Quest™

Foglight® 6.0.0

Performance Tuning Field Guide

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept.
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our website (https://www.quest.com) for regional and international office information.

Patents

Quest Software is proud of our advanced technology. Patents and pending patents may apply to this product. For the most current information about applicable patents for this product, please visit our website at https://www.quest.com/legal.

Trademarks

owners.

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.

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

Foglight Performance Tuning Field Guide
Updated - May 2021
Software Version - 6.0.0

Overview **7**
Characteristics of Poor Performance in Foglight 7
Parameters that Affect Foglight Performance 7
Critical Areas that Can Affect Foglight Performance 8
Identifying Performance Problems 8

Hardware and Operating System Tuning **9**
Running on Virtual Hardware 10
Performance Challenges 10
CPU 11
Memory 11
Shares 12
Disk 12

Management Server Tuning **13**
Dashboard Default Timeout 14

Java Virtual Machine Tuning **16**
Getting Started 16
Foglight JVM Configuration 16
JVM Options 17
Common Symptoms and Tuning Resolutions 19
JVM Code Cache 20

Backend Database Tuning **22**
Initial Database Configuration Settings 22
Monitoring and Managing Database Size 22
Backup and Recovery Recommendations 22
init.ora Configuration 22
Database Maintenance Recommendations 23
MySQL Tuning 23
Case 23
Oracle Tuning 24
How the Management Server Uses the Database 24
Index Management 24
Memory 25
Block Size 25
Oracle Striping 25
Oracle Tablespaces 25
Microsoft SQL Server Tuning 28
PostgreSQL Tuning 28
Troubleshooting 29
Management Server Load Preventing Database Connections 29

High Availability (HA) Tuning **32**

Tuning Connection Issues in an HA Implementation . . . 32
Managing Hosts with Multiple Network Interfaces . . . 33
JDK with IPv6 on Linux . . . 34
Management Server Automatically Restarted . . . 34
Other JGroup Related Issues and Information . . . 34

Agent Tuning . . . 35
Topology Changes and Topology Churn . . . 35
Canonical Data Transformations (CDTs) . . . 36
Agent Weight / Environment Complexity . . . 36
Large Topologies . . . 37
Sampling Frequency . . . 37
XML-HTTP Agent Adapter . . . 37
Dropped Agent Manager Log Messages . . . 38
Java EE Technologies . . . 38

Store and Forward Processing . . . 39
About Store and Forward Processing in Foglight . . . 39
Store and Forward Processing and the Foglight Management Server . . . 39
Store and Forward Processing and the Foglight Agent Manager . . . 40
Viewing the Settings that Control the Size of Stored Data . . . 40
Examples . . . 41
Example 1: Reporting on Disk Usage . . . 42
Example 2: Calculating Service Availability Metrics . . . 42

Appendix: Performance Health Check . . . 43
Is the Server Getting the Right Amount of Memory? . . . 43
Management Server Memory is Healthy if 43
Management Server Memory Checks . . . 44
Possible Actions . . . 44
Is the Server Getting Enough CPU? . . . 44
Management Server CPU is Healthy if 44
Management Server CPU Checks . . . 44
Extended Browser Interface Checks . . . 45
Possible Actions . . . 45
Is the Database too Slow? . . . 45
The Database is Healthy if 45
Database Checks . . . 46
Possible Actions . . . 46
Is the Database Growth Reasonable? . . . 46
Database Growth is Healthy if 47
Database Checks . . . 47
Possible Actions . . . 47
Is There Too Much Data to Process? . . . 47
Data Volume is Healthy if 47
Data Volume Checks . . . 47
Possible Actions . . . 47

Is the Model Stable? . . . 48
The Model is Healthy if 48
Model Checks . . . 48
Possible Actions . . . 48
Are Too Many Alarms Firing? . . . 48
The Number of Alarms is in a Healthy State if... . . . 48
Alarms Checks . . . 49
Possible Actions . . . 49
Is the Business Logic Properly Tuned? . . . 49
Business Logic is Healthy if... . . . 49
Business Logic Checks . . . 49
Possible Actions . . . 49
Are There Too Many User Requests? . . . 50
User Activity Is Healthy If... . . . 50
Possible Actions . . . 50

Appendix: Analyzing a Support Bundle . . . 51
Server Log . . . 51
Topology Sync . . . 51
Topology Limits . . . 51
Diagnostic Snapshot . . . 52
Memory Consumption . . . 52
Threads, Deadlocks, and Overall CPU Usage . . . 52
Useful Information . . . 52
---- jboss.system:type=ServerInfo . . . 52
---- jboss.jca:service=ManagedConnectionPool,name=jdbc/nitrogen . . . 53
---- jboss.web:type=RequestProcessor,* . . . 53
---- com.quest.nitro:service=Derivation . . . 53
---- com.quest.nitro:service=Topology . . . 54
---- com.quest.nitro:service=DataCacheEviction . . . 55
Analyzing a Performance Report . . . 56
Server Rule Information . . . 56
System-wide Topology Changes . . . 57
JVM Memory Usage . . . 57
Management Server Garbage Collectors . . . 59
JDBC Connection Pool . . . 60
Derivation Rulette . . . 61

About Us . . . 62
We are more than just a name . . . 62
Our brand, our vision. Together. . . . 62
Contacting Quest . . . 62
Technical support resources . . . 62

1

Overview

Performance is a measure of the efficiency of an application running in an environment or of the overall efficiency of multiple applications running in the same environment.

You tune performance in order to optimize it.

- Characteristics of Poor Performance in Foglight
- Parameters that Affect Foglight Performance
- Critical Areas that Can Affect Foglight Performance
- Identifying Performance Problems

Characteristics of Poor Performance in Foglight

The following items characterize poor performance in the Foglight® environment:

- poor performing (slow) browser interface
- missing data
- high CPU load for the Foglight Management Server process
- high CPU load for the database backend process
- excessive memory utilization (or out of memory)
- long-running SQL statements

Parameters that Affect Foglight Performance

Foglight® is a complicated system, and its runtime performance depends on many variables within the environment.

The following parameters have an effect on Foglight performance:

- the processing power of the host running the Management Server
- the memory of the host running the Management Server
- the memory allocated for the Management Server process (heap)
- the database backend type (MySQL™/Oracle®)
- the processing power of the host running the database backend
- the memory of the host running the database backend

- the database setup
- the number of Foglight Agent Managers

Foglight's runtime performance also depends on what it is monitoring.

Critical Areas that Can Affect Foglight Performance

It can be difficult to determine that Foglight® performance is suffering and then identify the appropriate steps to take. This guide provides information about the various things that can have an effect on overall performance and describes the applicable performance-related options.

The areas that can have an effect on overall performance are:

- Hardware and Operating System Tuning
- Management Server Tuning
- Java Virtual Machine Tuning
- Backend Database Tuning
- High Availability (HA) Tuning
- Agent Tuning

Identifying Performance Problems

There are two main methods you can use to proactively check the performance of a Foglight® installation:

1. You can carry out a performance health check to quickly verify whether or not Foglight is functioning properly. For more information, see Appendix: Performance Health Check on page 43.
2. You can analyze a support bundle to identify problems. For more information, see Appendix: Analyzing a Support Bundle on page 51.

2

Hardware and Operating System Tuning

Foglight® processes and analyzes observations and presents them to users. This is resource intensive. Even if you perform the tuning exercises described in other sections of this guide, the overall resource use of the Management Server might still eventually exceed that which the underlying operating system and hardware can support. However, you can increase Foglight performance linearly by putting Foglight on faster hardware (that is, more CPU and increased I/O) or by increasing the physical memory available (see Java Virtual Machine Tuning on page 16).

For the minimum Management Server hardware requirements, refer to the *System Requirements and Platform Support Guide*.

- Running on Virtual Hardware

Table 1. Physical Memory

Motivation	When you run multiple processes (for example, the Management Server and embedded database) on the same machine, it is important that there is enough physical memory available to satisfy the Heap settings (+JVM/native thread stack overhead) and accommodate any other processes (for example, the MySQL memory cache).
Symptom	The browser interface performance is poor (that is, it responds slowly).
Diagnosis / Verification	The diagnostic snapshot for the Management Server shows that it is overloaded. The OS page-hit percentage is low.
Tuning	Increase the available physical memory to accommodate the heap settings of the Management Server and overhead.

Table 2. CPU

Motivation	Foglight relies on highly efficient threading to achieve its throughput in handling data and serving content to the user. The system requirements state minimums for number of CPUs and cores that need to be guaranteed. Running other applications creates competition for CPU and therefore has a negative affect on Management Server performance.
Symptom	The browser interface performance is poor (that is, it responds slowly).
Diagnosis / Verification	CPU Utilization is high.
Tuning	Move the Management Server to a better performing machine or reduce overall load of host by shutting down other applications.

Table 3. I/O Throughput

Motivation	Foglight constantly receives, stores, and retrieves data. It is an I/O-intensive system. Therefore, I/O can be a limiting factor.
Symptom	The browser interface performance is poor (that is, it responds slowly).

Table 3. I/O Throughput

Diagnosis / Verification	The diagnostic snapshot for the Management Server shows that it is overloaded. The snapshot also shows that threads are waiting on I/O. The operating system statistics show high I/O utilization rates.
Tuning	Deploy faster storage hardware (network, drives, and so on) or implement striping for multiple data destinations. If the I/O throughput for the system seems to be below industry standards, check for I/O chain misconfiguration: ensure the firmware is up to date, validate the SAN (storage area network) routing configuration, and so on. Reduce the latency between the database and the Management Server.

Table 4. File Handle Limits

Motivation	Foglight requires a minimum number of file handles. The native launcher checks for a minimum number of file handles, but problems can still occur.
Symptom	The JVM reports a `java.lang.OutOfMemoryError`.
Diagnosis / Verification	Determine if Foglight has run out of native file handles for a process because the operating system has a maximum number of file handles per process defined.

Tuning	Check the current values:
	`[root@stnuat]/# ulimit -Sa`
	`...`
	`nofiles(descriptors) 1024`
	`...`
	`and`
	`[root@stnuat]/# ulimit -Ha`
	`...`
	`nofiles(descriptors) 65536`
	`...`
	If the per-process file handle maximum is too low for the application, try raising it.
	The values can be set before starting the server (to the maximum, for example) using:
	`ulimit -n 65536`

Running on Virtual Hardware

The Foglight® Management Server can be run on a virtual machine (VM), instead of physical hardware.

If you are going to run the Management Server on a VM, ensure that the Management Server is well-tuned.

Performance Challenges

The following subsections describe some of the performance challenges the Foglight® Management Server might face in an environment that has interacting virtual components.

The information in the following subsections helps you optimize the performance of the VM in which the Management Server is running. However, as the performance demands of monitoring the virtual infrastructure increases, so do the underlying processing requirements.

CPU

Before you run the Foglight® Management Server in a virtual image, you must make sure that the planned number of processors will be made available. In certain scenarios, the virtual image will not exist exactly as planned.

To confirm how many processors the virtual image, and therefore the Management Server, will receive, generate a support bundle and check the Diagnostic Snapshot. In the Diagnostic Snapshot, look for AvailableProcessors.

Ready Time

It is important to monitor the Percent Ready value for each VM. A Percent Ready value in excess of zero (0) indicates that, of the amount of time the process was ready to run, for that percentage of the time it could not because it was waiting for resources. A Percent Ready value under two percent (2%) is considered good; values in excess of 2% can pose a challenge.

When a Percent Ready value is in excess of 2% for a few seconds or longer, the application might become jerky and appear to pause. This is why the responsiveness of an application might deteriorate after a certain number of users is reached.

