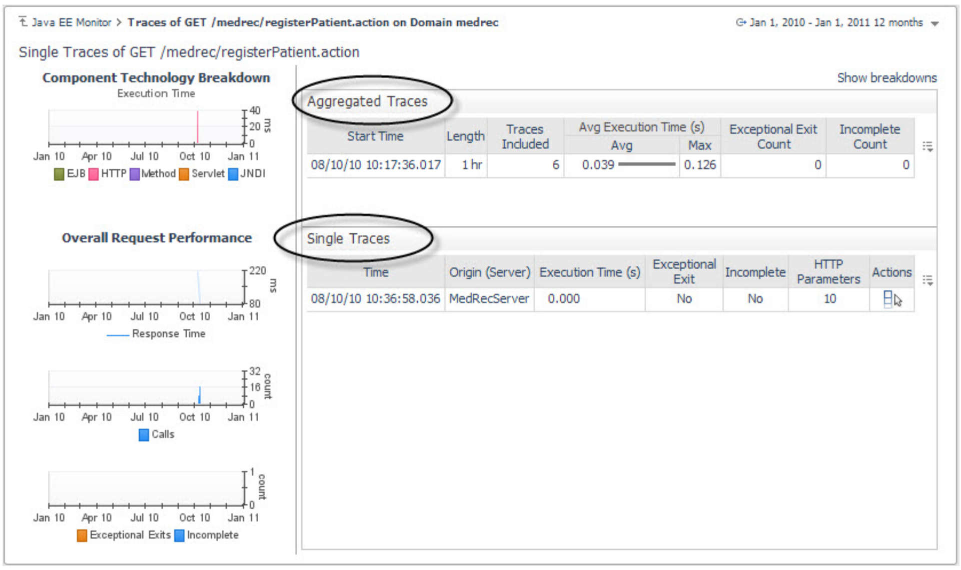
After you have configured trace collection, you can review the trace details through the Traces view.

To view single trace details:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**.
2. On the Application Servers Monitor dashboard, click the **Requests** tile.
3. In the **Traces** column of the **Request Types** table, click the number of traces for the request.



The Traces view opens, displaying both Aggregated Traces and Single Traces.


Java EE Monitor > Traces of GET /medrec/registerPatient.action on Domain medrec
Jan 1, 2010 - Jan 1, 2011 12 months
Single Traces of GET /medrec/registerPatient.action
Component Technology Breakdown
Execution Time
EJB HTTP Method Servlet JNDI
Overall Request Performance
Response Time
Calls
Exceptional Exits Incomplete
Show breakdowns

Aggregated Traces

Start Time	Length	Traces Included	Avg Execution Time (s)		Exceptional Exit Count	Incomplete Count
			Avg	Max		
08/10/10 10:17:36.017	1 hr	6	0.039	0.126	0	0

Single Traces

Time	Origin (Server)	Execution Time (s)	Exceptional Exit	Incomplete	HTTP Parameters	Actions
08/10/10 10:36:58.036	MedRecServer	0.000	No	No	10	

4 **Optional —** To view aggregated trace details, click a server name in the **Aggregated Traces** table, or to view single trace details, click a server name in the **Single Traces** table. The **Trace Diagnosis** view opens.

Traces view

Use the Traces view to review individual traces of a selected request. This view is divided into four sections: Aggregated Traces table, Single Traces table, Component Technology Breakdown, and Performance charts.

From the Traces view, you can set a breakdown of execution time by selecting Tiers and Component Technologies. By setting any of these options, a selected single trace or an aggregate trace can be used as a comparison benchmark for all the traced metrics. Once specified, both the list view and the tree view display the comparison in the Average Execution Time and Average Exclusive Time columns, and the rest of the metrics comparisons are shown at the bottom of the page, in the method details area.

To learn more about how to set a breakdown, see Viewing breakdowns on page 40.

Aggregated Traces table

The Aggregated Traces table, on the top right side of the view, lists a series of aggregate traces over the time period for the selected requests. Aggregate traces are created using the aggregate time interval, which is set to one (1) hour by default. They contain information about all the traces collected within the aggregate time interval, and are useful for investigating a group of single traces all at the same time. Aggregate traces are identified by the start time of the aggregation. There is only one aggregate per interval.

Table 18. The Aggregated Traces table contains the following columns:

Column Name	Description
Time	Start time of the aggregation.
End Time	End time of the aggregation.
Length	Total time of all requests within the aggregation to complete.
Traces Included	Total number of single traces included in the aggregate trace.
Execution Time (s) Avg, Max, and Total	Average, maximum, and total time taken by the request to complete.
Exceptional Exit Count	Number of requests which exited with an exception.
Incomplete Count	Number of requests which have timed out.

Table 18. The Aggregated Traces table contains the following columns:

Column Name	Description
Avg Execution Time by Component Technology (s)	Average execution time spent using the specified technology. For more information, see Viewing breakdowns on page 40.
Avg Execution Time by Tier (s)	Average execution time spent in the specified tier, in seconds. For more information, see Viewing breakdowns on page 40.

NOTE: Some of the columns described are hidden by default. To add columns to the table, see Show/Hide table columns on page 9.

Single Traces table

The bottom right side of the Traces view contains data related to the single trace selected in the Traces table. This table is the list of individual single traces. Clicking a time listed in this table displays the Single Trace view.

Table 19. The Single Traces table contains the following columns:.

Column Name	Description
Time	Start time.
Origin (Server)	Where the single trace started collecting.
Request URL	The URL of the traced request, truncated according to the URL transformation rules that are in use (default or custom). For more information about URL transformations, see the topic Setting Rules for Transforming URLs in the *Foglight for Application Servers Administration and Configuration Guide* or help.
Execution Time (s)	The time the trace took to run.
Response Time (s)	The amount of time, in seconds, that the trace took to complete.
Exceptional Exit	Whether this trace exited with an exception.
Incomplete	Whether this trace timed out.
HTTP Parameters	HTTP parameters are defined in the `recording.config` property file. If the single trace is using the specified parameters, information is displayed in the table. Click the HTTP parameter value to view more details in the FxV interface. For more information, see Viewing Request Data in FxV in the *Foglight for Application Servers Administration and Configuration Guide*.
Avg Execution Time by Component Technology (s)	Average execution time spent using the specified technology. For more information, see Viewing breakdowns on page 40.
Avg Execution Time by Tier (s)	Average execution time spent in the specified tier. For more information, see Viewing breakdowns on page 40.
Related Alarms	The number of alarms related to this single trace.
Actions	Shows or hide s the Actions icon.

NOTE: Some of the columns described are hidden by default. To add columns to the table, see Show/Hide table columns on page 9.

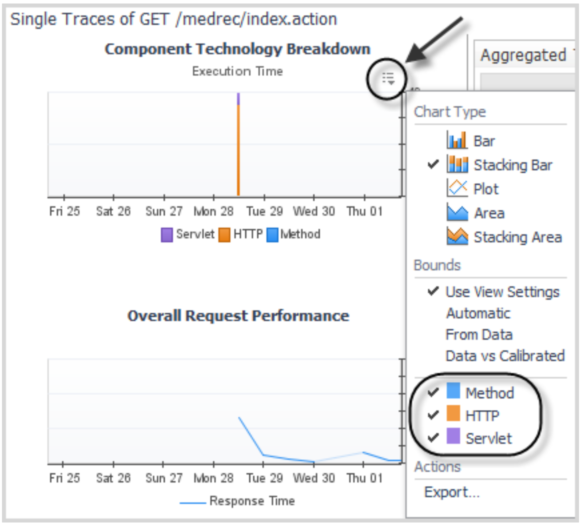
Performance charts

The Charts area contains graphs of the execution time and performance data for the selected single trace over the selected time period.

Component Technology Breakdown graph

The graph displays the Execution Time spent in each component technology (for example, Servlet, RMI Outgoing, HTTP, .NET Method, or ADO) for the aggregated traces.

Figure 13. To add or remove component technologies from the graph, click the customizer icon and select the technologies you want to see.



Overall Request Performance charts

There are three types of breakdowns:

- Response Time — the amount of time, in milliseconds, that the request took to complete
- Calls — the number of calls performed in the selected time period
- Exceptional Exits and Incomplete — the number of incomplete requests and requests that exited due to an exception during the selected time period

Viewing breakdowns

Breaking down the execution time by component technology and/or tier allows you to visualize the relative performance and breakdowns of all traces in a set. You can show the amount of time spent in each component of the application server for every trace in a set (aligned by time), and which servers each trace crossed. This is helpful when you have defined custom components to track a set of packages and classes.

Selecting any number of these custom components and tiers displays the average execution time in the Aggregated Traces table and the total execution time for a component technology and tier in the Single Traces table. You can also see breakdowns in chart form for a given request type.

NOTE: The Requests view displays charts with similar breakdowns. For more information, see Sampled Breakdowns tab on page 31.

To show breakdowns:

1. In the Traces view, click the customizer icon beside the table you want to modify (either Aggregated Traces or Single Traces). The **Show columns** menu opens.
2. Select the component technologies and tiers to display by clicking the check boxes.
3. Click **Apply**.

   The specified breakdown columns are added to the Aggregated Traces or Single Traces table.

Trace Diagnosis view

You can view the full details for an aggregated or single trace in the Trace Diagnosis view.

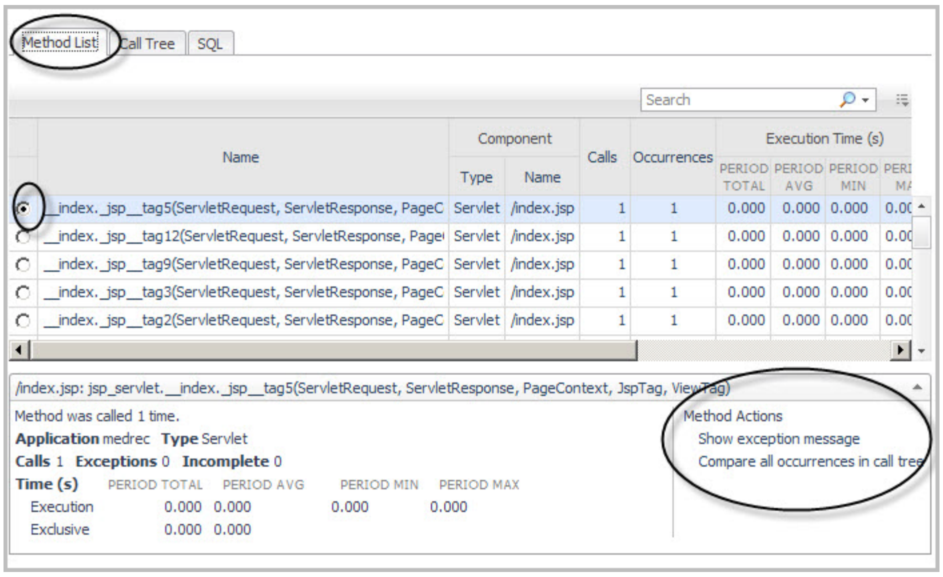
To access this view, select a trace by clicking the value in the **Time** column of either the Aggregated Traces table or the Single Traces table.

Method List tab

The Method List tab lists all the methods of the selected request trace in a flat mode. Methods that occurred in multiple places in the request tree are shown once in the list with their metrics aggregated.

Hover over an item in a row to display metric information or click to open a pop-up. In addition, if the method has exited with an exception, the pop-up displays the exception messages identifying the cause of the problem.

Figure 14. To investigate a single trace, select the option button in the first column of the Methods List tab.