Since ready time is the time a process spends waiting when it could be running, it has a direct impact on response time.

For example: A number of virtual machines are pinned to a single CPU and 600 seconds of CPU time are available. Four of the virtual machines have load and about 300 seconds of ready time has accumulated, so you can expect a slowdown of approximately 50 percent. An event with a response time of 1 second could be expected to take 1.5 seconds under that level of load as a direct result of the delays caused by the ready time.

This is not atypical for observations of multiuser systems under load. The same factors apply to an ESX® Server host running multiple virtual machines. Also, the more virtual CPUs the VM has, the more CPUs have to be free, or the VM may have to wait longer for *x* number of CPUs to be free.

Memory

Before you run the Foglight® Management Server in a virtual image, make sure that the image is configured with the necessary amount of memory.

The Management Server is very active and consistently uses a significant portion of the allocated memory, so ensure that the memory you allocate to the virtual image is committed. For Management Server performance, it is far better to assign the appropriate amount of memory and configure it to be fixed than to assign more memory and configure it to be dynamic.

Also, it is often the case that plenty memory has been allocated to the VM in question, but the memory limit has been left at a level that is too low for the VM to obtain the memory it needs.

To confirm how much memory will be allocated to the virtual image, and therefore to the Management Server, generate a support bundle and check the Diagnostic Snapshot.

Diagnostic Snapshot

```
---- jboss.system:type=ServerInfo
   MBean attributes:
   attr ActiveThreadCount = 390
   attr AvailableProcessors = 1
   attr OSArch = amd64
   attr MaxMemory = 9556328448
   attr HostAddress = 10.4.52.60
   attr JavaVersion = 1.6.0_04
   attr OSVersion = 2.6.9-42.EL
   attr JavaVendor = Sun Microsystems Inc.
   attr TotalMemory = 9556328448
```

```
attr ActiveThreadGroupCount = 21
attr OSName = Linux
attr FreeMemory = 6509179336
attr HostName = host25.example.corp
attr JavaVMVersion = 10.0-b19
attr JavaVMVendor = Sun Microsystems Inc.
attr JavaVMName = Java HotSpot(TM) 64-Bit Server VM
```

Review the MaxMemory, TotalMemory, and FreeMemory entries. For more information about the diagnostic snapshot, see Diagnostic Snapshot on page 52.

Shares

For the Foglight® Management Server to run in a virtual image, share allocations must provide the Management Server with sufficient priority over the other VMs to ensure that the Management Server can process incoming data and browser interface requests in a timely manner.

Disk

Disk size is fixed at the time of virtual image creation, so ensure beforehand that enough disk is allocated to the image (as per the platform-sizing guideline).

In a VMware® environment, disk (in the form of LUNs) is allocated to a Data Center, ESX® Servers, and VMs. The VMs from many ESX Servers can reside on the same LUN, which can also be shared by many ESX Servers. Therefore, the disk activity in a VM on an ESX Server can have a serious adverse impact on the performance of the VM in which the Foglight® Management Server is running, even when the VMs are running on separate ESX Servers—a situation that cannot arise when the Management Server is running on physical hardware.

3

Management Server Tuning

The parts of the Foglight® Management Server that have the greatest influence on runtime performance are topology (management and querying), observations (conversion, storage, and retrieval), and alarms and alarm processing (derivations and rules). To provide these components and the associated calculations, an architecture that includes queues, thread-pools, caches, and so on, is required.

The Management Server architecture can be tuned in the following ways.

- Alarm Limit
- Topology Limit
- Number of CPUs Used for Thread Pools
- Dashboard Default Timeout

Table 1. Alarm Limit

Motivation	A lot of the information presented in the browser interface is derived from alarms (alarm counts, topology object states, and so on). If rules are configured poorly, this can result in the creation of a very large number of alarms within the system. The server imposes a default limit of 10000 alarms that can be displayed and used to calculate object states. When the server enforces this limit, it gives preference to the current alarms so that objects are presented in the correct state.
Symptom	The browser interface performance is poor (that is, it responds slowly), especially when displaying the alarms table.
Diagnosis / Verification	The server is not overloaded, and is still processing data. CPU usage on the server is high when it is providing alarm information. Threads may become blocked for several seconds, intermittently, when attempting to load alarm details.
Tuning	The alarm limit may need to be reduced. MBean: `*:service=Alarm` Attribute: `MaxAlarms` Expected old value: `10000` New value: `5000`
Note 1	You can set this parameter using the `foglight.alarm.query.max_alarms` Java™ system property. For example in *server.config*, add: `server.vm.option0="-Dfoglight.alarm.query.max_alarms=5000";`
Note 2	If the alarm limit is reduced to the point where the server is not able to load all of the current alarms into the system, then some topology objects may be displayed with an incorrect state.

Table 2. Topology Limit

Motivation	The amount of processing that a server has to perform is often proportional to the number of topology objects (for example, rulettes) in the system. Cartridges that monitor very large systems may end up creating enough topology objects to bring the server into an overload situation. To protect against this, the topology service is configured to limit the number of instances of each object type that can be created.
Symptom	The browser interface performance is poor (that is, it responds slowly). The server memory usage is high. The server may be overloaded.
Diagnosis / Verification	Examine the topology instance counts shown in a support bundle.
Tuning	If there is an object type with an excessive number of effective topology objects, then some of those objects may need to be deleted. Try to determine why the cartridge created so many objects of that type. Ideally, you should modify the cartridge configuration to prevent it from recreating those objects. As a precaution, you can reduce the topology limit for the object type. The limit for a type can be set using the `foglight.limit.instances` registry variable.
Note	When the limit for an object type is reached, messages appear in the server log and an alarm is raised in the browser interface. If it is reasonable for the object type in question to have a large number of instances, then the limit for that type should be increased to prevent the error messages from being generated. If it is not reasonable for a type to have a large number of instances, you should tune your agent so that it does not create so many of them.

Table 3. Number of CPUs Used for Thread Pools

Motivation	The Management Server utilizes a scaling algorithm to calculate the number of threads in its thread pools. This algorithm takes into account number of CPUs available on the server. However, in certain cases where the Management Server shares the server with other applications, it is desirable to limit the number of CPUs the Management Server takes into account. Please note that this setting does not limit the actual CPUs the Management Server will use.
Note	You can set this parameter using the `foglight.threadpool.cpu.count` Java™ system property. For example in *server.config*, add: `server.vm.option0="-Dfoglight.threadpool.cpu.count=4";`

Dashboard Default Timeout

If your dashboards are timing out, try increasing the default timeout value:

To increase the console default timeout value:

1. Open the *<foglight_home>/server/default/deploy-foglight/console.war/scripts/* directory.
2. Locate the following four files:
   a. *wcf-ie.js*
   b. *wcf-ipad.js*
   c. *wcf-standard.js*
   d. *util.js*
3. Search for `DEFAULT_TIMEOUT`.

4 Increase the value of `DEFAULT_TIMEOUT` to `180000`. This increases the console default timeout to 3 minutes. (The number is based on milliseconds; 1000 is one second.)

5 Save the files, being careful not to change the file format.

6 Refresh (press F5) the page in your Web browser.

4

Java Virtual Machine Tuning

Before you proceed with Java™ Virtual Machine tuning, it is important to note the following:

- Foglight® 5 is a Java application that ships with and uses the Oracle® Java Hotspot Virtual Machine (JVM). While there are other vendors of JVMs, the Oracle JVM should not be replaced or altered.

  NOTE: Quest acknowledges that under certain circumstances you may have to use a different Java Runtime Environment than either of the ones (one for the Management Server and one for the Agent Manager) bundled with Foglight. If that is the case, see the “Updating the Java Runtime Environment” section in Chapter 2 (“Best Practices”) of the Foglight Upgrade Guide.

- Foglight is designed to run on a wide range of platforms and configurations, and is very scalable. It can be used to monitor several computers/nodes within a department and to monitor an entire enterprise with thousands of nodes.

  NOTE: Some out-of-the-box settings may not be optimal for every scenario.

This section provides information on how to tune the JVM to achieve maximum performance.

- Getting Started
- Foglight JVM Configuration
- Common Symptoms and Tuning Resolutions

Getting Started

The following factors heavily impact JVM utilization:

- the number of monitored computers/nodes
- the number of observed metrics (such as CPU load, memory utilization, I/O throughput, and so on) for each node
- the length of the time period for which Foglight® is configured to preserve historical data about observations
- the granularity of the observations over that time period
- the number of alarms in the system
- the platform architecture (32- versus 64-bit address space)
- the processes (services, database, Foglight Agent Managers, and so on) competing for CPU and memory

Foglight JVM Configuration

To review or modify Foglight® JVM configuration options, locate the following file:

`<foglight_home>\config\server.config`

Foglight reads JVM options from values of `server.vm.optionsX` properties, where X is a number from 0 to 99.

For example:

```
server.vm.option0 = "-Xms2048m";
server.vm.option1 = "-Xmx2048m";
```

NOTE: After any change to *server.config*, you must restart the server.

The JVM options currently in effect (the default Foglight JVM options, *server.config* settings, and command line arguments) are all logged in the server log file during startup.

JVM Options

The following table describes the most commonly used Oracle® Hotspot JVM options:

-XX:+UseConcMarkSweepGC

This switch enables a concurrent low-pause garbage collector.

Default

Enabled, since release 5.2.3.

Recommendation

Do not change this setting unless advised by Quest Support.

-Xmx and -Xms

These switches specify the maximum and initial size (respectively), in bytes, of the memory allocation pool (the heap). Each value must be a multiple of 1024 and greater than 512 MB. Append the letter m or M to indicate megabytes, or g or G to indicate gigabytes.

Examples:

```
-Xmx2048m
-Xmx2g
```

Default

32-bit: 256m < 75% avail < 1024m

64-bit: 512m < 75% avail < 4096m*

Recommendation

Increasing memory can improve the server's performance. You should run the server with the default settings first. Close monitoring of Server Load in the Performance Overview dashboard (Foglight|Diagnostic) will reveal any over-utilization or under-utilization of memory:

IF

- the overallLoad plot is moving between 0.8 and 1 (high troughs)
- the memory_usage plot is staying close to totalMemory over time

THEN increase the maximum heap size

IF

- the overallLoad is fluctuating between 0.x and 1 (low troughs)
- the memory_usage plot minimum is clearly below totalMemory

THEN consider decreasing the heap size to a lower value between the highest minimum of the memory_usage plot and the totalMemory to reduce resource allocation

Check the overall memory allocation on the machine hosting Foglight.

- Hard page faults are giving away "over-allocation" (1 GB for the OS + 12 GB for the VM heap + 2 GB for the VM overhead + 1 GB for the Agent Manager and the agents + 2 GB for an embedded MySQL database is more than enough for a 16 GB setup).

-Xms for the Agent Manager

When optimizing Agent Manager startup JVM parameters, the -Xms and -Xmx values do not have to be the same, because the memory usage pattern of the Agent Manager is quite different from that of the Management Server. For the Agent Manager, -Xms should be set to the lowest estimated value of the expected steady state of the Agent Manager when running all agent instances. This -Xms configuration is different from what is recommended for the Management Server.

-XX:NewSize, -XX:MaxNewSize, and -XX:NewRatio

These switches specify the (initial/maximum) size of the young object space where new objects are allocated. For the NewSize switches, append the letter m or M to indicate megabytes, or g or G to indicate gigabytes.

Example:

```
-XX:NewSize=2048m
```

Default

```
-XX:NewRatio=4                          (1:4)
-XX:NewSize=(-Xmx/5)
-XX:MaxNewSize=(-Xmx/5)
```

Recommendation

Foglight® receives and processes a lot of transient data that is best handled in the JVM's new generation part of the heap. If you reserve a minimum size for this type of memory, that guarantees fast throughput with low memory management overhead. The result of a large young space size may be increased garbage collection pause times.

Unless specifically instructed to do so, you should not change this parameter.

-XX:MaxPermSize

This switch specifies the memory allocated to hold the reflective data (such as class objects and method objects) of the VM itself. These reflective objects are allocated directly into the permanent generation, which is sized independently from the other generations.

Example:

```
-XX:MaxPermSize=512m
```

Default

```
96m < -Xmx/4 < 512m*
```

Recommendation

Foglight is a dynamic application platform that loads and manages a lot of deployed dynamic components. The default value has been tested and utilized in the field.

Unless specifically instructed to do so, you should not change this parameter.

-Xss

This switch specifies the thread stack size of the memory allocation pool. The value must be a multiple of 1024. Append the letter k or K to indicate kilobytes, m or M to indicate megabytes, or g or G to indicate gigabytes.

Examples:

```
-Xss=512k
```

```
-Xss=1m
```

Default

```
-Xss=256k
```

Recommendation

The stack size should not be changed unless a specific problem (stack overflow) has been identified by Quest Support.

Common Symptoms and Tuning Resolutions

The following table provides common Foglight® performance issues and tuning that resolves those issues.

Table 1. Common Symptoms and Tuning Resolutions

Symptoms	Diagnosis/Verification	Tuning
• The browser interface is intermittently slow. • Data is missing or incomplete. • The Server dashboard shows large spikes in heap memory utilization. • The server logs show that the server is discarding data during garbage collection.	The JVM is configured to invoke the garbage collector only when heap utilization reaches an upper threshold. Consequently, to operate, the garbage collector must suspend the application. In this particular deployment, it would be suitable to have a concurrent garbage collector.	Check to see if a concurrent garbage collector is enabled. If not, enable one using the `-XX:+UseConcMarkSweepGC` switch. **NOTE:** In Foglight 5.2.3 and later, a concurrent low-pause garbage collector is enabled by default.
• Data is missing or incomplete. • The Server dashboard shows consistently high heap memory utilization (80 percent or more). • The server logs show `java.lang.OutOfMemory errors`.	By default, Foglight acquires 75% of the available physical memory (up to a maximum of 1 GB on a 32-bit operating system). These symptoms usually occur when the size of queued information and cached information combined exceeds the allocated memory size, and therefore the server cannot operate as expected.	Increase the maximum heap size using the `-Xmx` switch, and consider switching to a 64-bit operating system. **NOTE:** Double any assumptions about memory requirements when running on a 64-bit operating system.

Table 1. Common Symptoms and Tuning Resolutions

Symptoms	Diagnosis/Verification	Tuning
• On a 32-bit Windows® operating system, the server logs show java.lang.OutOf Memory errors. • On a 32-bit Windows operating system, the server logs show java.lang.OutOf MemoryError:unable to create new native thread.	By default, Foglight acquires 75% of the available physical memory (up to a maximum of 1 GB on a 32-bit operating system). If too much memory is allocated by the server, then there is not enough address space available for other operations, such as creating a new thread or starting a child process.	Consider reducing the memory allocated by Foglight using the `-Xmx1G` switch. Alternatively, consider reducing the amount of memory allocated per thread using the `-Xss=512k` or `-Xss=256k` switch.
• On a 32-bit Windows operating system, the server fails to start with more than 0.8/1GB of heap, but other Java™ applications start normally. • On a 32-bit Windows operating system, the server output displays a 'Could not reserve enough space for object heap' message.	Foglight's launcher process utilizes Java DLLs, instead of running *java.exe* as *run.bat* does. Internal memory parameters such as 'size of contiguous free memory' or 'memory layout' can lead to related problems and/or limitations.	Customers who plan to have memory requirements that are greater than 800 MB should consider running Foglight on a 64-bit operating system.

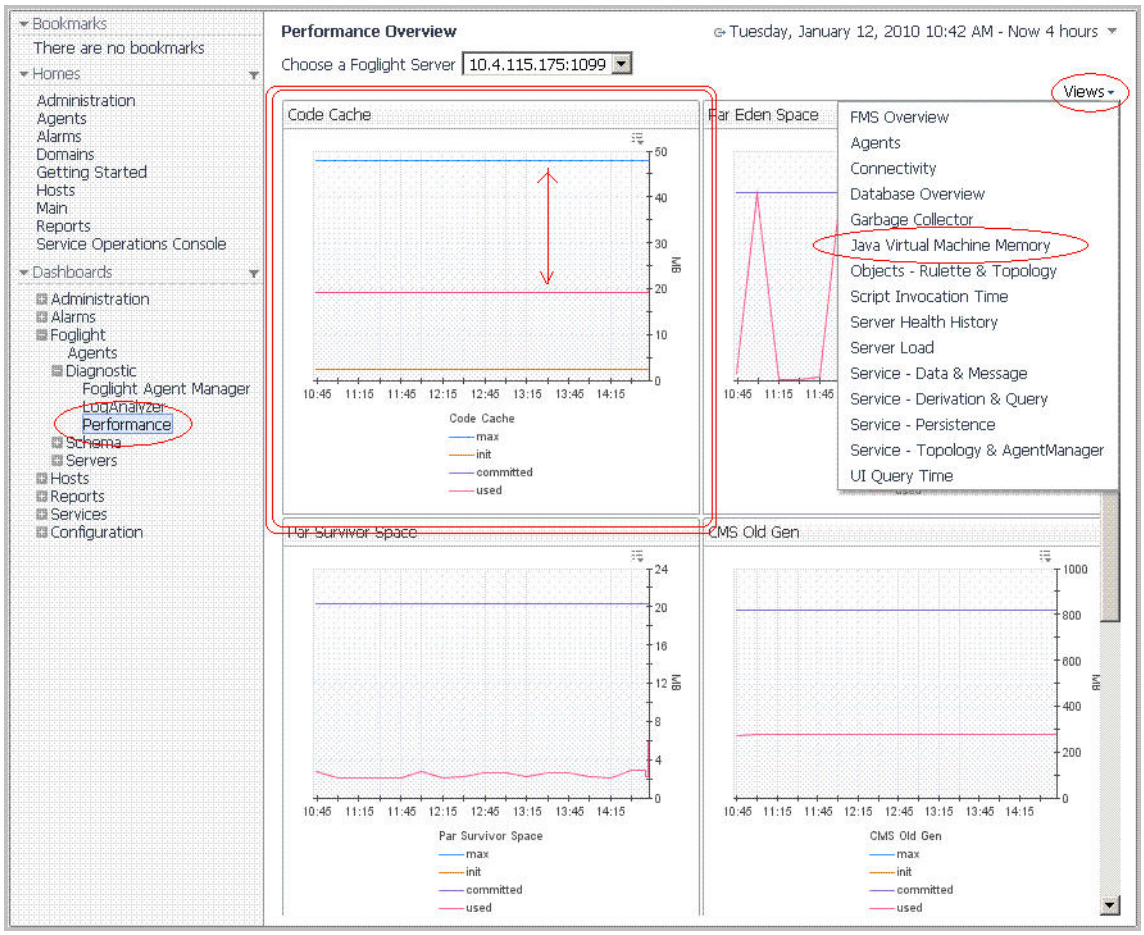
JVM Code Cache

Native code runs faster than the equivalent Java™ code.

The JVM maintains a Java code cache from which native code can be generated. In a large environment running a number of cartridges, this code cache may approach its maximum (though this will taper off eventually). If the code cache is filling up, this results in a decreasing Compilation Time Delta JVM metric.

Use the Performance Overview dashboard (**Dashboards > Foglight > Diagnostic > Performance**) to check the status of the code cache.

Figure 1. Performance Overview dashboard



To optimize code execution in larger environments, you can increase the size of the JVM code cache.

To increase the size of the JVM code cache:

NOTE: The default code cache size has been changed to *256m* since the Foglight Management Server version 5.7.5.1.

1. Open *<foglight_home>/config/server.config*.
2. Locate the `server.vm.option0` line and change the cache size, as needed:

   ```
   server.vm.option0 = "-XX:ReservedCodeCacheSize=512m";
   ```

3. Restart the Management Server.

5

Backend Database Tuning

Foglight® Management Server stores the topology model and observations in the backend database, and it retrieves potentially large amounts of data out of the database when it is asked to by the rules or browser interface. If the database does not perform well, the Management Server slows down as a result. The symptoms of this slow down can vary from general user interface sluggishness to the loss of observations. A properly tuned backend database helps the Management Server run smoothly.

- Initial Database Configuration Settings
- MySQL Tuning
- Oracle Tuning
- Microsoft SQL Server Tuning
- PostgreSQL Tuning
- Troubleshooting

Initial Database Configuration Settings

This section touches on some of the more important database configuration settings. However, your particular database environment should be setup and managed by an experienced database administrator (DBA) who can properly make the decisions necessary when it comes to the database settings involved.

Monitoring and Managing Database Size

For information on monitoring and managing database size, see the “About Database Management in Foglight” section in Chapter 3 (“Setting Up Foglight”) of the Foglight® *Administration and Configuration Help*.

Backup and Recovery Recommendations

For database backup and recovery recommendations, see Chapter 3 (“Setting Up Foglight”) in the Foglight® *Administration and Configuration Help*.

init.ora Configuration

The following are recommended for *init.ora*:

- Put all data files in a separate redundant array of independent disks (RAID), or use Automatic Storage Management (ASM) with enough disks.
- Create at least 7 redo log files and put them in a separate fast RAID (or use ASM with enough disks).
- Use dispatchers to save RAM.

Sample init.ora:

```
db_file_multiblock_read_count = 16
dispatchers = '(protocol=TCP)(disp=4)(con=50)'
job_queue_processes = 10
sga_max_size = 6544M
undo_retention = 3
db_block_size = 8192
open_cursors = 300
pga_aggregate_target = 1599M
processes = 150
sga_target = 4544M
```

Database Maintenance Recommendations

For database maintenance recommendations, see the appropriate sections below.

MySQL Tuning

Table 1. MySQL Tuning

Motivation	MySQL caches data pages in the memory buffer pool. It is important that you have a large enough memory buffer pool for MySQL to have a high cache hit ratio.
Symptom	The browser interface performance is poor (that is, it responds slowly).
Diagnosis / Verification	The diagnostic snapshot for the Management Server shows that it is overloaded.
Tuning	Increase `the innodb_buffer_pool_size` to 1 GB or more (up to 80 percent of the free memory). The default is 512 MB.

For more information on MySQL performance, you can try http://www.mysqlperformanceblog.com and http://www.oracle.com/technetwork/index.html.

Case

Exception: `com.quest.nitro.service.topology.LockTimeoutException: Error obtaining cache lock for object`

Cause: This exception is thrown when a thread that is trying to update a topology object from a Canonical Data Transformation (CDT), times out waiting for a write lock on the object. If it takes a long time for the object to be updated, it is probably because the database is dealing with a heavy load.

If you run your server in debug mode, the server logs thread and lock information, and therefore reports when it encounters such an error. For example, in one case where this exception was thrown, a thread dump was logged approximately a minute later. That thread dump showed that a few threads were inserting topology objects, and that several threads were waiting for an alarms query to finish. It seemed that the database had been allocated a lot of memory, but it was difficult to tell the status of the I/O. Some configurations are configured to flush the transaction logs to disk at the end of every transaction. The standard, or typical, configuration is more relaxed about this. It sets the following in the *my.cnf* file:

```
innodb_flush_log_at_trx_commit=2
```

That setting is beneficial in cases like the one described above.