	Name	Component		Calls	Occurrences	Execution Time (s)			
		Type	Name			PERIOD TOTAL	PERIOD AVG	PERIOD MIN	PERI MA
●	__index._jsp__tag5(ServletRequest, ServletResponse, PageC	Servlet	/index.jsp	1	1	0.000	0.000	0.000	0.00
○	__index._jsp__tag12(ServletRequest, ServletResponse, Page	Servlet	/index.jsp	1	1	0.000	0.000	0.000	0.00
○	__index._jsp__tag9(ServletRequest, ServletResponse, PageC	Servlet	/index.jsp	1	1	0.000	0.000	0.000	0.00
○	__index._jsp__tag3(ServletRequest, ServletResponse, PageC	Servlet	/index.jsp	1	1	0.000	0.000	0.000	0.00
○	__index._jsp__tag2(ServletRequest, ServletResponse, PageC	Servlet	/index.jsp	1	1	0.000	0.000	0.000	0.00

/index.jsp: jsp_servlet.__index._jsp__tag5(ServletRequest, ServletResponse, PageContext, JspTag, ViewTag)

Method was called 1 time.
Application medrec **Type** Servlet
Calls 1 **Exceptions** 0 **Incomplete** 0

Time (s)	PERIOD TOTAL	PERIOD AVG	PERIOD MIN	PERIOD MAX
Execution	0.000	0.000	0.000	0.000
Exclusive	0.000	0.000		

Method Actions
Show exception message
Compare all occurrences in call tree

The table below the list populates with data for the selected trace. Two method actions are available:

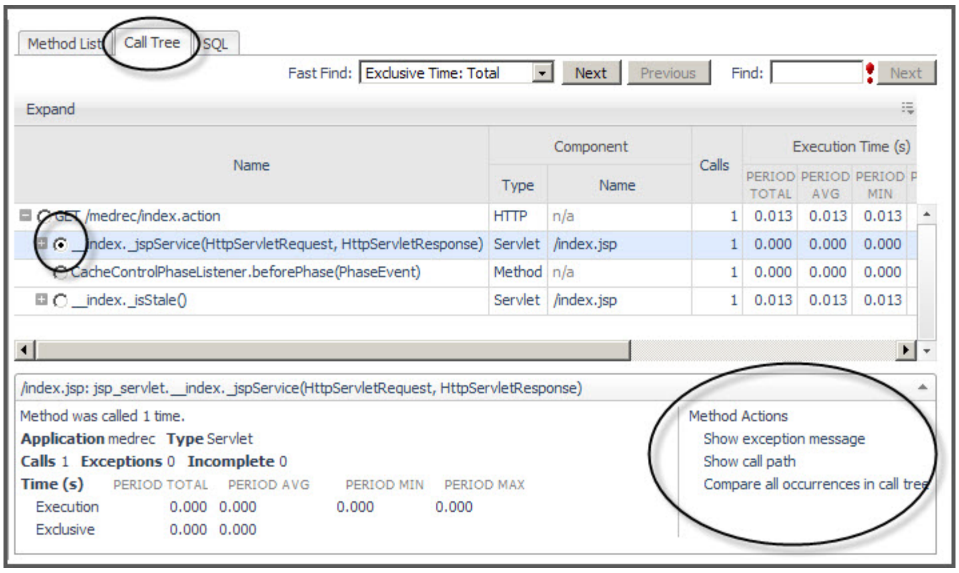
- **Show exception message** — Use this action when you want to view a list of detailed exception messages for a selected method.
- **Compare all occurrences in call tree** — Use this action when you want to view method metrics along with a unique call path for a selected method.

Using the Method List table, you can compare a single trace to another single trace or to an aggregate trace to use as a comparison baseline for all the trace metrics. For more information, see Selecting a trace to baseline or compare metrics on page 43.

Call Tree tab

The Trace Diagnosis Call Tree tab preserves the request execution path by showing the trace in a tree form.

Figure 15. Example of the Call Tree tab.



Name	Component Type	Component Name	Calls	Execution Time (s) PERIOD TOTAL	PERIOD AVG	PERIOD MIN
GET /medrec/index.action	HTTP	n/a	1	0.013	0.013	0.013
__index._jspService(HttpServletRequest, HttpServletResponse)	Servlet	/index.jsp	1	0.000	0.000	0.000
CacheControlPhaseListener.beforePhase(PhaseEvent)	Method	n/a	1	0.000	0.000	0.000
__index._isStale()	Servlet	/index.jsp	1	0.013	0.013	0.013

/index.jsp: jsp_servlet.__index._jspService(HttpServletRequest, HttpServletResponse)

Method was called 1 time.
Application medrec **Type** Servlet
Calls 1 **Exceptions** 0 **Incomplete** 0

Time (s)	PERIOD TOTAL	PERIOD AVG	PERIOD MIN	PERIOD MAX
Execution	0.000	0.000	0.000	0.000
Exclusive	0.000	0.000		

Method Actions
Show exception message
Show call path
Compare all occurrences in call tree

Select a single trace to investigate by selecting the option button in the first column of the Call Tree tab. The table below then populates with data for the selected trace and offers you two method actions:

- **Show exception message** — Use this action when you want to view a list of detailed exception messages for a selected method.
- **Show call path** — Use this action when you want to view the call path for a selected method in a numbered list.
- **Compare all occurrences in call tree** — Use this action when you want to view method metrics along with a unique call path for a selected method.

Expand

The entire tree and sub-trees below any node can be viewed by selecting the node, and then clicking **Expand** in the top left corner of the tree table.

Fast find

This functionality finds the most expensive node in the tree based on a selected metric. That row is highlighted in the table, and the tree is expanded up to the selected row. All the metrics in the Request Trace table are listed in the **Fast Find** drop-down menu and can be used as Fast Find criterion.

Click **Next** and **Previous** to move through fast find results.

Find text

Use the **Find** text box to find a particular method by typing part or all a method name. The first occurrence that matches the text is highlighted in the table. Click **Next** to move to the next occurrence.

Selecting a trace to baseline or compare

From the Call Tree tab, you can compare a single trace to another single trace or to an aggregate trace to use as a comparison baseline for all the trace metrics. For more information, see Selecting a trace to baseline or compare metrics on page 43.

SQL tab

The SQL tab displays SQL statements of the selected request trace by grouping the methods according to the statements they called.

Table 20. The SQL table has the following columns:

Column Name	Description
Statement	SQL Statement.
Calls	The number of times the method was called.
Execution Time (s) TOTAL	Total time spent running the SQL statement across all methods, or total time each method spent to run the statement.
Execution Time (s) AVG	Average time spent running the SQL statement across all methods, or average time each method spent to run the statement.
Execution Time (s) MAX	Maximum time spent running the SQL statement across all methods, or maximum time each method spent to run the statement.

Selecting a trace to baseline or compare metrics

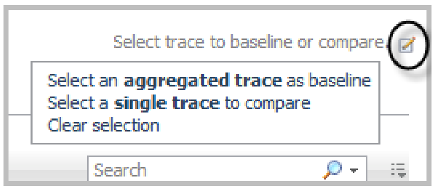
The **Select trace to baseline or compare** feature can display the relative performance of a single trace as compared to another single or aggregate trace. Selecting a baseline trace adds columns and fields that show the difference (DIFF) between the currently viewed single trace and the selected baseline trace.

To select an aggregated trace as baseline:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**.
2. On the Application Servers Monitor dashboard, click the **Requests** tile.
3. On the Requests view, click the value in the Traces column for the request you want to investigate.

   The Traces View opens.
4. In the Traces view, click the name of a single trace.

   The Trace Diagnosis view opens.
5. Select either the Method List tab or the Call Tree tab.
6. Select a method.
7. Click the edit icon beside the text **Select trace to baseline or compare**.

   
   Select trace to baseline or compare
Select an aggregated trace as baseline
Select a single trace to compare
Clear selection
Search
8. In the menu that opens, click **Select an aggregated trace as a baseline**.
9. In the **Select Trace for Comparison Dialog — Aggregated** dialog box, click an aggregated trace to use as a baseline for comparison.

   The view updates with the message: *“Comparing to trace at: <time stamp>”*.

The Method List and Call Tree tables now include a DIFF column, which compares the Total Execution Time and the Average Exclusive Time. A DIFF column also appears for the Execution and Exclusion times in the Exceptions and Incomplete table, on the bottom right of the view.

To set a single trace to compare metrics against:

1. Perform step 4 through step 7 in the previous procedure.
2. Click **Select a single trace to compare**.
3. In the **Select Trace for Comparison Dialog — Single** dialog box, click a single trace to compare.

   The view updates with the message: “Comparing to trace at: <*time stamp*>”.

   The Method List and Call Tree tables now include a DIFF column, which compares the Total Execution Time and the Average Exclusive Time. A DIFF column also appears for the Execution and Exclusion times in the Exceptions and Incomplete table, on the bottom right of the view.

To clear the selection:

1. Click the Edit icon.
2. Click **Clear selection**.

7

Using Object Tracking to Locate Memory Leaks

Foglight for Java EE Technologies includes the ability to examine monitored class breakdowns of tracked objects for each application server that participated in a request. You can use the breakdowns to investigate why memory usage problems are occurring for a particular application server.

Over the runtime of an application, objects that are allocated memory in the Java Virtual Machine (JVM) heap become eligible for garbage collection once all references to them have been released. If object references are not released, the objects loiter in memory taking up space that could be reused.

To examine these memory leak problems, you can enable object tracking and review the root of the problem.

NOTE: Object tracking is off by default. Enabling object tracking may cause added overhead on the performance of your application server. Also note, increasing the LiveObjectLifeSpan agent property value increases memory requirements of the agent. For more information about this agent property, see Updating object tracking agent properties on page 46.

Typically several application servers can be used to service a request and this request can cross JVM boundaries. On each JVM, memory is allocated for each application running. When monitoring your JVMs, you may notice a change in the memory performance for all your JVMs. In this situation, you can investigate the memory leak on one JVM first and then narrow down the investigation to find the application causing the leak. Once you have the memory leak source identified, you can then change the code to improve memory allocation for that application to resolve the leak.

Understanding when memory usage problems occur

You may observe a generally and consistently upward trend in the memory usage of a monitored JVM. For example, this occurs when object instances are added to a persistent, pinned collection but due to a coding error they are never removed. In this case, use object tracking to detect the object instances that are allocated but not collected.

In some situations, although memory usage appears normal, an excessive number of garbage collections may occur. The Foglight Management Server runs garbage collection rules to check on the time spent on garbage collection and if it exceeds a pre-defined threshold, alarms are generated. Excessive garbage collection may happen when allocating unnecessary or redundant object instances in a frequently used code path. While not a leak, this can still be a performance problem due to the garbage collection algorithm using more CPU time than would otherwise be necessary from the applications. In this case, object tracking can detect the most frequently allocated classes, regardless of whether they are collected.

The next step in both of these cases is to look more closely at object allocations in the affected JVM.

Workflow for enabling object tracking to detect general memory problems:

1 Updating object tracking agent properties on page 46

2 Enabling object tracking on page 48

3 Examining object tracking data on page 49

Updating object tracking agent properties

Object tracking applies to Foglight for Java EE Technologies only. To use object tracking, specify the packages and classes that you want to monitor in the agent configuration property file. The intention is that object tracking configuration should be changed infrequently. For example, you may change the configuration to include the packages of newly deployed applications that you want to monitor using object tracking.

By default, object tracking is configured such that the Java utility collection classes can be tracked. This may be sufficient for an initial investigation, however, to track objects allocated in your applications, add or replace the default configuration with your own packages.

Configuring object tracking does not enable object tracking; it only specifies what is tracked when you do enable object tracking.

NOTE: Restart your application servers after changing the object tracking classes in the configuration in order for the changes to take effect.

To update object tracking agent properties:

1 On the navigation panel, under Dashboards, click **Application Servers > Administration**.

2 On the Application Servers Administration dashboard, click the **Java tab**.

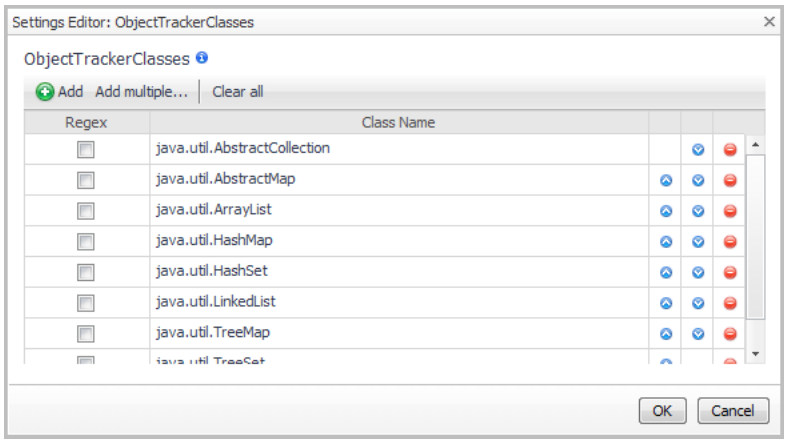
3 On the Java Administration dashboard, click the **Agent Configuration** tab.