Oracle Tuning

When tuning an Oracle® backend database, you should first read the information below. You want the information you obtain to be sufficient enough for you to make decisions about the initial setup and configuration of the Oracle backend. Once the backend is operational, a DBA should use Oracle tools to check the parameters of execution and adjust them based on the observed behavior.

How the Management Server Uses the Database

The following are the Foglight® Management Server backend database usage characteristics:

- For the most part, the Management Server performs online transaction processing (OLTP) combined with occasional small-to-medium batch operations.
- The Management Server does not use Database Management System (DBMS) jobs.
- The Management Server configuration tables are mainly infrequent reads/updates (tablespace group A). For more information, see Oracle Tablespaces on page 25.
- The Management Server alarm tables are frequent inserts (alarm_*, tablespace group A). For more information, see Oracle Tablespaces on page 25.
- The Management Server topology tables are less frequent inserts/updates (topology_*, tablespace group A). For more information, see Oracle Tablespaces on page 25.

The following sections elaborate on specific aspects of Oracle® parameters.

Index Management

We recommend you consult Oracle® analysis tools to check the effectiveness of indices used by the Foglight® Management Server. There are no known backend operations through the Management Server that would invalidate specific indices on particular events. Indices should only be rebuilt at the discretion of the DBA, after a thorough analysis of the behavior of the backend at runtime has been performed.

Table 2. Oracle Database Analysis

Motivation	Oracle has the ability to analyze database tables and change its data access/management strategies. • A fresh, 10G or later, installation of Oracle analyzes tables automatically. The database keeps track of the tables that are being modified and analyzes heavily-updated tables more frequently. This frequency has yet to be confirmed. It may be only once per day. • If DBAs disable this functionality and use manually scheduled scripts, then Foglight tables definitely have to be analyzed, especially the large ones like topology, observations, and alarms.
Symptom	It seems that starting with a fresh Management Server installation and attaching a large number of agents almost immediately leads to poor database performance, which persists until statistics have been calculated.

Table 2. Oracle Database Analysis

Diagnosis / Verification	The DBA should use Oracle analysis tools to check for strategy effectiveness.

Tuning	We recommend you run database analysis periodically, if it is not happening automatically (during off hours).
	Calls:
	`exec dbms_utility.analyze_schema('FOGLIGHT', 'COMPUTE');`
	or
	`exec dbms_utility.analyze_schema('FOGLIGHT', 'ESTIMATE'); //`
	
	Note: Estimate is likely to be quicker than Compute, but it is not as thorough.

Memory

At the moment, there is no direct correlation between the amount of memory required by the Foglight® Management Server and the amount of memory required for the database. The management tools that ship with Oracle® can provide some tuning information to help you understand how much memory the database requires at load time.

Typically, we would expect that for a Management Server heap of 1 GB, the database System Global Area (SGA) should be between 1 and 2 GB. This should be set up at the DBA's discretion.

Block Size

At this time, we are not aware of any specific effect caused by running with different block sizes. 8 K blocks, in particular, should be fine. Due to the way in which the database is primarily used, larger block sizes are not expected to perform better. For information on how the database is primarily used, see How the Management Server Uses the Database on page 24.

Oracle Striping

Oracle® can utilize striping to increase the throughput of the database. A responsible Oracle DBA will analyze the I/O numbers of our sizing spreadsheet and then set up striping if they feel that faster disk access is warranted. In testing, we have not seen any need for striping for the supported load numbers. Use striping at your own discretion.

Oracle Tablespaces

You can improve database performance by using multiple tablespaces that are located on different physical hard disks. This setup can yield much better I/O throughput than a setup with a single tablespace on one disk.

Foglight® Management Server basically manages two sets of tables:

- Group A for configuration tables, topology information, and alarms.
- Group B for observations (the operational data that is continuously arriving).

NOTE: There is no benefit to separating configuration data from observation data (that is, in setting up multiple destinations for the different types of data, potentially on different physical disks).

Quest does not recommend any tablespace separation for data, index, and LOB. Quest does not expect such separation to have any impact on real performance. Your DBA can decide how to configure such things based on his or her own specific requirements about where to store data, index and LOB. This is typically done according to database administration conventions.

During the installation, you can select a tablespace for group A (advanced database setup, by default USERS). The configuration tables are mostly read, and are not modified (inserts/updates) very frequently because configuration changes are not happening all the time.

The topology tables (topology_*) in the same tablespace will see more activity (inserts for the most part, but also a fair amount of updates), depending on the agent data. Changes should only occur periodically, as changes in the monitored environment trickle into the Management Server.

The alarm tables (alarm_*) will, for the most part, receive a constant stream of alarms (inserts) along with the occasional time-based read as the user looks at alarm history. Features like alarm annotations do not cause frequent updates.

For group B, you can specify a list of tablespaces at install time (advanced database setup, by default USERS). The observation tables are used for insertion only. The server creates new tables (obs_*) at runtime when it needs to store data. When the data in a table is no longer required, the table truncates and is then used for another set of data.

The configuration of multiple tablespaces enables the Management Server to spread the observation tables over multiple hard drives (if configured that way) and, generally, to achieve higher throughput for incoming data, retrieval, and rollup operations. At present, we are not aware of a bottleneck in Oracle® performance with the supported load numbers and known hardware. For higher supported load numbers, and, depending on the hardware, this setup should be considered by the DBA.

For more information, consult Oracle's documentation on tablespaces.

Undo tablespace

With Oracle9i, Oracle introduced the UNDO_RETENTION parameter, which specifies how far back in time Oracle retains undo information. Oracle8i made use of rollback segments instead. With Oracle8i, you had to have a rollback segment large enough (as suggested by your DBA) to be able to complete the longest possible transaction, and you had to make sure you had enough rollback segments to process several concurrent transactions. When a transaction is finished, a rollback segment can be reused for another transaction.

With Oracle9i and beyond, the system should be able to accomplish what it needs to without you having to add rollback segments, and it should be able to go as far back in time as the UNDO_RETENTION parameter specifies. When UNDO_RETENTION is equal to 900 (the default value), the undo retention time is 15 minutes. If the database is transaction-intensive (as is the case with Foglight®), Oracle9i creates a lot of rollback segments for each transaction on the undo table space. When the transaction is finished, it keeps the rollback segments to satisfy the UNDO_RETENTION value. To be able to serve new transactions, it creates new rollback segments. If the autoextend parameter is on, the UNDO_TABLESPACE keeps growing. In comparison with Oracle8i, if UNDO_RETENTION=20, the Oracle9i UNDO_TABLESPACE size has to be 20 times more than all rollback segments in Oracle8i, given the same transaction rate. It is possible to have a fixed UNDO_TABLESPACE size, like in Oracle8i. In that case, it has to initially be large enough to be able to process all transactions, and, at the same time, keep all used rollback segments to satisfy the UNDO_RETENTION. So, in our case, we need to reduce UNDO_RETENTION and estimate the transaction rate. Oracle recommends that you use the following formula:

```
UndoSpace = UR * UPS + overhead
```

where:

- `UndoSpace` is the number of undo blocks
- `UR` is the UNDO_RETENTION value in seconds
- `UPS` is the number of undo blocks per second, or the transaction rate

- `overhead` is the small overhead (disk space) for the metadata (transaction tables, bitmaps, and so on)

The original size of the UNDO_TABLESPACE on creation was a default of around 400MB. Oracle does not reuse the space on this tablespace, so it keeps growing. We only know of one way to reclaim the space, which is to create a new undo tablespace (CREATE UNDO TABLESPACE “UNDO2” datafile.....), issue the alter system command to point to the new tablespace (ALTER SYSTEM set undo_tablespace = UNDO2), and drop the original undo tablespace, including the datafile. You should not restrict the max size of the datafile, because you will get an ‘unable to extend’ error. It does not seem to be supported.

Example instructions (full set):

```
ALTER SYSTEM set undo_retention=3 scope=both;

CREATE UNDO TABLESPACE [NEWUNDOTABLESPACENAME] DATAFILE '[TABLESPACEDATAFILE]' SIZE
2000M REUSE AUTOEXTEND ON;

ALTER SYSTEM SET UNDO_TABLESPACE = [NEWUNDOTABLESPACENAME];

DROP TABLESPACE [OLDUNDOTABLESPACENAME];
```

(for example, `NEWUNDOTABLESPACENAME] = UNDOTBS2, [TABLESPACEDATAFILE] = /data02/oracle10g/oradata/FGL/undo0201.dbf, [OLDUNDOTABLESPACENAME] = UNDOTBS1`)

Custom Tablespaces

You can create custom tablespaces by modifying the *storage-config.xml* file, that is located in the *config* directory of the Management Server installation.

Group A (configuration) tables can only be created using the SQL scripts available in the *<foglight_home>/scripts/sql* directory. This can be done using the database tool, *foglight-database-tool.jar*. For complete information, about this tool, see the *Foglight Installation and Setup Guide set*.

Group B (observation) tables can also be added. Adding them before the server starts for the first time causes them to be created in the custom tablespace. If the default tablespace already contains group B tables at the time *storage-config.xml* is updated, they remain in that tablespace until the server rolls up or purges the data in those tables.

Adding group B tables after the Management Server starts causes the server to move them over to the new tablespace as it rolls up long-term data. This means that for some tables, for example, it can take as long as three months before they are created in the database. The actual period of time depends on the configured retention policies. Some tables can remain the default tablespace indefinitely. For complete information about retention policies, see the *Administration and Configuration Help.*

To create custom tablespaces:

1 Open the *storage-config.xml* file for editing.

2 Create one or more tablespaces in the database to contain the group A tables.

In the *storage-config.xml* file, locate the `<configuration-destination>` element and replace its `<default>` element with a `<tablespace>` element using the following syntax:

```
<tablespace data= 'CUSTDATA' index='CUSTINDEX' lob='CUSTLOB' />
```

Where `CUSTDATA`, `CUSTINDEX`, and `CUSTLOB` are the names of your custom tablespaces.

NOTE: These names are examples. You can name your custom tablespaces whatever you want.

3 Create one or more tablespaces in the database to contain the group B tables.

In the *storage-config.xml* file, locate the `<destination>` element and replace its `<default>` element with using the following syntax:

```
<tablespace data= 'SAMPLEDATA' index='SAMPLEINDEX' lob='SAMPLELOB' />
```

Where `SAMPLEDATA`, `SAMPLEINDEX`, and `SAMPLETLOB` are the names of your custom tablespaces.

NOTE: These names are examples. You can name your custom tablespaces whatever you want.

4 If the database user has not been granted a quota on the newly created tablespaces, after creating tablespaces in the *storage-config.xml* file, the database administrator must edit the *scripts/sql/oracle_create_db.sql* script and add grants for the custom tablespaces, like the one that grants a quota on the USERS tablespace.

   For example:

```
ALTER USER @db.schema.username@ QUOTA UNLIMITED ON CUSTDATA;
ALTER USER @db.schema.username@ QUOTA UNLIMITED ON CUSTINDEX;
ALTER USER @db.schema.username@ QUOTA UNLIMITED ON CUSTLOB;
```

5 Save your changes and close the file.

6 **Creating custom tables before the server starts up for the first time only**.

   a Run the database tool, *foglight-database-tool.jar*, to create the group A tables in the specified tablespaces.

   b Start the Management Server.

      The server creates the group B tables on this initial startup, followed by creating additional group B tables, as needed, at runtime. The tables are created in the specified tablespaces.

7 **Creating custom tables after the server starts up for the first time only**.

   a Have your DBA move the group A tables to the configured tablespace.

   b Instruct the server to update the configuration and create the group B tables using the JMX console. In the JMX console, invoke the `mergeConfiguration()` method on the `Storage Manager Service MBean` in the JMX console. This only affects the Group B tables that the server creates from that time on.

Microsoft SQL Server Tuning

Foglight® Management Server accepts the instance-level default for database recovery (regardless of the recovery model of the model database, which is purely a template used to create new databases). There is no harm in changing the recovery model from full to simple, other than point-in-time recoverability is lost. If you require the ability to perform disaster recovery on your Foglight database, you should use the full recovery model and you should make frequent transaction log backups to ensure that it does not grow beyond an undesirable threshold. If you do not require point-in-time recoverability, you can use the simple recovery model, and there will be no additional ramifications.

PostgreSQL Tuning

PostgreSQL Server ships with a basic configuration tuned for wide compatibility rather than performance. Foglight Management Server have done some tuning configurations for the embedded PostgreSQL database, such as:

- Increase the value of “shared_buffers” to 512MB, because the previous default value is too low for the real-world workload.
- Configure “work_mem” to 8MB at least, which keeps the intermediate data in the memory and reduces the query response time.

You could take the above configurations into consideration when setting up an external PostgreSQL database for Foglight Management Server, because these configurations have been verified and are more suitable for Foglight Management Server compared to default values. If you have set up PostgreSQL for other application previously, you could adjust these configurations as needed to keep them in the same PostgreSQL.

The Foglight Management Server uses the following configurations for embedded PostgreSQL database.

- *shared_buffers* = 512MB
- *temp_buffers* = 32MB
- *work_mem* = 8MB
- *maintenance_work_mem* = 128MB
- *checkpoint_segments* = 64
- *checkpoint_warning* = 60s
- *checkpoint_timeout* = 30min

For more information about these configurations and how to tune PostgreSQL, refer to Tuning Your PostgreSQL Server and Performance Optimization.

Troubleshooting

Management Server Load Preventing Database Connections

High Foglight® Management Server load may consuming all connections to backend database. This results in any of the following problems:

- Foglight Management Server is slow to respond, or does not respond at all.
- Incoming data from monitoring agents is dropped.
- Multiple users logged in simultaneously are experiencing problems.
- Errors in the Management Server log files. For example:

```
YYYY-MM-DD HH:MM:SS.DDD ERROR [DataAccess-7-thread-4]
   com.quest.nitro.util.DefaultExceptionHandler -
   DefaultExceptionHandler.uncaughtException: [DataAccess-7-thread-4,
   java.lang.RuntimeException: org.jboss.util.NestedSQLException: No
   ManagedConnections available within configured blocking timeout
   ( 30000 [ms] ); - nested throwable: (javax.resource.ResourceException:
   No ManagedConnections available within configured blocking timeout (
   30000 [ms] ))]
java.lang.RuntimeException: org.jboss.util.NestedSQLException: No
   ManagedConnections available within configured blocking timeout
   ( 30000 [ms] ); - nested throwable: (javax.resource.ResourceException:
   No ManagedConnections available within configured blocking timeout
   ( 30000 [ms] ))
at com.quest.nitro.service.util.JDBCHelper.getConnection
   (JDBCHelper.java:124)
at com.quest.nitro.service.persistence.obs.handler.query.
   AbstractBatchQueryExecutor.run(AbstractBatchQueryExecutor.java:140)
at java.util.concurrent.Executors$RunnableAdapter.call
   (Executors.java:441)
at java.util.concurrent.FutureTask$Sync.innerRun(FutureTask.java:303)
at java.util.concurrent.FutureTask.run(FutureTask.java:138)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask
   (ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run
   (ThreadPoolExecutor.java:908)
```

```
        at java.lang.Thread.run(Thread.java:662)
        Caused by: org.jboss.util.NestedSQLException: No ManagedConnections
           available within configured blocking timeout ( 30000 [ms] ); - nested
           throwable: (javax.resource.ResourceException: No ManagedConnections
           available within configured blocking timeout ( 30000 [ms] ))
```

…

To resolve this issue, increase available connections to the database.

Directions for FMS 5.9.2

To increase the configured number of connections for the backend database:

1. Edit the appropriate *<FOGLIGHT HOME>/config/datasource/datasource-dbtype.properties* file. Take SQL Server for example, *<FOGLIGHT HOME>/config/datasource/datasource-sqlsvr.properties*.
2. Uncomment the following lines and set to your desired value. There is only one related properties:
   - *maxTotal:* The maximum number of connections that can be kept in the pool. Ensure that *maxTotal* >= *maxIdle*.
   - *maxIdle*: The maximum idle connections kept in the pool.

   Example: *maxTotal*=45 and *maxIdle*=15. The pool will have at most 15 idle connections, and can dynamically increase size up to 45 under heavy db usage for concurrent operations.
3. Save your changes, stop, and then start the Foglight service (Windows) or Foglight daemon (Unix or Linux).

Directions for FMS 5.9.1

To increase the configured number of connections for the backend database:

1. Edit the appropriate *<FOGLIGHT HOME>/config/datasource/datasource-dbtype.properties* file. Take SQL Server for example, *<FOGLIGHT HOME>/config/datasource/datasource-sqlsvr.properties*.
2. Uncomment the following lines and set to your desired value. There are four related properties:
   - *maxTotal:* The maximum number of connections that can be kept in the pool. Ensure that *maxTotal* >= *maxIdle* >= *minIdle*.
   - *initialSize*: The initial number of connections that are created when the pool is started. The *initialSize* is only used when the pool is created.
   - *maxIdle*: The maximum idle connections kept in the pool.
   - minIdle: The minimum idle connection kept in the pool.

   Example initial size 15 and max 45: *maxTotal*=45, *initialSize*=15, *maxIdle*=15, *minIdle*=15. The pool will have at least 15 connections to database, and can dynamically increase size up to 45 under heavy db usage for concurrent operations.
3. Save your changes, stop, and then start the Foglight service (Windows) or Foglight daemon (Unix or Linux).

Directions for FMS 5.6.x up to 5.7.5.8

To increase the configured number of connections for the backend database:

1. Ensure that your user account has appropriate permissions to access the *<FOGLIGHT HOME>\server\default\conf* directory (Management Server 5.6.x) or *<FOGLIGHT HOME>\server\jobss\conf* (Management Server 5.7.1 and higher):
   - **Windows®**: Log in as an Administrator.
   - **UNIX®/Linux®**: Use the Foglight user service account.

2 In the above directory, open one of the following files for editing, as applicable. If you are not sure which file to Edit, contact Quest Support.

- **Microsoft®** SQL Server®: *sqlsvr-ds.xml*
- **Oracle®**: *oracle-ds.xml*
- **MySQL™**: *mysql-ds.xml*
- **Postgres**: *Postgresql-ds.xml*

3 Locate the following line in the file, and then change the value of `20` to `50` or higher.
`<max-pool-size>${foglight.database.max_pool_size:20}</max-pool-size>`

4 **(Management Server 5.7.5.3 and higher only)** You must remove the *foglight.database.max_pool_size entry* and just enter the hardcoded value:
`<max-pool-size>50</max-pool-size>`

5 Save your changes, stop, and then start the Foglight service (Windows) or Foglight daemon (Unix or Linux).

Directions for FMS 5.5.8.x and lower

To increase the configured number of connections for the backend database:

1 Ensure that your user account has appropriate permissions to access the *<FOGLIGHT HOME>\server\default\conf* directory:

- **Windows®**: Log in as an Administrator.
- **UNIX®/Linux®**: Use the Foglight user service account.

2 In the above directory, open one of the following files for editing, as applicable. If you are not sure which file to Edit, contact Quest Support.

- **Microsoft®** SQL Server®: *sqlsvr-ds.xml*
- **Oracle®**: *oracle-ds.xml*
- **MySQL™**: *mysql-ds.xml*

3 Add `min-pool-size` and `max-pool-size` elements between `password` and before `prepared-statement-cache-size`, as follows:

```
<password>${foglight.database.password}</password>
<min-pool-size>20</min-pool-size>
<max-pool-size>50</max-pool-size>
<prepared-statement-cache-size>100</prepared-statement-cache-size>
```

4 Save your changes, stop, and then start the Foglight service (Windows) or Foglight daemon (Unix or Linux).

The described configuration changes increase the load on the Foglight database server. You must make sure that you have adequate resources for both the Foglight Management server and the database server hosting the Foglight repository.

NOTE: If you need to enlist our professional services group to help size your environment, contact your sales representative. This is beyond the scope of Quest Support.

6

High Availability (HA) Tuning

This chapter provides information about the high availability related options that can have an effect on performance, and describes the applicable performance-related options.

- Tuning Connection Issues in an HA Implementation
- Managing Hosts with Multiple Network Interfaces
- JDK with IPv6 on Linux
- Management Server Automatically Restarted
- Other JGroup Related Issues and Information

Tuning Connection Issues in an HA Implementation

High Availability (HA) means running multiple Foglight® Management Servers in a cluster, where one is the primary and one or more are secondaries (on standby). The general setup, as well as the communication between the members of the cluster, requires special attention.

In some cases, the communication between servers in the cluster is not reliable. Sometimes unwanted behaviors occur, such as:

- A secondary server takes over while the primary server is still running.
- Messages from an HA member are discarded as “message from non-member”.
- When starting up a secondary server, it fails to recognize the primary server.

These issues may be attributable to JGroup (the underlying communication package that JBoss® uses for its HA implementation) and to the fact that a Foglight HA implementation uses UDP for communication by default and UDP is by nature an unreliable protocol.

If you encounter these HA issues, consider performing the following tuning exercise.

To tune the Management Server:

1 Ensure the servers have synchronized system clocks. Out of sync server system clocks greatly increase the risk of communication errors.

> **NOTE:** If the system clock is out of sync by 5 seconds or more during HA server startup, a warning message similar to the following appears in the log:
>
> `Warning: system time not in sync with primary server, Mon Aug 30 11:47:57 ECT 2010 vs Mon Sug 30 11:48:52 EDT 2010.)`

2 Reconfigure JGroup to use TCP instead of UDP as the communication protocol. This is suitable for clusters with a small number (under 5) of predetermined servers.

> **NOTE:** You must make the following changes to each server in the cluster.

a Shut down the servers.

b Edit *server/default/deploy/cluster-service.xml* by doing the following:

Comment out the following element:

```
<config>
     <udp....
</config>
```

Uncomment the next block:

```
<config>
     <tcp....
</config>
```

c Locate the line beginning `<TCPPING initial_hosts="thishost[7800],thathost[7800]"` and change `thathost` to the host name of the secondary server in the cluster. If there are more than two servers in the cluster, add those servers to the list as well: `thishost[7800],host2[7800],host3[7800].`

d Save your changes.

e Start the primary server. Wait for the primary server to start up completely, then restart the secondary server(s).

Managing Hosts with Multiple Network Interfaces

When configuring HA servers that are installed on hosts with multiple network interfaces (that is, with multiple IP addresses and host names), you must specify the IP address that is to be used for communication between the servers.

To specify the IP address or host name to be used for communication, set the following:

```
--host=<your_ip_or_hostname>
```

You can set this using the command line interface or by adding the following to the *server.config* file:

```
server.cmdline.option0 = "-Djboss.bind.address=127.0.0.1";
```

or

```
server.cmdline.option0 = “-Dbind.address”;
server.cmdline.option1 = “127.0.0.1”;
```

where `127.0.0.1` is the desired IP address

or

```
server.cmdline.option0 = "-Djboss.bind.address=host1.example.com";
server.cmdline.option1 = "-Dbind.address=host1.example.com";
```

where `host1.example.com` is the desired host name

You also need to reconfigure the restarter to perform server health check with the same host name or IP address. To do so, open the r*estart_monitor.config* file, find the line beginning with `health.check.url`, and replace `localhost` with the host name or IP address specified with `--host`.