4 Click the name of the agent profile that you want to edit (for example, Default Java).

5 In the menu that opens, select **Edit instrumentation settings**.

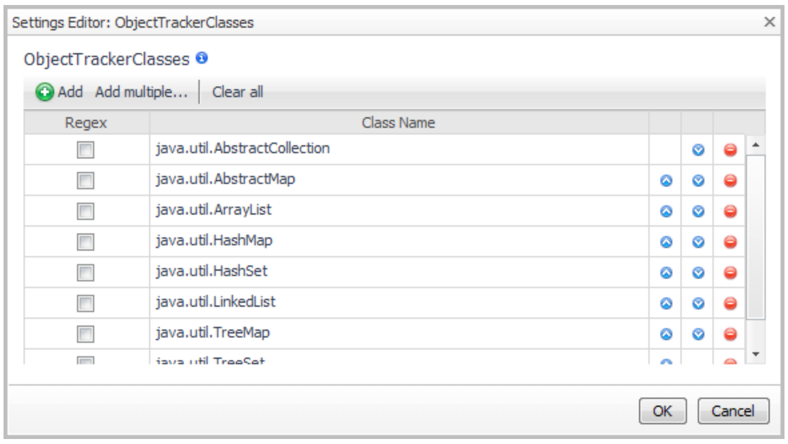
The Configuration Category Editor opens.

6 Click the **Object Tracking** tab.



Regex	Class Name
☐	java.util.AbstractCollection
☐	java.util.AbstractMap
☐	java.util.ArrayList
☐	java.util.HashMap
☐	java.util.HashSet
☐	java.util.LinkedList
☐	java.util.TreeMap
☐	java.util.TreeSet

7 Click the edit icon to the right of **ObjectTrackerClasses** to update the list of classes being included or excluded.


Settings Editor: ObjectTrackerClasses

ObjectTrackerClasses

Regex	Class Name
☐	java.util.AbstractCollection
☐	java.util.AbstractMap
☐	java.util.ArrayList
☐	java.util.HashMap
☐	java.util.HashSet
☐	java.util.LinkedList
☐	java.util.TreeMap
☐	java.util.TreeSet

For example, if you are only interested in tracking classes within the following two packages, `com.quest.pkg1.` and `com.quest.pkg2.`, do the following:

a In the Settings Editor dialog box, click **Add**. A blank row is added to the bottom of the list.

b Scroll down to the empty row.

c Click in the Class Name box of the empty row, and type the following: `com.quest.pkg1`.

d Click **Add** again.

e In the new blank box, type: `com.quest.pkg2`.

f Optional — If you want the class name to be interpreted as a regular expression (regex) instead of a literal string, click the check boxes beside the rows you added.

g Click **OK**. The new entries are added to the ObjectTrackerClasses list in the Configuration Category Editor dialog box.

IMPORTANT: These settings are valid when the Instrumentation Level is set to `FullDetail` or `ComponentDetail`. Changing their values has no effect if the Instrumentation Level is set to `BasicDetail` or `NoDetail`.

8 Click **Save**.

9 **Optional —** Edit the Nexus recording settings to set the maximum amount of time that the agent tracks a live object (that is, the `LiveObjectLifespanLimit)`. By default, the `LiveObjectLifespanLimit` is set to 900 seconds.

- Increasing this value gives the garbage collector more opportunity to collect live objects before the agent marks them as expired, but uses more memory in the application server process.
- Decreasing this value makes it more likely that the agent expires the live objects before they are collected, but uses less memory.

For more information, see Calculating the LiveObjectLifespanLimit value on page 48.

For detailed information about the Nexus recording settings, see the Managing Nexus Recording Settings section in the *Foglight for Application Servers Administration and Configuration Guide*.

10 **Optional —** Edit the Nexus recording settings to set the maximum number of live objects that the agent tracks. By default, the `LiveObjectCountLimit` is set to 524288.

- Increasing this value allows the agent to track more live objects concurrently, but uses more memory in the application server process.
- Decreasing this value makes it more likely that the agent expires the live object before it is collected, but uses less memory.

For detailed information about the Nexus recording settings, see the Managing Nexus Recording Settings section in the *Foglight for Application Servers Administration and Configuration Guide*.

11 Restart your application server.

Calculating the LiveObjectLifespanLimit value

Foglight for Java EE Technologies provides a suggested lifespan value for the agent to track a live object before marking it as expired. Depending on the complexity of your application along with your JVM heap settings and resulting garbage collection rate, the default value of 900 seconds (15 minutes) may not be appropriate.

An appropriate value for the `LiveObjectLifespanLimit` property can be derived from your observations of the time between the major garbage collections taking place. If the `LiveObjectLifespanLimit` is shorter than the time between major garbage collections, the agent may be prematurely marking objects as expired.

If you are able to have object tracking use additional memory, you may want to increase the `LiveObjectLifespanLimit` value to span the interval between major garbage collections. However, keep in mind that increasing this value does require additional memory within the application server heap.

Enabling object tracking

Prerequisite:

- Updating object tracking agent properties

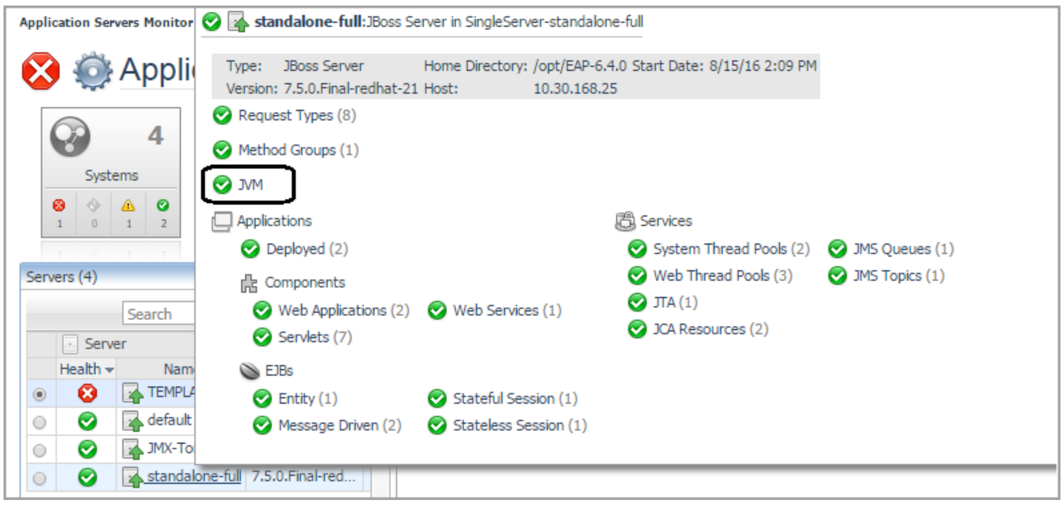
After you have set the required agent properties, enable object tracking for Foglight for Java EE Technologies.

To enable object tracking:

1 On the navigation panel, under **Dashboards**, click **Application Servers > Monitor**.

2 Click the **Servers** tile and select a server.

3 In the server details view, click **JVM** (not the status icon).

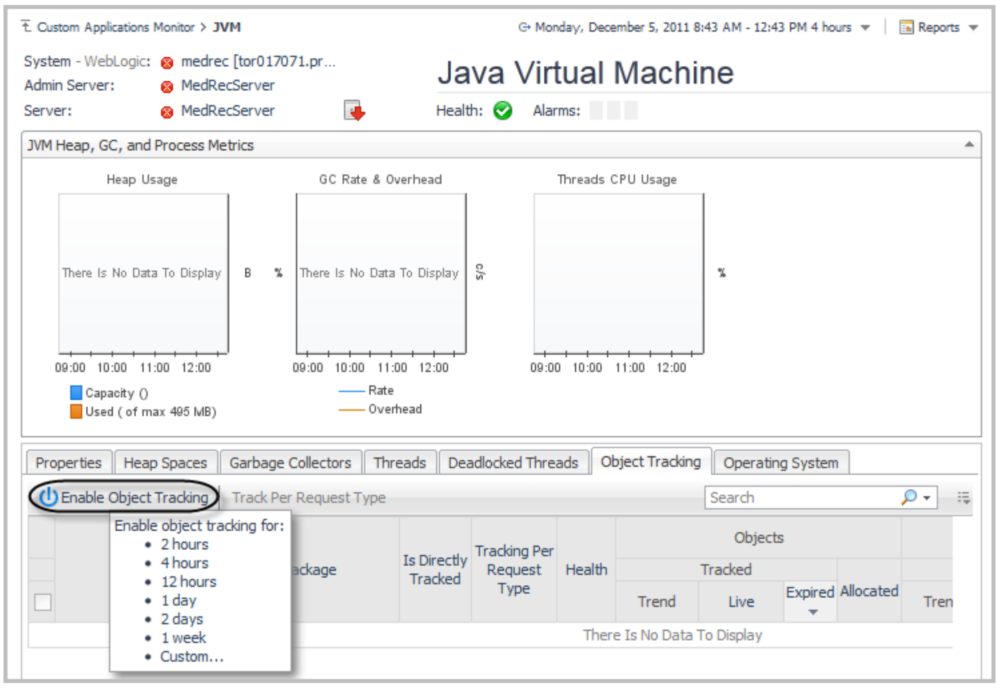


Type: JBoss Server Home Directory: /opt/EAP-6.4.0 Start Date: 8/15/16 2:09 PM
Version: 7.5.0.Final-redhat-21 Host: 10.30.168.25

4 In the Java Virtual Machine (JVM) view, click the **Object Tracking** tab.

5 Do one of the following:

- Click **Enable Object Tracking**.

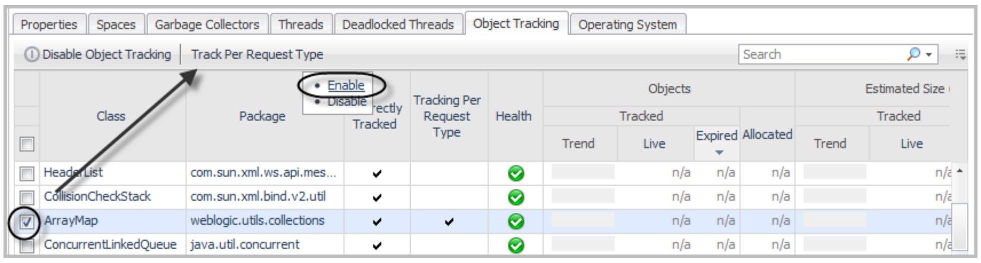


System - WebLogic: medrec [tor017071.pr...
Admin Server: MedRecServer
Server: MedRecServer

Select the length of time (duration) to enable object tracking.

or

- Select one or more classes from the list, then click **Track Per Request Type**.