In addition, an HA configuration has a feature that automatically redirects HTTP requests from secondary servers to the primary server. If the primary server has multiple network interfaces, you must specify the host name that is to be used to serve HTTP requests. You can do this using the command line interface, or by adding the following to the *server.config* file:

```
-Dquest.host.name=”host1.example.com”
```

JDK with IPv6 on Linux

Foglight® Management Servers running on a VMware® Linux® x86_64 image fail to start up and a `"java.net.BindException: Cannot assign requested address exception"` is raised.

This is a JDK issue with IPv6 on Linux, not a true High Availability issue. It also impacts standalone servers. However, the issue is documented here because it occurs when the JBoss® cluster service is starting up.

Workaround: Start the Management Server with the following command:

```
-Djava.net.preferIPv4Stack=true
```

For more information, see: the JGroups FAQ (http://community.jboss.org/wiki/JGroupsFAQ) and Oracle® Bugs for it (http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6521974).

Management Server Automatically Restarted

Sometimes, after what appears to be a normal and successful startup, an HA server would be automatically shut down and restarted. The most likely reason for this is a misconfiguration of health check URL in the *restart_monitor.config* file.

For example, you may have reconfigured the HTTP port of the Foglight® Management Server or reconfigured the IP address that the server is bound to, but forgotten to reconfigure the health check URL of the restarter. If the restarter cannot contact the Management Server for health check, it considers the Management Server not responsive and restarts it.

You can check the *server_restarter_xxxx-xx-xx_xxxxx_xxx.log* file to determine if this scenario is what caused the restart. If so, edit the *restart_monitor.config* file by locating the line beginning with `health.check.url` and configuring the URL properly.

Other JGroup Related Issues and Information

```
ERROR org.jgroups.protocols.UDP max_bundle_size (64000_ is greater than the largest
TP fragmentation size (8000):
```

https://jira.jboss.org/browse/JGRP-798

“Cross-talk” can occur between servers on different clusters:

https://jira.jboss.org/browse/JGRP-614

7

Agent Tuning

This chapter provides information about the agent-related options that can have an effect on performance and describes the applicable performance-related options.

- Topology Changes and Topology Churn
- Canonical Data Transformations (CDTs)
- Agent Weight / Environment Complexity
- Large Topologies
- Sampling Frequency
- XML-HTTP Agent Adapter
- Dropped Agent Manager Log Messages
- Java EE Technologies

Topology Changes and Topology Churn

Monitored data in the Foglight® Management Server can be sub-divided into two areas with distinct properties:

1. Topology—data representing monitored entities
2. Observations—data (including metrics) observed about monitored entities

It is generally assumed that topology objects change very little over time. Observations are expected to be highly volatile over time.

The decision of whether to add a particular piece of data to the topology model or to treat it as an observation is made during cartridge development. This decision is expressed in the software using CDT configuration.

The server is generally optimized to handle stable topology models where topology changes are infrequent. If topology changes occur on a more regular basis, this is known as Topology Churn, and it usually results in diminished server performance.

Table 1. Topology churn

Motivation	The server is optimized to handle stable topology models. Therefore, it is expected that the Management Server is configured to minimize topology changes.
Symptom	The browser interface performance is poor (that is, it responds slowly). Topology queries (in the browser interface as well as in groovy scripts) are slow. Data is being dropped by the Data Service.
Diagnosis / Verification	Check the System Changes chart on the Alarms System dashboard. Check the `batchesDiscarded` metric produced by the Data Service that is located in **Dashboards** > **Foglight** > **Servers** > **Management Server View** > **Data Service**. Explore the topology model using the Data dashboard located in **Dashboards** > **Configuration** > **Data** and try to locate any noise (that is, potentially redundant) or unexpected objects.

Table 1. Topology churn

	In some cases, the agent defaults may be too broad. The agent may be monitoring everything that is visible to it. Configure agents to monitor only what is required.
Tuning	Agent and/or CDT changes may be required. In such cases, support bundles are very helpful in the investigation. Allocate more system resources to the Management Server.
Note	The amount of topology changes that the server can process depends greatly on the overall system configuration (Host OS, database, and hardware).

Canonical Data Transformations (CDTs)

Theoretically, CDTs can be a performance bottleneck. Unfortunately, it is not easy to tune them. In most cases, a cartridge update is required.

Table 2. Canonical Data Transformations (CDTs)

Motivation	CDTs convert data received from agents into the server's internal representation (that is, into the Canonical Data Form). Although this process is usually fast, it can be computation-intensive and therefore may cause performance issues.
Symptom	The server is generally slow overall. The CDT `transformTime` metrics are high. Typical values for OS Cartridge agents are in the 0.01 - 0.1 second range, in 15-minute intervals. CDT `transformTime` metrics can be accessed through the browser interface by going to **Dashboards** > **Configuration** > **Data** > **Foglight** > **All Data** > **AllTypeInstances** > **TopologyObject** > **subTypes** > **CanonicalDataTransform** > **instances** > **...** > **transformTime**.
Diagnosis / Verification	CDT tasks are visible in thread dumps.
Tuning	Tuning will probably have to be done by the agent development team. A support bundle along with the thread dumps will be very helpful in the investigation.

Agent Weight / Environment Complexity

Foglight® Management Server maintains an internal metric that represents, roughly, the amount of work the server has to do in order to process the data collected by the agents.

This metric is called `aggregateAgentWeight`. It is available from the EnvComplexityEstimator service in the Management Server Metrics dashboard: **Dashboards** > **Foglight > Servers > Management Server Metrics** > **<*CatalystServer*>** > **Services** > **com.quest.nitro:service=EnvComplexityEstimator** > **aggregateAgentWeight**.

This metric is derived from the number and types of connected agents according to: `<foglight_home>/config/agent-weight.config`.

The value of the metric is typically expressed in agent units. Recent server builds generally work well with up to 4000 agent units connected.

NOTE: The maximum agent weight a server can support depends greatly on the host system configuration and the hardware capacities.

The `agent-weight.config` set-up is based on Quality Assurance (QA) capacity test results. Generally, it should not be changed. However, if new data on the relative agent weight is available, the configuration file can be adjusted manually. You must restart the server after you change this configuration file.

Large Topologies

Table 3. Large Topologies

Motivation	On occasion, agents may send so much observation data that the resulting topology model becomes too large and causes performance problems in the server.
Note	Java EE Technologies agents are the most likely to cause this type of situation, if they are not carefully pre-configured.
Symptom	The server is in overload condition. Agent data is being dropped. The browser interface performance is poor (that is, it responds slowly).
Diagnosis/Verification	Create a support bundle and check the topology size breakdown. A topology size breakdown by type is available in the diagnostic snapshot files that are part of each Management Server support bundle. Look for large topology object instance counts. **NOTE:** Acceptable count ranges will differ depending on the amount of resources allocated to the Management Server.
Tuning	1. Stop data collection on agents that produce an excessive number of topology objects. 2. Re-configure those agents to produce reasonable amounts of topology data. 3. Delete any excess topology objects. 4. Resume data collection on the affected agents.
Example	There is a large number of JavaEEAppServerRequestType topology objects. **NOTE:** When that is the case, there will also be a large number of objects for other topology types. You can reduce the number of JavaEEAppServerRequestType topology objects by adjusting the `FilteringRules` parameter in *recording.config*. **NOTE:** This is a Cartridge for Java EE Technologies topology object example. In principle, any cartridge may cause this kind of performance problem.

Sampling Frequency

All converted Foglight® 4 agents have a secondary ASP (usually called `SamplingFreq`) that controls data collection frequencies.

To access that ASP, select an agent in the Administration module of the browser interface and then choose Edit Properties.

XML-HTTP Agent Adapter

The XML-HTTP Agent Adapter enables agents to post data in XML format over HTTP(S).

By default this adapter is configured to reject posted XML text larger than 20MB.

Table 4. XML-HTTP Agent Adapter

Motivation	The XML-HTTP agent adapter rejects large XML submissions to protect the server from memory overload.
Symptom	An agent that uses the XML-HTTP protocol complains that the server is rejecting its data.
Diagnosis/ Verification	Check the agent logs for specific error messages indicating that the data was rejected due to size constraints.
Tuning	Open *web.xml*, which is located in *<foglight_home>/cartridge/XmlHttpAgentAdapter-*.car*, component *XmlHttpAgentAdapter-5_2_4\WebApp-\foglight-xml-agent-adapter.war*, file *WEB-INF\web.xml*, and adjust the `maxContentLength` parameter.
Note 1	You will need to unzip the *.car* file, then unzip the *.war* file, edit the *web.xml* file, and then re-package the cartridge manually (using zip).
Note 2	After you modify the cartridge, you will have to delete *<foglight_home>\state\cartridge.exploded\XmlHttpAgentAdapter-**, and then restart the Management Server.

Dropped Agent Manager Log Messages

The default settings for the Foglight® Agent Manager are configured to impact the system as little as possible. However some agents put additional load on the system, which results in dropped log messages such as the following:

```
2010-11-29 17:03:56.816 VERBOSE Upstream Polling-0
com.quest.glue.core.comms.transport.http.Client - Upstream message exchange took
100,525 ms which is close to or above the maximum polling interval of 10,000 ms.
Batch size has been decreased to 162.

.

.

.

2010-11-29 17:12:03.376 WARN CachedOutgoingQueueWorker
com.quest.glue.common.comms.CachedOutgoingQueue - Removed 253 messages because the
message queue is full.
```

To resolve this issue, increase the message cache size (that is, the `max-disk-space` setting in the *<FglAM_home>/state/default/config/fglam.config.xml* file).

Java EE Technologies

The configuration of the Cartridge for Java EE Technologies agent has the potential to significantly influence Foglight performance.

For information about configuring the Cartridge for Java EE Technologies, see the Foglight® for Java EE Technologies documentation.

8

Store and Forward Processing

Store and forward processing is a way to collect and store data on a monitored host, followed by forwarding the stored data to the server.

- About Store and Forward Processing in Foglight
- Store and Forward Processing and the Foglight Management Server
- Store and Forward Processing and the Foglight Agent Manager
- Examples

About Store and Forward Processing in Foglight

Store and forward processing refers to collecting and storing data on the Foglight® Agent Manager, followed by forwarding the stored data to the server. This type of processing is required when the communication between the Foglight® Agent Manager and the Foglight Management Server is disrupted.

Any data with older timestamps that is successfully forwarded to the server triggers data-driven rules and derived metrics. Time-driven rules do not re-evaluate data with older timestamps, even if that data meets their conditions and would cause them to fire if collected in real-time, as indicated in the table below. For complete information about rule triggers or derived metrics, see the *Administration and Configuration Help*.

Table 1. Entities and observations

Trigger	Current Observation	Forwarded Observation
Time-driven rule	Alarms and actions occur as expected. Alarm counts are correct.	Alarms and actions do not occur as expected. Alarm counts can be incorrect.
Data-driven rule		Alarms and actions occur as expected. Alarm counts are correct.
Event-driven rule		
Derived metric (including SLA	Data-driven derived metrics are calculated as expected. Time-driven derived metrics, such as the Service Level Agreement (SLA) are not re-calculated and as such can be incorrect.	

Store and Forward Processing and the Foglight Management Server

Foglight® Management Server has a configurable time window for accepting and processing the forwarded data. When the store and forward processing occurs as a result of a network disruption, the Foglight Agent Manager only stores the data on the host during that time window, enabling you to control the amount of data being stored on the monitored host for optimal performance.