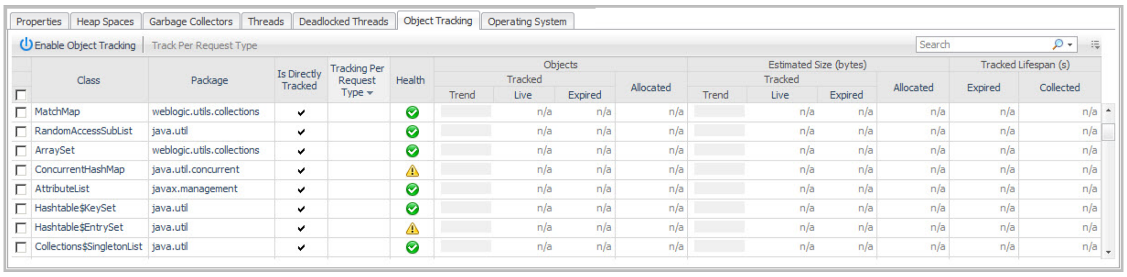
	Class	Package	Is Directly Tracked	Tracking Per Request Type	Health	Objects				Estimated Size	
						Tracked			Allocated	Tracked	
						Trend	Live	Expired		Trend	Live
	HeaderList	com.sun.xml.ws.api.mes...	✓		✓		n/a	n/a	n/a		n/
	CollisionCheckStack	com.sun.xml.bind.v2.util	✓		✓		n/a	n/a	n/a		n/
✓	ArrayMap	weblogic.utils.collections	✓	✓	✓		n/a	n/a	n/a		n/
	ConcurrentLinkedQueue	java.util.concurrent	✓		✓		n/a	n/a	n/a		n/

Select **Enable**.

Examining object tracking data

The results of any object tracking requests for Foglight for Java EE Technologies are listed on the Object Tracking tab of the Java Virtual Machine (JVM) view.

Figure 16. The Object Tracking tab view.



	Class	Package	Is Directly Tracked	Tracking Per Request Type	Health	Objects				Estimated Size (bytes)				Tracked Lifespan (s)	
						Tracked			Allocated	Tracked			Allocated	Expired	Collected
						Trend	Live	Expired		Trend	Live	Expired			
	MatchMap	weblogic.utils.collections	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	RandomAccessSubList	java.util	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	ArraySet	weblogic.utils.collections	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	ConcurrentHashMap	java.util.concurrent	✓		⚠		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	AttributeList	javax.management	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	Hashtable$KeySet	java.util	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	Hashtable$EntrySet	java.util	✓		⚠		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a
	Collections$SingletonList	java.util	✓		✓		n/a	n/a	n/a		n/a	n/a	n/a	n/a	n/a

Review the *Live* and *Expired* values. Classes with a large *Expired* value should be considered for investigation.

The Object Tracking tab contains a table that shows class group data for the selected JVM. This table contains the following information:

Table 21. Information contained in the class group data table for a selected JVM.

Data Item	Description
Class	Name of the class.
Package	The package of the tracked class.
Is Directly Tracked	This column is checked if the class is directly tracked. Classes that are directly tracked are those specified in the `ObjectTrackingClasses` property in agent property file. Classes that are not directly tracked are allocated by directly tracked classes, but these classes are only tracked when allocated by a directly tracked class.
Tracking Per Request Type	If checked, classes are tracked on a per request type basis when object tracking is enabled. It is recommended that per request type tracking is done only when actively investigating a specific class. Narrowing down your investigation places less overhead on your JVM.
Health	State of the class health. You can click the icon in the Health column to obtain alarm information.
Objects > Tracked > Trend	Trend of live objects displayed in a spark line.
Objects > Tracked > Live	The number of live objects tracked.
Objects > Tracked > Expired	The number of tracked expired objects.
Objects > Allocated	The number of allocated objects.
Estimated Size (bytes) > Tracked > Trend	Trend of live objects displayed in a spark line.
Estimated Size (bytes) > Tracked > Live	An estimated size of the tracked live objects.
Estimated Size (bytes) > Tracked > Expired	An estimated size of the tracked expired objects.
Estimated Size (bytes) > Allocated	An estimated size of the allocated objects.
Tracked Lifespan(s) > Expired and Collected	The estimate value of Expired and Collected life spans.

Reviewing track per request type data

The Object Tracking tab also offers the option to track by request type. This option is useful when you have identified a suspected memory leak source.

To enable tracking per request type:

1 On the Object Tracking tab of the Java Virtual Machine (JVM) view, select a class in the Object Tracking table.

2 Click **Track Per Request Type**.

3 Click **Enable**.

4 Wait for the data to collect.

5 Click the name of the class (in the **Class** column) to review a list of request types that allocate the object of the class.

From the Object Tracking Class view, you can review the metrics to identify a list of request types potentially causing the memory leak. Then you can collect single traces for that request type to find out where the memory has been allocated.

8

Monitoring Methods

The Methods tile on the Application Servers Monitor dashboard allows you to monitor specific named methods. From here you can review the invoked operations used by specific applications.

NOTE: Method monitoring is only applicable to Foglight for Java EE Technologies.

After your administrator has configured the named methods to monitor (see Collecting metrics on Java methods on page 52), you can use the **Methods** view to investigate how often a named method is called and to monitor the results of those called methods. For example, you can use Named Methods to monitor specific methods that connect to external systems.

All calls to the defined methods are tracked. The information presented is independent of request sampling. The metrics capture the behavior of the methods invoked during a request.

You can also find information on the Java request types that invoked the named method. For example, the number of log ins, calls to external system, and so on.

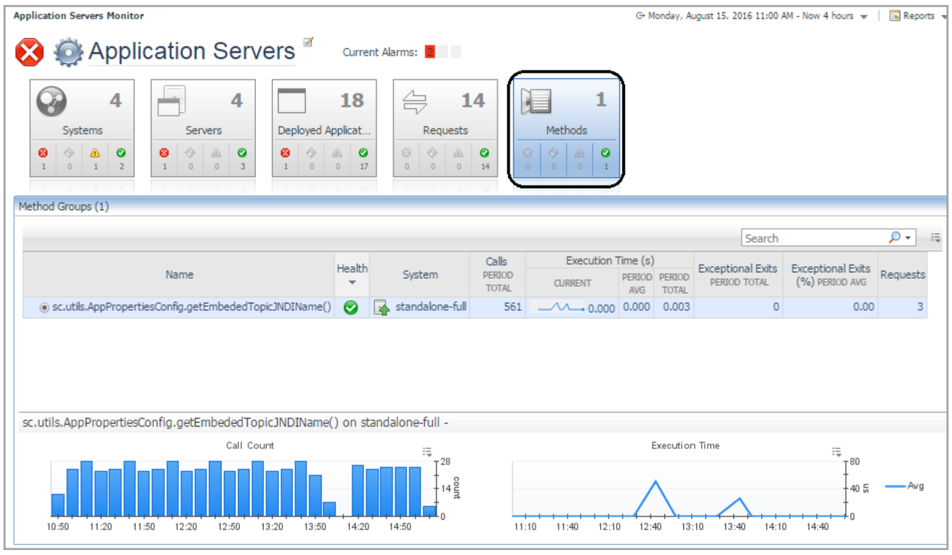
The Methods view shows you detailed information about the named methods being monitored in your environment.

From this view, you can perform the following actions:

- Monitor metrics associated with named methods.
- Review the invoked operations used by specific applications.
- Review charts that display metrics about the named methods.
- Collect Single Traces (from the action panel General tab)

To view the named methods:

1. On the navigation panel, under Dashboards, click **Application Servers > Monitor**.
2. On the Application Servers Monitor dashboard, click the **Methods** tile.


Application Servers Monitor

Monday, August 15, 2016 11:00 AM - Now 4 hours

Application Servers

Current Alarms: 2

Systems	Servers	Deployed Applicat...	Requests	Methods
4	4	18	14	1
1 0 1 2	1 0 0 3	1 0 0 17	0 0 0 14	0 0 0 1

Method Groups (1)

Name	Health	System	Calls PERIOD TOTAL	Execution Time (s) CURRENT	Execution Time (s) PERIOD AVG	Execution Time (s) PERIOD TOTAL	Exceptional Exits PERIOD TOTAL	Exceptional Exits (%) PERIOD AVG	Requests
sc.utils.AppPropertiesConfig.getEmbededTopicJNDIName()	✓	standalone-full	561	0.000	0.000	0.003	0	0.00	3

sc.utils.AppPropertiesConfig.getEmbededTopicJNDIName() on standalone-full -

The Methods view displays a Request Types table and corresponding metric chart for the selected method. In the chart area, you can quickly review the Call Count, Execution Time and Exceptional Exit metrics for the selected method.

Table 22. Information contained in the Method Group table.

Data Item	Description
Name	Name of the method group. Method group names at the top level are metrics at the domain level. Method Group names in the sub-items are at the server level.
Health	Health status of the method group. From this link you can access a list of outstanding Alarms and information on the changes made.
System	Name of the system the method is connected to. From this link you can select servers, applications, and clusters in your infrastructure that the method is calling.
Calls	Total number of calls for the selected time period.
Execution Time (s)	Execution time, in seconds, for the current period, the period average, and the period total.
Exceptional Exits	Total number of exceptional exits for this method.
Exceptional Exit%	Percentage of invocations that exited in exception.
Requests	Number of requests that reference the selected method.

Collecting metrics on Java methods

The Methods view allows you to view metrics selectively collected on one or more Java methods. To collect data for the Methods view, specify the methods to track in the agent instrumentation configuration. To define these methods you can specify the fully qualified class name, method name, and method arguments in the agent configuration profile.

NOTE: It is recommended that you make a backup of the default agent profile before editing.

NOTE: Restart your instrumented application servers after editing the instrumentation configuration (*agent/instrumentation.config*) in order for the changes to take effect.

For example:

You can group overloaded methods together. In the example below, we are defining a method group which uses similar methods. `Get Balance` is displayed in the user interface and all the methods within that group are aggregated.

```
NamedMethods = {
    "Get Balance": MethodList(
     include "com.globex.Account.close()",
     include "com.globex.Account.close(Ljava/lang/Integer;Z[B)I|)”,
     include "com.globex.Account.close(Ljava/lang/String;Z[B)I)",
   ),
}
```

To define named methods:

1. On the navigation panel, under **Dashboards**, click **Application Servers > Administration**.
2. On the Application Servers Administration dashboard, click the **Java tab**.
3. On the Java Administration page, click the **Agent Configurations** tab.

4 Create a backup of the configuration that you are about to edit.

   a Click the configuration name.

   b In the menu that opens, select **Copy**.

   c In the dialog box that opens, type a new name (for example, Default_Backup) in the **New ID** box.

   NOTE: You cannot use spaces in this name. Underscores, hyphens, periods, and alphanumeric characters are allowed.

   d Click **Copy**.

   e A message box opens. Click **Close**.

5 In the list of agent configurations, click the name of the configuration again.

6 Select **Edit instrumentation settings**.

   The Configuration Category Editor opens.

7 Click the **Named Methods** tab.

8 Click Edit to the right of **NamedMethods**.

9 In the Settings Editor, click **Add named method**.

10 In the dialog box that opens, type the name of the named method

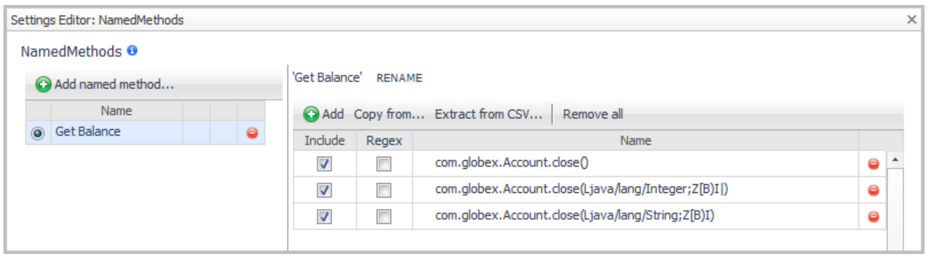
11 Click **OK**.

   The Settings Editor refreshes and a table displays on the right.

12 Click **Add** to add a row to the method list table.

13 Click in the **Name** box and type the name of the method.

   TIP: To use a regular expression, click the **Regex** check box.



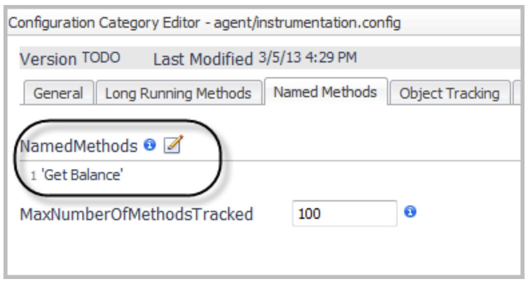
Name
Get Balance

Include	Regex	Name
✓	☐	com.globex.Account.close()
✓	☐	com.globex.Account.close(Ljava/lang/Integer;Z[B)I)
✓	☐	com.globex.Account.close(Ljava/lang/String;Z[B)I)

14 To add another method, repeat steps 11 and 12.

15 After adding all the methods, click **OK**.

   The Settings Editor closes, and the named method appears in the list on the Named Methods tab of the Configuration Category Editor.



Configuration Category Editor - agent/instrumentation.config
General
Long Running Methods
Named Methods
Object Tracking

Version TODO Last Modified 3/5/13 4:29 PM

NamedMethods
1 'Get Balance'

MaxNumberOfMethodsTracked 100

> **IMPORTANT:** These settings are valid only for `FullDetail` and `ComponentDetail` Instrumentation Levels. Changing the values has no effect if the Instrumentation Level is `BasicDetail` or `NoDetail`. You can set the instrumentation level on the General page of the Configuration Category Editor for the agent instrumentation profile.

16 Click **Save**. The Configuration Category Editor closes.

Understanding the implications of using named methods

Although the `NamedMethods MethodList` accepts a broad package definition, it is not recommended that you define your `NamedMethods` this way. Excessive `NamedMethod` instrumentation may increase the load on the Foglight Management Server and overhead on the application server, so it is best to include only the methods that you are interested in. To guard against excessive instrumentation, there is a limit on the number of `NamedMethods` that can be instrumented. Once this limit has been reached, no more methods are instrumented to display in the Application Servers Monitor > Methods view. A warning is logged for each method that is configured to be instrumented but is not instrumented when exceeding the allowed number of `NamedMethods`.

If you have configured too many `NamedMethods` and you are getting this warning, the first course of action is to modify your `NamedMethods MethodList` to only include the methods you are interested in or to be more specific in your named method definitions. If you have done that and are still getting warnings, you may increase the limit by editing the `MaxNumberOfMethodsTracked` property in the agent instrumentation profile. The default value is 100.

> **TIP:** You can set the MaxNumberOfMethodsTracked on the Named Methods tab of the Configuration Category Editor.

For more information about editing agent instrumentation configurations, see the *Foglight for Java EE Technologies Installation Guide*.

Application Servers Monitor Views

The Application Servers Monitor dashboard provides a high-level overview of your monitored environment that is divided into five main views: Systems, Servers, Deployed Applications, Requests, and Methods. You can drill down from any of these main views for more information about a selected component.

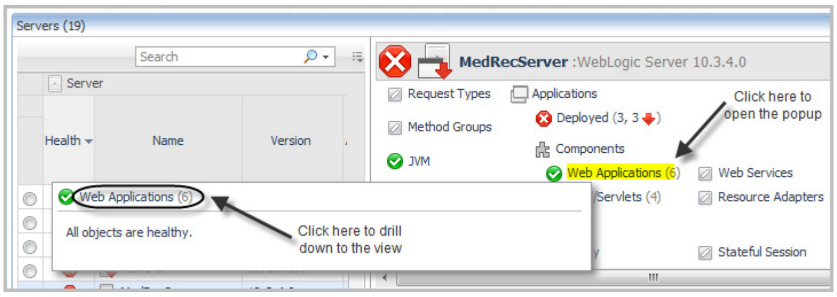
Hover your mouse over any text on the dashboard to determine whether it is a link, and click any link to open a pop-up or view.

To drill down to a view:

1 On the **Application Servers Monitor** dashboard, click a tile (**Systems**, **Servers**, Deployed **Applications**, **Requests**, or **Methods**).

2 In the selected tile view, click the name of the item you want to investigate.

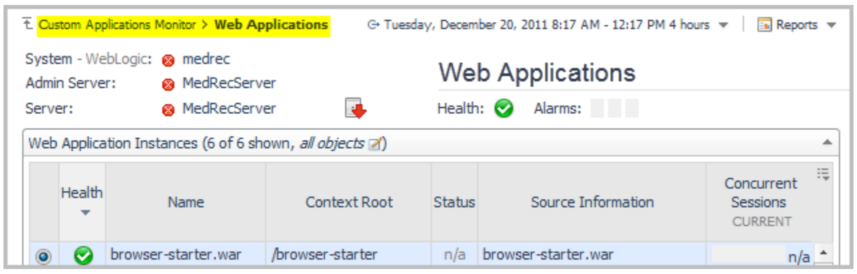
For example: click the **Servers** tile, select a WebLogic® server, and click the **Web Applications** link under Applications Components. A pop-up appears, listing the health status of the selected component.



All objects are healthy.

3 In the upper left corner of the pop-up, click **Web Applications** to open the drilldown view in the display area.

> **TIP:** Use the breadcrumb trail at the top of any drilldown view to return to the Application Servers Monitor dashboard.


Custom Applications Monitor > Web Applications
Tuesday, December 20, 2011 8:17 AM - 12:17 PM 4 hours
Reports
Web Applications
Health:
Alarms:

System - WebLogic: medrec
Admin Server: MedRecServer
Server: MedRecServer

Web Application Instances (6 of 6 shown, *all objects*)

Health	Name	Context Root	Status	Source Information	Concurrent Sessions CURRENT
✓	browser-starter.war	/browser-starter	n/a	browser-starter.war	n/a

JVM view

Purpose

Use this view to examine the status of the Java Virtual Machine (JVM) for a selected system or server.

How to get here

Drill down on any **JVM** link.

Table 23. Heap, GC, and Process Metrics Charts

Description	**Heap Used & Committed Chart —** plots the heap usage over time. **GC Rate & Overhead Chart —** plots the garbage collection rate and overhead over time. **Threads CPU Usage —** plots the amount of CPU resources used by JVM threads over time.

Table 24. Tabs

Properties	Provides an overview of the Java Virtual Machine, including version and architecture, and the runtime name.
Spaces	Provides detailed heap and non-heap space information in chart format.
Garbage Collectors	Provides information on the garbage collection cycles.
Threads	Provides charts for the live, waiting, deadlocked, and blocked thread counts over time.
Deadlocked Threads	Provides details about any deadlocked threads.
Object Tracking	Use this tab to enable or disable object tracking. This feature is only available when Foglight for Java EE Technologies is installed on your Management Server. For more information, see Using Object Tracking to Locate Memory Leaks on page 45.
Operating System	Use this tab to review information about the OS-specific memory and CPU usage, including Committed Virtual Memory, Total Physical Memory, and Total Swap Space.

Method Groups view

Purpose

Use this view to examine the status of Java EE Methods for a selected system or server.

How to get here

Drill down on any **Method Groups** link from the Systems or Servers views.

Table 25. Method list

Description	Lists the name, health, system, and statistics for a selected method. Charts at the bottom of the view display the Call Count and Execution Time averages over a selected period.

Request Types view

Purpose

Use this view to examine the status of requests for a selected system or server.

How to get here

Drill down on any **Requests** link in the Systems view, or any **Request Types** link in the Servers view.

Table 26. Requests list

Description	Lists the name, health, system, and statistics for a selected request. Tabs at the bottom of the view display the Response Time/Call Count, Sampled Breakdowns, and Incomplete/Exceptional Exits. For more information, see Monitoring Requests on page 28.

Entity EJBs view

Purpose

Use this view to examine runtime information about the entity beans in the selected application.

How to get here

Drill down on any **Entity** link.

Table 27. Entity Bean Instances table

Description	Lists all entity beans in the application and summarizes the current state of each bean, including its health status, pool usage, and rate activities. The graphs below the table display data for the selected bean.

NOTE: WebLogic versions earlier than 9.x do not provide Query Cache metrics for EJBs.

Message Driven EJBs view

Purpose

Use this view to examine runtime information about the message driven beans in the selected application.

How to get here

Drill down on any **Message Driven** link.

Table 28. Message Driven Bean table

Description	Lists all message driven beans in the application and summarizes the current state of each bean, including its health status, pool usage, and rate activities. The graphs below the table display data for the selected bean.

Stateful Session EJBs view

Purpose

Use this view to examine runtime information about the stateful session beans in the selected application.

How to get here

Drill down on any **Stateful Session** link.

Table 29. Stateful Session Bean table

Description	Lists all stateful session beans in the current application and summarizes the current state of each bean, including its health status, cache, and rate activities. The graphs below the table display data for the selected bean.

Stateless Session EJBs view

Purpose

Use this view to examine runtime information about the stateless session beans in the selected application.

How to get here

Drill down on any **Stateless Session** link.

Table 30. Stateless Session Bean table

Description	Lists all stateless session beans in the current application and summarizes the current state of each bean, including its health status, pool usage, and rate activities. The graphs below the table display data for the selected bean.

Deployed Applications view

Purpose

Use this view to review the status of all monitored applications deployed to a selected application server.

How to get here

Drill down on the **Deployments** link.

Table 31. Deployed Applications table

Description	Lists the alarms, status, and name or target of all applications deployed to a selected server.

JSPs/Servlets components view

Purpose

Use this view to monitor the health and performance of active servlets of a specified application residing on a specified application server.

How to get here

Drill down on any **JSPs/Servlets** or **Servlets** link.

Table 32. JSP/Servlet Instances table

Description	Lists all the servlets belonging to the selected application and summarizes the status of each servlet, including its health, related web application or module, and service times. The graphs below the table show data for the selected servlet.

Resource Adapters components view

Purpose

Use this summary view to monitor the health and connection information for server resource adapters.

How to get here

Drill down on any **Resource Adapters** link.

Table 33. Resource Adapter Instances table

Description	Lists the resource adapter instances for the selected server and summarizes the health, connections, connector, and resource information.

Web Applications components view

Purpose

Use this summary view to monitor the status of web applications in the selected domain, cell, server, or application.

How to get here

Drill down on any **Web Applications** link.

Table 34. Web Application table

Description	**NOTE:** The details listed are specific to the type of application server selected. Lists all the web applications in the selected domain or application and summarizes the status, including whether it is active, the source and context root, and session information. The graphs below the table show historical data for the selected web applications.

Web Services components view

Purpose

Use this summary view to monitor the status of web services in the selected domain or application.

How to get here

Drill down on any **Web Services** link.

Table 35. Web Service Instances Table

Description	Lists all the web services in the selected domain and summarizes the health of each service, including the number of active operations.

.NET views

This section describes the Systems and Applications views that are specific to Microsoft.NET.

- Application Pools view
- Executable Applications/Services view
- Sites view
- Web Applications view

Application Pools view

Purpose

Use this summary view to monitor the status of server application pools in the selected .NET server group.

How To Get Here

Drill down on the **Application Pools** link on the Systems or Applications view.

Table 36. Application Pools Table

Description	Lists all the application pools in the selected .NET Server Group and summarizes the status, including availability, processor time used, garbage collection cycles and time, exceptions, and processes. The graphs below the table show historical data (availability, percent processor time, heap size, and promoted bytes) for the selected server application pool.

Executable Applications/Services view

Purpose

Use this summary view to monitor the status of executable applications and Windows® Services in the selected .NET server group.

How to get here

Drill down on the **Applications** link on the Systems view.

Table 37. Applications table

Description	Lists all the applications in the selected .NET Server Group and summarizes the status, including processor time used, garbage collection cycles and time, exceptions, and processes. The graphs below the table show historical data (heap size, promoted bytes, and active threads) for the selected executable application or Windows Service.

Sites view

Purpose

Use this summary view to monitor the status of web sites in the selected .NET server group.

How to get here

Drill down on the **Sites** link on the Systems or Applications view.

Table 38. Sites table

Description	Lists all the web sites in the selected .NET Server Group and summarizes the status, including availability, current connections, method request rate, errors, and bytes sent and received. The graphs below the table show historical data (current connections, method requests, and total errors) for the selected web site. You can view this data by Connections and Errors, by Bindings, or by Web Applications.

Web Applications view

Purpose

Use this summary view to monitor the status of web applications in the selected .NET server group.

How to get here

Drill down on the **Web Applications** link on the Systems view.