By default, the Foglight Management Server accepts only the data that is collected within one hour from the moment it is received. This time window can be increased using a virtual machine (VM) option, `-Dfoglight.data_service.max_past_timestamp_delta`, in the configuration file *<foglight_home>/config/server.config*:

```
server.vm.option<x>="-
Dfoglight.data_service.max_past_timestamp_delta=<value_in_milliseconds>";
```

Where:

- *x* is the number associated with the first available `server.vm.option` in *server.config*.
- *value_in_milliseconds* specifies the duration of the store and forward period.

Increasing the length of store and forward time window is useful in case there is a longer-term network outage.

For complete information on how to increase the length of the time window reserved for store and forward processing, see the section *Enabling the Collection of Data with Older Timestamps* in Chapter 8 of the *Administration and Configuration Guide*.

IMPORTANT: Any data with older timestamps that is accepted on the Foglight Management Server triggers data-driven rules and derived metrics. Time-driven rules and derived metrics do not re-evaluate data with older timestamps, even if that data meets their conditions and would cause them to fire if collected in real-time. For more information about this behavior, see About Store and Forward Processing in Foglight on page 39.

Store and Forward Processing and the Foglight Agent Manager

When a Foglight Agent Manager's connection to the Management Server is disrupted, the Agent Manager continues to store data from hosts on disk. When the connection is restored, the Agent Manager forwards the stored data to the server.

The data is stored in the form of data packages, or messages. A message typically contains data samples or information about the agents' state. The collected data objects are serialized into messages and stored on disk. The serialized messages can be easily re-read from disk and translated back into the data objects, collected during the store and forward period.

The amount of the stored data is controlled with a set of configuration parameters. For more information, see Viewing the Settings that Control the Size of Stored Data on page 40.

Viewing the Settings that Control the Size of Stored Data

The default time window during which the host collects and stores data that will be accepted by the server after the connection is restored is one hour. This setting can be changed to a desired value using a VM option in a configuration file on the server side. The Foglight® Agent Manager keeps track of this time window and discards any messages that are collected and stored on the host outside of this time window. For more information about changing the length of the time window for store and forward processing, see Store and Forward Processing and the Foglight Management Server on page 39.

The size of the stored data is controlled with a set of parameters in the file *<foglight_agent_manager_home>/state/default/config/fglam.config.xml*. In that file, the `<config:queue-sizes>` element controls the amount of the stored data using the following attributes of its `<config:upstream>` and `<config:downstream>` sub-elements:

- `max-disk-space` specifies the amount of disk space in KB that can be used to store messages. Zero '0' means no messages are written to disk, minus one, '-1', indicates no limit. The default is `1024`, meaning

that the amount of messages written to disk cannot exceed 1024 KB or 1 MB, including upstream and downstream connections.

- `max-batch-size` specifies the number of messages that are sent from the queue to the server when a connection is established. Minus one, '-1', indicates that every available message is included in each batch. The default is `500`, meaning that a batch of 500 messages is sent to the server when the connection is restored, including upstream and downstream connections.

For example, the default configuration instructs the Foglight Agent Manager to reserve 1 MB of disk space for storing messages and to limit the number of messages in a batch to 500, for upstream connections:

```
<upstream max-queue-size="-1" max-disk-space="1024" max-batch-size="500"/>
```

NOTE: The `max-queue-size attribute` is used internally and should not be changed.

CAUTION: The above attributes illustrate a default configuration. Changing them is only recommended when instructed to do so by Quest Support.

To view the Foglight Agent Manager settings that control the size of stored data:

1. On the monitored host, using a file browser, navigate to the directory *<fglam_home>/state/default/config*, where *fglam_home* refers to the location of the Foglight Agent Manager installation.
2. Open the file *fglam.config.xml*.

   The file contains the settings used by the Foglight Agent Manager, such as the monitored host name, the URL and port number for connecting to the Foglight Management Server, and many others. These settings are specified at the installation time. For more information about installing the Foglight Agent Manager, see the *Agent Manager Guide*. For additional information about a specific element, view the contents of its `<config:documentation>` sub-element. For more information see the contents of *<fglam_home>/state/default/config/fglam.config.xml*.
3. In the *fglam.config.xml* file, locate the `<config:queue-sizes>` element and view the attribute values of its `<config:upstream>` and `<config:downstream>` sub-elements.

   ```
   <config:queue-sizes>
     …
     <config:upstream max-queue-size="-1" max-disk-space="1024" max-batch-
     size="500" allow-runtime-change="true"/>
     <config:upstream-verified max-queue-size="-1" max-disk-space="512" max-
     batch-size="250" allow-runtime-change="true"/>
     <config:downstream max-queue-size="-1" max-disk-space="1024" max-batch-
     size="500" allow-runtime-change="true"/>

   </config:queue-sizes>
   ```

   The above listing illustrates default settings, indicating that the disk space that is used to store non-sent messages is 1024 KB (or 1MB), and that a batch of messages that are sent to the server when the connection is restored can contain 500 messages, for both upstream and downstream connections.

   IMPORTANT: Changing the default settings of the above attributes is only recommended when instructed to do so by Quest Support.

Examples

This section contains scenarios, each illustrating a set of common results typically found in store and forward processing:

- Example 1: Reporting on Disk Usage on page 42
- Example 2: Calculating Service Availability Metrics on page 42

Example 1: Reporting on Disk Usage

A Foglight® Management Server monitors a group of hosts, each using the Foglight Agent Manager to communicate with the server. The network between a monitored host and the server becomes unavailable at 4:55 PM, and is restored at 5:00 PM. Prior to the connection disruption, the monitored host's disk usage was within a normal range.

At 4:55 PM, the connection with the server is lost, and the Agent Manager process continues to collect the data from the host, and to store it on disk.

At 4:57 PM, a high disk usage is reported. The Agent Manager stores this information on disk.

At 5:00 PM, the network connection between the Management Server and the Agent Manager is restored. The Agent Manager forwards the data stored between 4:55 and 5:00 PM to the Management Server. The increase of disk usage is reported at 5:00 PM, when the data arrived, not at 4:57, when it actually occurred.

Example 2: Calculating Service Availability Metrics

The SLA is calculated every minute for the current minute, and is stored as a time-driven derived metric. When the Foglight® Management Server accepts forwarded data, the SLA is not re-calculated using the forwarded data.

For example, the SLA calculations take into consideration the expected average response time. The Foglight Management Server monitors a host that becomes disconnected for three hours due to a network outage. The host uses the Foglight Agent Manager to communicate with the server. At the time the connection is lost, the Foglight Agent Manager continues to collect the data from the host, storing the collected data on disk, including the collected data, which now contains the average response metrics.

The Foglight Management Server is configured to accept data that is older than one hour, with the store-and-forward time widow set to eight hours. The server's configuration file, *<foglight_home>/config/server.config*, was edited prior to the server startup, and the option `foglight.data_service.max_past_timestamp_delta` is set to 28,800,000 milliseconds (which equals to eight hours) as follows:

```
server.vm.option0="-Dfoglight.data_service.max_past_timesta
   mp_delta=28800000";
```

The Agent Manager is aware of this configuration and continues to collect and store the data past the one-hour default, without deleting any new messages that come in.

During the connection disruption, the host's average response metrics change over time. After three hours, the connection between the host and the server is restored, and the Foglight Agent Manager forwards the stored metrics to the server, including the average response metrics. The Foglight Management Server persists the forwarded data into the repository. The Foglight Management Server does not make use of the forwarded data to re-calculate the SLA.

A

Appendix: Performance Health Check

The purpose of a health check is to quickly ensure that Foglight® is functioning properly. There are many things you can check. The more important ones are documented in this chapter.

- Is the Server Getting the Right Amount of Memory?
- Is the Server Getting Enough CPU?
- Is the Database too Slow?
- Is the Database Growth Reasonable?
- Is There Too Much Data to Process?
- Is the Model Stable?
- Are Too Many Alarms Firing?
- Is the Business Logic Properly Tuned?
- Are There Too Many User Requests?

Is the Server Getting the Right Amount of Memory?

By default, Foglight® tries to reserve three quarters of the local available physical memory. It is normal for Foglight to consume all of that memory on start up. Foglight reserves its full amount of memory on start up. Foglight is configured this way to avoid excess garbage collection.

Foglight uses a lot of temporary memory. This is due to the metric processing. It is normal and healthy for Foglight's internal memory consumption to rise and fall.

If there is enough memory reserved, the Foglight Java™ Virtual Machine (JVM) heap graph will display an adequate amount of free memory. It is normal for the memory to cycle up and down. As long as these memory cycles are not close together (which indicates excessive garbage collection), then all is well. Also, as long as free memory continues to be returned through garbage collection, then the amount of memory configured is fine.

Management Server Memory is Healthy if ...

- The amount of memory consumed is typically less than 80 percent of the maximum.
- There are no log messages indicating a failure to create threads.
- The operating system is not constantly paging to keep the Management Server's memory available.

Management Server Memory Checks

- Check the -Xmx and -Xms settings in the Foglight® log to make sure they match expectations.
- Look at the JVM heap graph and make sure the free memory is not stuck beyond 80 percent of the maximum memory.
- Look for a failure to create native threads in the log. If native threads cannot be created, then either stack space or total memory should be decreased.
- Look at the Management Server system's paging. If memory is frequently paging, then the memory requested by the server is too much for the system. This could be because another large process is running.

Possible Actions

- If the memory utilization is consistently greater than 80 percent:
  - Allocate more memory to Foglight by increasing the value of the –Xmx configuration parameter, if the environment has enough available physical memory to support the increase.
  - Reduce the load on the server.
- If the system is paging, reduce the amount of memory allocated to Foglight by decreasing the value of the –Xmx configuration parameter.

Is the Server Getting Enough CPU?

The server may be starved for CPU on the system hosting the Management Server. When this happens, it is typically because Foglight® is competing for CPU with other processes.

Management Server CPU is Healthy if ...

- The Foglight® browser interface performance is fine.
- The Management Server is the top CPU-consuming process on the system.
- The CPU consumption does not average over 90 percent.

Management Server CPU Checks

- After you log in for the first time and the Hosts page is drawn for the first time, test the performance of the Hosts dashboard. If it takes longer than 20 seconds for the Hosts dashboard to be drawn, then it is possible there is a CPU issue. Continue to the next check.
- Check to see if Foglight® is consuming the bulk of the CPU on the system. If other processes are consuming CPU, then these processes should be moved to another system. Foglight needs the CPU.
- If there are items in the run queue, Foglight may be competing for resources with another process. Check the top CPU consumers to make sure the Management Server is getting the CPU it needs.
- If Foglight is consuming in excess of 90 percent of the CPU on average, then it needs a stronger CPU or the server needs to be tuned to do less.
- Make sure that Foglight is consuming on all available CPUs. If it is not, a process or processor affinity setting may need to be tweaked.

- If Foglight is not consuming significant CPU, then it is likely there is a database issue or some other performance deadlock.
- If Foglight is running on a virtual image, it is possible that the image configuration is not providing enough resources (that is, if the browser interface is slow, you are not doing a lot of GCs, and there does not seem to be any swapping). If the CPU utilization is not 100 percent, the Management Server is not underpowered. In this scenario, if you see a run queue as well, but you do not see other processes competing, then it is likely that your ESX® configuration is not providing enough CPU.

Extended Browser Interface Checks

Although the two are not directly coupled, user interface performance is related to CPU health. Therefore, it makes sense to check browser interface performance when verifying whether or not the CPU is healthy. To check browser interface performance, measure the first time and reload times for the following views:

- Hosts
- Alarms
- Services
- Administration Home
- Manage Rules
- Edit a Rule
- Agent Status

You should perform these tests in order to detect any known bottlenecks.