Table 39. Web Applications table

Description	Lists all the web applications in the selected .NET Server Group and summarizes the status, including application pool, site, requests, sessions, and transactions. The graphs below the table show historical data (requests, requests rate, sessions, and transactions) for the selected web application.

JBoss Services views

This section describes the Services views that are specific to JBoss:

- JBoss JCA Resources view
- JBoss JTA view
- JBoss Thread Pools view
- JBoss JMS Cache/Queues/Topics views

JBoss JCA Resources view

Purpose

Use this view to examine the runtime status of the resource adapter connection pools.

Table 40. Connection Pools table

Description	Lists all the connection pools associated with the selected resource adapter and summarizes the status of each pool, including whether it is active, its health status, capacity, waiters, and information about connections. The graphs below the table display data for the selected resource adapter.

JBoss JMS Cache/Queues/Topics views

Purpose

Use this summary view to examine the health of the Java messaging service (JMS) running in the selected JBoss application server.

Table 41. Message Cache tab

Description	Lists the number of cache hits, misses, and total cache size. The graphs below the table display historical data for the message cache.

Table 42. Queues tab

Description	Lists the number of receivers, scheduled message count, and depth details. The graphs below the table display historical data for the queue.

Table 43. Topics tab

Description	Lists durable and total count details for messages and subscriptions. The graphs below the table display historical data for the topic.

JBoss JTA view

Purpose

Use this view to examine runtime information on transactions that the selected JBoss application server handles. Transactions can rollback, commit, or time out.

Table 44. Transactions Managers table

Description	Lists all the transaction managers and summarizes the status of each transaction, including its health status, total number of transactions, commit counts, and rollback data. The charts below the table reflect the data for the selected transaction.

JBoss Thread Pools view

Purpose

Use this view to obtain an overview of the performance of thread pools in the selected JBoss application server.

Table 45. System Thread Pools table

Description	Lists all the thread pools for the selected JBoss application server and summarizes the current state of each pool, including its health, utilization, configured size, and queue information. The System Threads charts below the table display historical data for the selected pool.

Table 46. Web Thread Pools table

Description	Lists all the web thread pools for the selected JBoss application server and summarizes the current state of each pool, including its health, and the number of threads that are currently busy, in use, and the maximum number of configured threads. The chart below the table displays historical data for the selected pool.

Oracle Services views

This section describes the Services views that are specific to OracleAS:

- OracleAS JCA Connection Pools view
- OracleAS JDBC Connection Pools view
- OracleAS JMS view
- OracleAS JTA view
- OracleAS Thread Pools

OracleAS JCA Connection Pools view

Purpose

Provides an overview of the performance of the JCA connection pools deployed to the Oracle OC4J process.

Table 47. JCA Connection Pool table

Description	Lists all JCA connection pools in the selected OC4J process and summarizes the status of the pool usage, waiters, and faults. The charts below the table show use time and wait time data for the selected connection pools.

OracleAS JDBC Connection Pools view

Purpose

Use this view to monitor the JDBC Connection Pools that the selected application uses.

Table 48. JDBC Connection Pools table

Description	Lists all JDBC Connection Pools in the selected application and summarizes its status, including whether it is active, its health status, and information on connections. The charts below the table show use time and wait time data for the selected connection pool.

OracleAS JMS view

Purpose

Use this view to examine the health of the selected OC4J process messaging service. This view contains information on messages that have been sent, are expired, or have been rolled back, among other data.

Table 49. Current Statistics table

Description	Displays the current JMS runtime statistics, including the messages expired, queued, enqueued, recovered, discarded, pagedout, pagedin, active handlers, and active connections. The charts below the table display data for the selected item.

OracleAS JTA view

Purpose

Use this view to examine runtime transactions that the selected OC4J process handles. This view contains information on transactions that have committed, rolled back, or timed out.

Table 50. JTA table

Description	Displays the current JTA runtime statistics, including the number of active transactions, the number of named transactions by server, total transactions, cumulative transaction counts, and rollback data. The charts below the table display data for the selected item.

OracleAS Thread Pools

Purpose

Use this view to examine thread pool information for a selected host.

Table 51. Thread Pools table

Description	Lists all the thread pools for the selected OC4J process and summarizes the current state of each pool, including its health, thread usage, and information on requests. The charts below the table display pool size and queue size data for the selected pool.

Tomcat Services views

This section describes the two Services views that are specific to Tomcat:

- Tomcat Datasources
- Tomcat Thread Pools

Tomcat Datasources

Purpose

Use this view to monitor the health of a datasource, its URL, target, and connection information.

Table 52. Datasources table

Description	Lists all the host datasources being monitored, their global and application-specific health, name, URL, target and connections values for active and idle.

JDBC Charts

Description	Represents the active connections and the number of idle connections.

Tomcat Thread Pools

Purpose

Use this view to examine thread pool information for a selected Tomcat host.

Table 53. Thread Pools table

Description	Lists all thread pools that the selected Tomcat host uses and summarizes the current state of each pool, including its health, thread usage, and information about requests. The charts below the table display historical data for the selected pool.

WebLogic Services views

This section describes the Services views that are specific to WebLogic application servers:

- WebLogic Execute Queues view
- WebLogic JDBC Data Sources view
- WebLogic JMS Servers view
- WebLogic Security view
- WebLogic Self-Tuning Thread Pool view
- WebLogic Store and Forward view
- WebLogic Work Managers view

WebLogic Execute Queues view

Purpose

Use this view to examine runtime information for all the execute queues in the selected WebLogic server.

Table 54. Execute Queues table

Description	Lists all the queues and thread pools for the selected WebLogic server and summarizes the current state of each queue or pool, including its health, thread usage, and information on requests. The graphs below the table display data for the selected queue or pool.

WebLogic JDBC Data Sources view

Purpose

Use this view to monitor the health of the connection pools that the selected WebLogic server uses.

An important indicator of the performance of a JDBC connection pool is the presence or lack of waiters. Especially important to observe is the presence of failed waiters (in the Connection Failures After Waiting column) as these are waiters that were unable to successfully obtain a database connection.

Table 55. JDBC Data Sources table

Description	Lists all the JDBC data sources in the selected WebLogic server and summarizes the state of each data source, including whether it is active, its health status, as well as information on connections, waiters, and the prepared statement cache. The graphs below the table display historical data related to connection delays, capacities, number of requests, requests forced to wait, and the occurrence of leaked connections.

WebLogic JTA Runtime view

Purpose

Use this view to examine runtime transactions the selected WebLogic server handles. Specifically, it displays transactions that are processing successfully (committing), transactions that cause errors (rollbacks), and transactions that time out (abandoned).

There are four types of rollbacks:

- Application rollback is expected to occur during the running of an application. High occurrences of application rollbacks may point to a problem in the application.
- Resource rollbacks can represent problems in the WebLogic resource, such as an unavailable database.
- System rollbacks can represent serious problems in the WebLogic server.
- Timeout rollbacks can occur when a transaction cannot complete within the configured timeout interval. This indicates either that you have to increase the interval to allow more time for transactions to complete or that there is a problem in the application or environment that is causing transactions to take too long to complete.

Table 56. JTA table

Description	Displays the current JTA runtime statics, including the number of active transactions, the number of named transactions by server, total transactions, cumulative transaction counts, and rollback data. The graphs below the table reflect the historical data in the table.

WebLogic JTA Named Transactions view

Purpose

Use this view to examine runtime information on named transactions that the selected WebLogic server handles.

To access this view, in the WebLogic JTA Runtime view, click **View Named Transactions**.

Table 57. Named Transactions table

Description	Lists all the named transactions and summarizes the status of each transaction, including its health status, total number of transactions, cumulative transaction counts, and rollback data. The graphs below the table reflect the data for the selected transaction.

WebLogic JMS Servers view

Purpose

Use this view to examine the health and statistics of the local and remote JMS Servers.

You can drill down from this view to the WebLogic JMS Server Destinations view and WebLogic JMS Server Session Pools view.

Table 58. JMS Servers tab

Description	Lists all the JMS servers running in the selected WebLogic server and summarizes the status of each JMS server, including the number of messages and bytes (current and pending), session pools, destinations, connections, and which activities are paused. The graphs below the table display historical data for the selected JMS server.

WebLogic JMS Server Destinations view

Purpose

You can use this detail view to examine the destinations available for the selected JMS server.

To access this view, in the **Local or Remote Servers** table of the WebLogic JMS Servers view, click a value in the **Destination** column. The time spent in queue details displays.

Table 59. JMS Server Destination table

Description	Lists all the destinations of the selected JMS server running in the selected WebLogic server and summarizes the status of each destination, including the type, its state, number of messages and bytes (current and pending), and which activities are paused.

WebLogic JMS Server Session Pools view

Purpose

You can use this detail view to examine the session pools available for the selected JMS server.

To access this view, in the Local or Remote Servers table of the WebLogic JMS Servers view, click a value in the **Session Pools** column to view details.

Table 60. JMS Server Session Pools table

Description	Lists all the session pools for the selected JMS server running in the selected WebLogic server and summarizes the status of each pool, including its health and usage.

WebLogic Security view

Purpose

Use this view to examine runtime information about user logins and user lockouts for the selected WebLogic server.

Table 61. Security table

Description	Displays information about user logins and lockouts for the selected WebLogic server. It also states whether JACC (J2EE Authorization Contract for Containers) is used for managing security and role-based authorization. The graphs below the table display the same data plotted over time.

WebLogic Self-Tuning Thread Pool view

Purpose

Use this view to examine runtime statistics for the thread pool in the selected WebLogic server.

Table 62. Self-Tuning Thread Pool Table

Description	Displays statistics for the self-tuning thread pool in the selected WebLogic server, including data about requests and thread usage. The throughput and pending user request data is an aggregate of all work managers in the system. The graphs below the table display historical data.

WebLogic Store and Forward view

Purpose

Use this view to examine statistics for the store-and-forward agents in the selected WebLogic server.

Table 63. Store-and-Forward Agents table

Description	Displays the health, name, service type, window size, and message statistics for the store-and-forward agents.

WebLogic Work Managers view

Purpose

Use this view to examine runtime statistics for the work managers in the selected WebLogic server.

Table 64. Work Managers table

Description	Lists all the work managers associated with the self-tuning thread pool. For each work manager, you can see the rate of completed requests and the number of requests that are pending. The graphs below the table display data for the selected work manager. This data includes: • Pending Requests — the number of requests that were waiting for an available thread. If this value is high, then it is possible that your work manager or thread pool is configured with too few resources. • Request Throughput — the number of requests processed per second. Throughput is a good indicator of the amount of work that the WebLogic application server is processing. In a tuning effort, your goal is to minimize response time while maximizing throughput.

WebSphere Services views

This section describes the Services views that are specific to WebSphere application servers:

- WebSphere Dynamic Cache view
- WebSphere JCA Factories view
- WebSphere JDBC Data Sources view
- WebSphere JTA Runtime view
- WebSphere Listener Ports view
- WebSphere SIB view
- WebSphere Thread Pools view

WebSphere Dynamic Cache view

Purpose

Use this view to get an overview of the dynamic caches for the WebSphere server. WebSphere 6.x defines two caches: one for servlets and one for objects. The purpose of the dynamic cache is to capture dynamically generated content and serve it from a cache rather than recomputing it each time. The charts capture the number of objects in the cache and the number of satisfied or missed requests.

You can review the dynamic cache by cache objects, or by cache templates using the **View by** links in this view.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **Dynamic Cache** link.

Table 65. Cache Objects table

Description	Lists cache type, size, requests, invalidations, and their hit and miss rates. The graphs below the table display the access rate, hit rate, invalidation rate, and miss rate plotted over time.

WebSphere JCA Factories view

Purpose

You can use this view to monitor the JCA Connection Factories for the selected WebSphere server.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **JCA Factories** link.

Table 66. Connection Factories table

Description	Lists all connection factories in the selected WebSphere server and summarizes its status, provider, connections, threads, and faults.

WebSphere JDBC Data Sources view

Purpose

You can use this view to monitor the JDBC Data Sources that the selected WebSphere server uses.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **JDBC Data Sources** link.

Table 67. JDBC Data Sources table

Description	Lists all JDBC Data Sources in the selected WebSphere server and summarizes its status, including whether it is active, its health status, as well as information on connections, waiters, and the prepared statement cache.

WebSphere JTA Runtime view

Purpose

Use this view to examine runtime transactions that the selected WebSphere cell handles. Specifically, it displays transactions that are processing successfully (committing), transactions that cause errors (rollbacks), and transactions that time out (timeouts), for both local and global transactions.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **JTA** link.

Table 68. JTA table

Description	Displays the current JTA runtime statics, including the number of active, committed, rolled back, and timed out transactions, both locally and globally. The graphs provide historical data for the transactions.

WebSphere Listener Ports view

Purpose

Use this view to examine the status of message listener ports for a selected WebSphere server.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **Listener Ports** link.

WebSphere SIB view

Purpose

You can use this view for a summary of the health of the WebSphere 6 JMS components. From this view you can drill down into the individual JMS components.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **SIB** link.

Table 69. Service Integration Bus Messaging Engines table

Description	Lists the name and summarizes the queue, topic space, and storage management health of the selected components.

WebSphere Thread Pools view

Purpose

Use this view to examine thread pool information for the selected WebSphere server.

How to get here

On the Application Servers Monitor dashboard, select a WebSphere server and drill down on the **Thread Pools** link.

Table 70. Thread Pools table

Description	Lists all the thread pools for the selected WebSphere server and summarizes the current state of each pool, including its health, thread usage, and information on requests. The charts below the table display historical data for the selected pool.

JMX Administration dashboard

Purpose

The JMX Administration dashboard is used to specify the type of data collected by creating, importing, or editing MBean Server Models.

How to get here

To access this dashboard, on the navigation panel, under Dashboards, click **Application Servers > Administration > JMX**.

Use this dashboard to do the following:

- Configure what MBean types are collected and change which MBean server model is used by an agent
- Select the Configured MBean Type and set Instance Collection Overrides, to reduce the amount of data collected
- Create, import, and export MBean server models

TIP: To show or hide columns in the tables on this view, click the customizer icon to the right of the Search box.

Description of the view

Table 71. Server Models tab

Data Displayed	• **ID.** The user-defined identifying name for the server model. • **Name**. The name of the MBean Server Model. If not specified, the ID is displayed. • **Description**. A description of the Server Model. • **Server Group.** The group to which the Server Model belongs. • **Used By**. The number of agents using a particular Server Model.
Where to go next	Drill down on: • **Create Server Model**. Starts a wizard for creating a server model. For more information, see Creating MBean Server Models in the *Foglight for JMX User Guide*. • **Import Server Model**. Opens a wizard to assist you in importing a previously created MBean server model. For more information, see Importing a Server Model in the *Foglight for JMX User Guide*. • **ID** or **Name.** Opens a menu of options for managing the server model. • **Used By.** For more information, see Agents Using Server Model view on page 73.

Agent Instance Overrides view

Purpose

Use this view to specify which MBean instances you want to collect for a selected agent. By default all exposed MBean objects are collected. By setting overrides, you are able to reduce the amount of unnecessary data you are collecting from your monitored hosts.

Use this view to do the following:

- Specify collection overrides to collect all available instances, or reduce the data collected by specifying only some available instances
- Use an object name query to specify which instances to include or exclude in your data collection

How to get here

On the Application Servers Administration dashboard, click the name of a JMX agent in the Monitoring Agents table. In the list that opens, select Agent Overrides.

Description of the View

Table 72. Available MBean Types table

Data Displayed	• **Type**. A list of MBean type instances available for collection. • **Agent Override**. The state of the overrides. • **Server Model**. The default state of the instances being collected, as defined by the server model.
Where to go next	Drill down on: **The value in the Type column. The Monitored Instance Overrides dialog box opens.**

Agents Using Server Model view

Purpose

Use this view to identify which agents are using a particular server model without leaving the Server Model tab of the JMX Administration dashboard.

Use this view to do the following:

- Specify collection overrides to collect all available instances, or reduce the data collected by specifying only some available instances.
- Use an object name query to specify which instances to include or exclude in your data collection.

How to get here

To access this view:

1. Open the JMX Administration dashboard.
2. If the **Used by** column is not included in the Server Models table, click the customizer icon (at the top right corner of the table) and use the menu to add the column to the table.
3. Click an entry in the **Used by** column for a server model that at least one agent is using.

Description of the view

Table 73. Agents Using Server Model table

Data Displayed	• **Name**. The name of the agent.

Server Model Editor dashboard

Purpose

Use this dashboard to review which MBean types are configured for collection. From here you can select and configure the attributes you want to collect, set the exposed relationships, customize the display name for the MBean type (so that the MBean information listed in the Java EE Monitor dashboard is more meaningful for the user,) and edit the type configuration.

How to get here

On the JMX Administration dashboard, click an entry in the **Name** column of the Server Models table. In the list that opens, select **Edit**.

Description of the view

Table 74. Configured MBean Types table

Data Displayed	• **Name**. The name of the MBean root type. • **Instance Availability**. Indicates the availability for monitoring of MBean instances of the root type. For example, All by Root Query indicates that all instances are available for monitoring using a root query. • **Monitored Instances**. Indicates the MBean instances that are monitored by default. • **Collected Attributes**. The number of attributes of the MBean instance that are being collected, as a fraction of all available attributes (for example 2/25). • **Exposed Relationships**. The number of relationships that have been exposed for the selected MBean instance. • **Display Name**. The display name of the MBean instance, if different from the default name.
Where to go next	**Drill down on:** • **Add Root Type**. Starts the wizard for adding root types. • **Update Type Definitions. Updates the list of type definitions, if at least one agent is connected to the MBean server.** • **Name**. Opens a list of options for managing configurations. • **Monitored Instances**. Opens a dialog box that allows you to define instances to monitor. • **Collected Attributes**. Opens a dialog box that allows you to select and configure attributes for collection. • **Exposed Relationships**. Opens a dialog box that allows you to expose MBean relationships. • **Display Name**. Opens a dialog box that allows you to change the display name for an MBean.

Table 75. Agents and Monitored Instance Overrides table

Data Displayed	• **Name**. The name of the agent. • **Overrides**. The number of MBean instance overrides for the agent, displayed as a fraction of total MBean instances available (for example, 02/25).
Where to go next	Drill down on: • **Overrides**. Opens the Instance Overrides dialog box for the selected MBean type and agent.

JMX Explorer dashboard

Purpose

Use the JMX Explorer dashboard to get a quick overview of how many MBeans are currently being monitored. From this view you can select an MBean to view its properties.

How to get here

On the navigation panel, under Dashboards, click **Application Servers > JMX > Explorer**.

Embedded views

This dashboard is composed of the following embedded views:

- Available MBeans
- MBean Summary view

Available MBeans

The Available MBeans view contains two tabs: List and Structure. Use these two tabs to view the MBean's Name, Health, and Type (Info) in either a list, or a tree structure format.

Table 76. Where to go next from the Available MBeans view.

Where to go next	Drill down on: • **Name**. For more information, see MBean Summary view on page 75. • **Health**. For more information, see MBean Health summary view on page 76.

MBean Summary view

Purpose

This view provides information about the MBean instance's health, Key Indicator Metrics, Relationships, Substats, and Properties.

Embedded views

This view is composed of the following embedded views:

- MBean Summary view
- MBean Health summary view
- MBean Metric summary view
- MBean Relationships view
- MBean Substats view
- MBean Properties view

How to get here

This view is displayed as an embedded view on the JMX Explorer dashboard when you select an MBean instance from the Available MBeans list.

To access this view for an MBean without changing the embedded view that is displayed in the dashboard, click the name of an MBean instance in either the List or Structure tab.

Description of the view

Table 77. MBean Summary View

Data Displayed	• **Health**. A health state icon. • **Name**. The name of the MBean. • **Key Indicator Metrics**. A list of any key indicator metrics for the MBean. • **Relationships**. A list of any exposed containment or reference relationships. • **Substats**. A list of any substats available. • **Properties**. A list of any properties (attributes) of the MBean.
Where to go next	Drill down on: • **Health**. For more information, see MBean Health summary view on page 76. • **Key Indicator Metrics**. Click the metric name to open the MBean Metric summary view for that metric. • **Relationships**. For more information, see MBean Relationships view on page 76. • **Substats**. For more information, see MBean Substats view on page 77. • **Properties**. Links to the property (attribute) Summary view. For more information, see MBean Properties view on page 78.

MBean Health summary view

Purpose

Provides an overview of the MBean's health status, and any outstanding alarms.

How to get here

Click the health icon for an MBean in either the MBean Summary view or the Available MBeans view.

MBean Metric summary view

Purpose

If data is available, MBean Metrics are displayed in this view. Click the metric name to display more details.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. If metrics are available, the view is populated.

Description of the view

Table 78. MBean Metric Details

Data Displayed	Metric Name and chart.
Where to go next	Click the metric name to open a Metric Summary with more details, including: description, the date it was last updated, and the source.

MBean Relationships view

Purpose

Use this view to examine the relationship details of a selected MBean.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. If relationships have been exposed, the view is populated.

Description of the view

Table 79. Relationship Details

Data Displayed	**Health icon**. Indicates the health of the MBean. **Name**. A list of MBeans that the selected MBean either contains or refers to.
Where to go next	Drill down on: **Health icon**. For more information, see MBean Health summary view on page 76. **Name**. For more information, see MBean Relationships view on page 76.

MBean Relationships pop-up

Purpose

Use this pop-up to examine the health and alarms on instances or types for a contained or referenced MBean.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. In the MBean Relationships view, click the name of a contained or referenced MBean.

Description of the view

Table 80. Instances tab

Data Displayed	**Alarms**. Icons indicating any current alarms on the MBean instance. **Name**. The name of the MBean instance.
Where to go next	Drill down on: **Alarms**. Opens an Alarms overview pop-up. **Name**. For more information, see MBean Summary view on page 75.

Table 81. By Type tab

Data Displayed	**Health**. Icons indicating any current alarms on the MBean instance. **Type Name**. The name of the MBean instance. **Instances**. The number of instances of that MBean type.
Where to go next	Drill down on: **Health icon**. For more information, see MBean Health summary view on page 76. **Instances**. For more information, see MBean Summary view on page 75.

MBean Substats view

Purpose

Use this view to review the sub-stats for an MBean.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. If substats are available, the view is populated.

Description of the view

Table 82. Substats details

Data Displayed	A list of sub-stats.

MBean Properties view

Purpose

Use this view to examine the values of any attributes (properties) for which data is collected.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. If attributes are being monitored, the view is populated.

Description of the view

Table 83. Properties Single Value or Observation Details

Data Displayed	Displays the names and values of MBean attributes.
Where to go Next	Drill down on: **Attribute Value**. For more information, see MBean Property summary view on page 78.

MBean Property summary view

Purpose

Use this view to review the latest value, period values, description, and source of the attribute.

How to get here

In the JMX Explorer Available MBeans view, select an MBean. In the MBean Properties view, click the value of an attribute.

Description of the view

Table 84. Properties single value or observation details

Data Displayed	Displays the name of the attribute and its latest value, period values (if available), description, and source.
Where to go Next	Drill down on: **Edit Configuration**. Opens the Attribute — Topology Property Configuration dialog box, which allows you to edit the name, storage type, and metric derivations (if applicable).

Java Virtual Machine (JVM) dashboard

Purpose

Use the JVM dashboard to review the operating status of a selected JVM.

How to get here

On the JMX Explorer dashboard, click the **JVM name**. The JVM Summary view pop-up opens. Click the **Heap Usage** chart to open the JVM dashboard.

Description of the view

Table 85. JVM details

Data Displayed	• System, Server, and JVM names and health icons • JVM Heap, GC, and Process Metrics charts • Properties, Spaces, Garbage Collectors, Threads, Deadlocked Threads, and Operating System tabs.
Where to go Next	Click any of the tabs to view details about the selected JVM metric.

A

Appendix: Regular Expressions

Regular expressions employ *metacharacters*. A metacharacter is a character with special meaning. It does not denote a normal letter, but instead affects the interpretation of other characters in a regular expression. Metacharacters only occur inside a regular expression; for example, an asterisk (*) in a data string is not a metacharacter. There are only a few metacharacters in the regular expression language, but their effect can be powerful.

IMPORTANT: The regular expression implementation in Foglight for Java EE Technologies and Foglight for Microsoft .NET differs from that used in other Foglight components (such as Foglight for Infrastructure). For example, the {N,M} syntax is not supported, and Advanced Regular Expression features such as bounds are not included.

The following table lists the metacharacters in Cartridge for JMX and Foglight for Microsoft .NET, and describes their function.

Table 86. Metacharacters and their function.

Metacharacter	Description and Example
asterisk (*) alias: star	The metacharacter that matches zero or more repetitions of the literal character or sub-expression it follows. For example: the regular expression `'White *space'` matches `'Whitespace'`, or `'White space'`, or `'White@@any number of space characters@@space'`, but not `'White *space'` (the string containing a literal asterisk). **NOTE:** `'nexus*'` does NOT match `'nexus.bat'`, or `'nexusconfig'`. Path name-style matching (file globbing) is not the same as regular expression syntax.
period (.) alias: dot	The period is a metacharacter that matches any single character or sub-expression unless it is enclosed in a character class, in which case it is simply a period. For example: • The regular expression `'HTTP1[.]0'` matches `HTTP1.0`. Placing the period in a character class removes its metacharacter status. • The regular expression `'HTTP1.0'` matches `'HTTP1.0'` (the string containing the literal period), or `'HTTP1A0'` or `'HTTP110'`, but not `'HTTP1.x` or `'HTTP1..x'`. The combination `'.*'` (dot-star) matches any string of characters. It is an idiom for 'the rest of the characters' in a string, but not including the end of line character. For example: the regular expression `'HTTP.*'` matches `'HTTP1.0'`, or `'HTTP1://www.example.com/index.html'`. **CAUTION: The regular expression `'mayb.com.*'` matches `'maybecompletely wrong!'`. The dot in the regular expression matches any character, in this case an `'e'`.** **NOTE:** Because the dot is a metacharacter, use the character class `[.]` to match a literal dot in a string. It is more robust than attempting to protect the dot with a backslash `(\.)`.
question mark (?)	The question mark metacharacter matches zero or one occurrence of the character or sub-expression it follows. For example: the regular expression `'i?www'` matches `'www'` or `'iwww'`. The regular expression `'(ab)?cd'` matches `'abcd'` and `'cd'`.

Table 86. Metacharacters and their function.

Metacharacter	Description and Example
plus (+)	The plus sign metacharacter matches one or more occurrences of the character or sub-expression it follows. For example: the regular expression `'i+www'` matches `'iwww'` or `'iiwww'`, but not `'www'`.
square brackets ([])	A pair of bracket metacharacters encloses a character class. Any character in the class is a suitable character for the match. For example: the regular expression `'[bcp]at'` matches `'bat'`, `'cat'`, or `'pat'`, but not `'mat'`. **NOTE:** The characters that are considered metacharacters are different inside character classes. The function of these characters is also different.
minus (-) alias: dash	The minus sign is a metacharacter only if it occurs between two characters in a character class, in which case it represents a range of characters. For example: the regular expression `'[b-h]at'` matches `'bat'`, or `'cat'`, or `'fat'`, or `'hat'`, but not `'mat'`.
caret (^)	Unless it appears as the first entry in a bracketed character class, the caret metacharacter matches the special character that represents the start of a string. The combination `'[^...]'` represents a negated character class. All characters *except* the ones listed are allowed in the match. For example: • The regular expression `'^abc'` matches `'abcxyz'` but not `'xyzabc'`. • The regular expression `'[^cpt]at'` matches `'hat'`, or `'mat'`, or any three-letter combination ending in `'at'` that does not begin with the letters `'c'`, `'p'`, or `'b'`.
dollar sign ($)	Unless it appears as the first entry in a bracketed character class, the dollar metacharacter matches the special character that represents the end of a string. **NOTE:** The dollar sign ($) is a shell metacharacter in Unix. If you are running Cartridge for JMX on a UNIX® system and want to include the dollar sign as a metacharacter in a regular expression, surround the expression with single quotes. For example: `sh nexusctl.sh start-recording -fir '\.[Gg][Ii][Ff]($|\?)/'`
vertical bar (|) aliases: or, bar	The vertical bar metacharacter matches either of the expressions it separates. For example: the regular expression `'foo|bar'` matches `'foo'` or `'bar'`, but not `'foobar'`.
backslash (\\) (reverse solidus)	This metacharacter enforces the literal interpretation of the character following it. It has no effect when placed before an ordinary character. When placed before a metacharacter, it cancels out the metacharacter status and reduces the character to its literal meaning. For example: the regular expression `'myhost\.com'` matches `'myhost.com'`, but not `'myhostcom'`.
parentheses (())	Parentheses are used for group, for example, to limit the scope of the `'or'` operator. For example: the regular expression `'http://(my|our)host\.com'` matches `'http://myhost.com'` or `'http://ourhost.com'`.

CAUTION: Do not use parenthesized expressions followed by a plus sign (+) or an asterisk (*) in any of the configuration files for Foglight for Java EE Technologies and Foglight for Microsoft .NET.

How regular expressions are scanned

When ambiguities are possible, use these rules to decide how regular expressions do their matching.

The rules for applying regular expressions to target strings are as follows:

- The first match is chosen

  If a regular expression could possibly match two different parts of an input string, it matches the one that begins earliest. Example: if the regular expression is `'a.*t'` and the target string is `'cataract'`, the match succeeds with everything after the `'c'`.
- The left-most choice in a parenthesized group wins

  If a regular expression contains `"|"` operators, the left most matching sub-expression is chosen. Example: if the regular expression is `'(cat|dog)'` and the target string is `'cats and dogs'`, the match succeeds with `'cat'` after the first three letters have been scanned.
- The longest match wins

  In `"*", "+"`, and `"?"` constructs, longer matches are chosen in preference to shorter ones. Example: if the regular expression is `'^He's ba*'` and the target string is `'He's baaaaack!'`, the match succeeds with `'He's baaaaa'` after the full list of contiguous a's have been scanned.
- In sequences of expression components, the components are considered from left to right

Typical regular expression patterns

The caret and dollar represent the start and end, respectively, of a string. They do not represent characters, but simply the abstract notion of begin and end. They are useful as anchors for the rest of the regular expression.

Table 87. Regular expression match patterns

Regular Expression	Matches
`^www[.]quest[.]com$`	www.quest.com but not http://www.quest.com or www.quest.com/index.html
`^com[.]quest[.].*`	com.quest.performasure, but not homedir.com.quest.performasure
rmi$	java.rmi, but not java.rmi.activation

NOTE: The meaning of the caret is different if it is the first entry in a character class. In this case, it means 'anything but' the characters following it. For example, [^abc] in a regular expression means that some single character must be matched at the position of the character class within the regular expression, but it can't be 'a' or 'b' or 'c'.

If a caret (^) appears anywhere else but the first position in a character class, it simply stands for itself. The same is true for a dollar sign ($) in a character class: it is just a dollar sign, not a metacharacter.

Matching with character classes

Using character classes provides a range of possibilities and enables you to give a specific list. The first example in the table shows the use of a regular expression to match the American or British spelling of the word 'analyze.'

Table 88. Regular Expression match patterns with character classes

Regular Expression	Matches
`Analy[sz]e this`	Analyze this, and Analyse this
`www[0-9][0-9]?[.]example`	www0.example, www01.example, but not www001.example The regular expression matches ‘www’ followed by a numeral, followed by no more than one more numeral, followed by ‘.example’. Thus, 001 fails the match.
`www[^a-zA-Z]?\.example`	www.example, www1.example, but not wwwx.example

`www[.]quest[.]com`
`www.quest.com`
	This is the recommended way of matching the dot separator in Java package names and fully qualified names.

! **CAUTION: The dash (-) is a metacharacter only when it occurs between two characters in a character class. Otherwise, it is just an ordinary character. The exception to the rule of interpreting a dash as a range operator when it is between two characters is the following: the regular expression [^-*] means ‘exclude dashes and asterisks from a match.’**

Matching any character: the dot

You use the dot (.) metacharacter when you want to indicate that any character is allowed at this position. Combined with parentheses, dot metacharacters can be used to indicate the number of arbitrary characters to be allowed after another pattern. See the second entry in the table below for an example of this case. The first entry in the table is an illustration of the commonly seen ‘dot-star’ combination, which matches any number of arbitrary characters. Also, it’s important to remember that dots are not metacharacters when they occur within a character class.

! **CAUTION: Because the dot is a metacharacter, real periods in package names and URLs must be enclosed in square brackets or preceded by backslashes when they are part of a regular expression.**

Table 89. Regular expression match patterns with the dot

Regular Expression	Matches
`com\.quest\..*`	com.quest.the rest of the string, commas, spaces, periods, and all. As long as the test string contains ‘com.quest.’ followed by any number of additional characters, the match succeeds.
`[ ]co(.|..|...|....)[ ]`	The ‘words’ cod, coal, codes, collar, and even co13 The character classes at both ends of the regular expression contain the space character, which limits the match to space-separated words. The matched word must begin with ‘co’, followed by one, two, three, or four characters.
`594[-./]1026`	594-1026, 594.1026, 594/1026 A dot inside a character class is not a metacharacter.

! **CAUTION: A forward slash (/) marks regular expression boundaries in the configuration files for Foglight for Java EE Technologies. It is used to start and end regular expressions. You can escape with a backslash (\) if you are using a forward slash in a regular expression in a Foglight for Java EE Technologies configuration file.**

The forward slash is a regular character when it is used outside of the configuration files for Foglight for Java EE Technologies. A dot in a character class is not a metacharacter. A dash is not a metacharacter if it is the first or last character in a character class, or if it follows a leading caret.

Alternation in regular expressions

Use the 'or' (|) metacharacter to combine several possibilities into a single expression. Usually, the 'or' metacharacter is used with parentheses to provide grouping.

Table 90. Regular expression match patterns with alternation.

Regular Expression	Matches
`(java|javax)`	This is an explicit equivalent to javax? java, javax.
`[(s|t)]`	(, or s, or |, or t, or) None of these characters are metacharacters when they are part of a character class.
`com\.(sun|quest|klg)`	com.sun, com.quest, com.klg

About Us

We are more than just a name

We are on a quest to make your information technology work harder for you. That is why we build community-driven software solutions that help you spend less time on IT administration and more time on business innovation. We help you modernize your data center, get you to the cloud quicker and provide the expertise, security and accessibility you need to grow your data-driven business. Combined with Quest's invitation to the global community to be a part of its innovation, and our firm commitment to ensuring customer satisfaction, we continue to deliver solutions that have a real impact on our customers today and leave a legacy we are proud of. We are challenging the status quo by transforming into a new software company. And as your partner, we work tirelessly to make sure your information technology is designed for you and by you. This is our mission, and we are in this together. Welcome to a new Quest. You are invited to Join the Innovation™.

Our brand, our vision. Together.

Our logo reflects our story: innovation, community and support. An important part of this story begins with the letter Q. It is a perfect circle, representing our commitment to technological precision and strength. The space in the Q itself symbolizes our need to add the missing piece—you—to the community, to the new Quest.

Contacting Quest

For sales or other inquiries, visit https://www.quest.com/company/contact-us.aspx or call +1-949-754-8000.

Technical support resources

Technical support is available to Quest customers with a valid maintenance contract and customers who have trial versions. You can access the Quest Support Portal at https://support.quest.com.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request.
- View Knowledge Base articles.
- Sign up for product notifications.
- Download software and technical documentation.
- View how-to-videos.
- Engage in community discussions.
- Chat with support engineers online.
- View services to assist you with your product.