Possible Actions

- Remove other processes from the system.
- Reconfigure the virtual image to have access to more resources.
- Move Foglight to a more powerful machine.
- Configure the operating system to allow Foglight® to have access to all available CPUs.

Is the Database too Slow?

A slow database slows down database queries. This in turn slows down metric queries from the browser interface.

The Database is Healthy if ...

- The number of JDBC connections available is close to the maximum. This means that queries are not getting stuck waiting for the database to return.
- The CPU utilization on the database machine is substantial, but not greater than 90 percent on average, indicating a non-blocked system that has enough resources available.
- The browser interface is performing well on metric queries. This can be tested with any dashboard that graphs a metric.

Database Checks

- First, perform the Hosts dashboard performance test from the CPU section of the health check. Poor browser interface performance coupled with low CPU utilization indicates either a server block or a database problem. To check to see if there is a database problem, proceed to the next check.
- Check the available JDBC connections.
- Check CPU and I/O utilization on the database machine.
- Check the amount of memory allocated to the database:
  - On an Oracle® database, check the SGA size:

    ```
    SELECT SUM(bytes) FROM v$sgainfo WHERE resizeable = 'Yes'
    ```

  - On a MySQL database, check the InnoDB buffer pool size:

    ```
    SHOW VARIABLES LIKE 'innodb_buffer_pool_size'
    ```

- Check the database timing metrics.

Possible Actions

- Tune the database cache parameters. You may need to allocate more memory to the database if I/O utilization is high.
- Move the database to a more powerful machine.
- Tune the database optimization.
- Run the following commands to check the database timing metrics:
  - `./fglcmd.sh -usr foglight -pwd foglight -cmd util:topologyexport -f handlers.xml -topology_query CatalystPersistenceHandler`
  - `./fglcmd.sh -usr foglight -pwd foglight -cmd util:metricexport -f retrieve-last-n-values-time.csv -metric_query "retrieveLastNValuesTime from CatalystPersistenceHandler for 1 week" -output_format csv`
  - `./fglcmd.sh -usr foglight -pwd foglight -cmd util:metricexport -f retrieve-time.csv -metric_query "retrieveTime from CatalystPersistenceHandler for 1 week" -output_format csv`
  - `./fglcmd.sh -usr foglight -pwd foglight -cmd util:metricexport -f retrieve-earliest-time.csv -metric_query "retrieveEarliestTimeTime from CatalystPersistenceHandler for 1 week" -output_format csv`
  - The metrics returned in the CSV file are in milliseconds. Look for times of 500 or greater in AVERAGE, or greater than 10000 in MAX. If these values are consistently high, the database is failing to keep up with the load, causing the Management Server to be slow.

Is the Database Growth Reasonable?

Database growth is a long term concern, but it should be assessed early. Foglight controls database growth with persistence policies that determine how data is rolled up.

A health check on database growth is really about making sure that you have a database size expectation. This can be done using the Foglight® sizing spreadsheet. Once a size is determined, it can be configured into Foglight's self-monitoring rules to ensure that adequate warning is provided.

Without a purge policy in place, Foglight will grow indefinitely. By default, Foglight is not equipped with a purge policy.

Database Growth is Healthy if ...

- You have a target maximum for how large you will allow your database to get.
- You have appropriate policies in place to keep the database smaller than that maximum.
- The database growth patterns indicate proper growth rate.

Database Checks

- The Foglight® database growth rate cannot be assessed in the early stages of an installation. To properly assess the Foglight growth rate, wait a couple of weeks and then measure the database growth rate (slope of the curve).
- Set up the registry variables that Foglight uses to monitor its own size. If alarms occur, then database purge operations are required.

Possible Actions

- Set a retention policy to ensure that data is deleted after a certain amount of time has passed. This can be done globally, per type, or per metric.

Is There Too Much Data to Process?

It is possible for Foglight® to be performing fine, but still be overloaded. Foglight does have documented limits for the amount of load it can handle. If these limits are exceeded, Foglight will consume a lot of CPU on the Management Server system and on the database system, and have a slow browser interface (that is, it will exhibit the union set of all symptoms).

Data Volume is Healthy if ...

- Data is not being dropped. If Foglight is able to process all the incoming data, then the data volume is fine.
- The measured unit number for data is fine.

Data Volume Checks

- Look at the data service graph to see if there are any dropped batches of data. It is fine for data to be dropped occasionally due to temporary overload. However, even a small amount of dropped data over a long period of time indicates data overload.
- Look in the Management Server logs for dropped data messages.
- Look at the batch processing time. If the batch processing time is excessive, then data is backing up. This is an early indicator that data will be dropped. If batch processing time is growing, this will eventually result in data overload.

Possible Actions

- Remove agents.

- Provide more resources to the Management Server, so that it can keep up.
- Tune agents to collect data less frequently.

Is the Model Stable?

Foglight® requires a stable topology in order to function well. A stable topology has infrequent topology changes. If topology changes occur often, it means the model is growing or changing. Model growth leads to memory issues. Model changes cause churn in the business logic that in turn result in CPU consumption.

The Model is Healthy if ...

- There are no significant topology changes over time.
- If the model size (topology object count) is constant.

Model Checks

- Check the topology change count on the Alarms page. If there are changes in every time interval, then there is a strong possibility that the model is unstable.
- Check the topology object count as a function of time. If no agents have been added or significantly changed recently, then the model should not be growing.

Possible Actions

- Tune agent collection to avoid changes.

Are Too Many Alarms Firing?

It is possible for Foglight® to be configured in such a way that too many alarms are firing. There are two main manifestations of alarm instability:

- Massive alarm fire-and-clear (that is, an excessive number of alarms are firing and clearing).
- A large number of total alarms.

A large number of alarms can create problems when viewing alarms pages and can have an adverse affect on the functioning of federation. When a large number of alarms are generated, that consumes resources on the Management Server, which can slow down other operations.

The Number of Alarms is in a Healthy State if...

- The total alarm count on the Alarms page meets the expectations for your monitoring goals. Any total greater than 1000 should be a concern. Alarms are clipped at 10,000, so any number approaching 10,000 indicates a problem.
- The total number of alarms, including cleared alarms, is reasonable. It is possible to have a low number of alarms while having a huge number of cleared alarms. This is an indication of massive alarm fire-and-clear.

Alarms Checks

- Look at the total count on the Alarms page.
- On the Alarms page, in the alarm table, change the alarm filters to include cleared alarms and observe the new alarm totals.

Possible Actions

- Tune rules.
- Disable rules.

Is the Business Logic Properly Tuned?

It is possible to configure Foglight® so that it is doing too much on the models. This usually happens as a result of massive misguided customization. Foglight's customizability makes this easy to do. A set of derived metrics bound to TopologyObject can add literally thousands of expensive expressions.

Business Logic is Healthy if...

- CPU on the system is low when no browser interface operations are initiated.
- The number of rulettes and derived metric expressions on the system is in proportion to the number of topology objects. If there are more than 100,000 derived metric expressions, then the system is doing too much.

Business Logic Checks

- With little data coming in and no browser interface operations (including report generation), observe the CPU utilization on the Foglight® Management Server. If it is high, and the memory settings for the Management Server are reasonable, and it is the Management Server that is consuming the CPU, then it is likely that there is too much business logic running.
- Look at the rulette and derived metric expression counts. If they are greater than 100,000, or more than two times the total number of objects in the system, then it is possible there is too much business logic running.

Possible Actions

- Revisit the customizations that have been applied.
- Turn off rules.
- Clean up the model by removing items that are no longer needed.
- Contact Quest for information on how to optimize business logic per cartridge or domain.

Are There Too Many User Requests?

In Foglight®, user requests include browser interface requests and report generation. It is possible that Foglight can become slow simply because there are too many users trying to access the system.

User Activity Is Healthy If...

- The number of users is low (less than 10).
- The number of scheduled reports is low.
- Users report reasonable response time (5 to 15 seconds for a page to be drawn).
- Pages perform slowly only rarely.

Possible Actions

- Reduce the number of users, or the number of generated reports.
- Use Federation to separate user requests from data processing. Consider a stand-alone Federation instance for reporting.

B

Appendix: Analyzing a Support Bundle

The support bundle is the first line analytic tool for checking the performance of the server. It can be generated through the command-line interface or through the **Administration > Setup and Support > Manage Support Bundles** dashboard.

This appendix describes what to look for in the support bundle.

- Server Log
- Diagnostic Snapshot
- Analyzing a Performance Report

Server Log

The support bundle contains server log files. The log files are located in *<foglight_home>/logs/* and their names adhere to the following format:

ManagementServer_yyyy-mm-dd_hhmmss_nnn.log

NOTE: Any log entry that contains the keyword ERROR is suspicious.

Topology Sync

Derivation and Rule Service synchronize their state against the topology whenever topology changes. Frequent topology changes may cause frequent synchronization effort, and this is noteworthy. The length of time it takes for a synchronization indicates overall load or topology complexity. Take note of the frequency and length of the following message bracket:

```
yyyy-mm-dd hh:mm:ss.mmm VERBOSE [TopologySyncQuartzScheduler_Worker-X]
com.quest.nitro.service.derivation.DerivationService - Starting derivation topology
sync.

...

yyyy-mm-dd hh:mm:ss.mmm VERBOSE [TopologySyncQuartzScheduler_Worker-X]
com.quest.nitro.service.derivation.DerivationService - Derivation topology sync
finished.
```

Topology Limits

Foglight® restricts the number of instances of an object of any type to 10000. When this limit is reached, the event is logged and should be investigated. Either it is an exceptional situation or the limit has to be raised (per type, using the `foglight.limit.instances` registry variable).

```
yyyy-mm-dd hh:mm:ss.mmm ERROR   [Data-X-thread-Y]
com.quest.nitro.service.topology.TopologyService - Failed to create a topology
object with name Something: {...}:
com.quest.nitro.service.sl.interfaces.topology.TopologyChangeVetoException: The
configured limit of 10000 instances of type SQLServer_Backup has been reached.
```

Diagnostic Snapshot

The diagnostic snapshot is a text file contained in the Foglight® Management Server support bundle. It contains information that is very useful for diagnosing Management Server performance issues. Due to the fact that there is often a lot information in the snapshot, the information can be difficult to interpret.

The following is intended to serve as a map for extracting useful information from the snapshot. It is not meant to cover all aspects of the snapshot, just the more typical ones.

Memory Consumption

The memory consumption section shows the available and used memory:

```
-------------------------------------------------
Current memory load: 0.6002591924983527
Rounding error threshold: 1.0E-5
-- Heap --
Min: 0.5
Max: 0.995
Min Free: 67107840
Weight: 0.2
Current max: 1070399488
Current used: 842635096
Current load: 0.11604667718075726
-- Memory Pool 'CMS Old Gen' --
Min: 0.5
Max: 0.995
Min Free: 67107840
Weight: 0.8
Current max: 1040187392
Current used: 831741608
Current load: 0.4842140916194773
-------------------------------------------------
```

Threads, Deadlocks, and Overall CPU Usage

A thread dump shows the current thread activity followed by thread utilization. It also provides markers to show deadlocked threads.

Useful Information

---- jboss.system:type=ServerInfo

This service provides important information about the Management Server hardware and software environment, including:

- AvailableProcessors
- OSName
- OSVersion
- OSArch
- JavaVMName (tells you 32 or 64 bits)
- JavaVersion
- MaxMemory
- FreeMemory
- ActiveThreadCount

----
jboss.jca:service=ManagedConnectionPool,name=jdbc/nitrogen

This service provides information about the database connection pool:

- MaxConnectionsInUseCount
- MaxSize
- InUseConnectionCount
- AvailableConnectionCount

---- jboss.web:type=RequestProcessor,*

This service provides information about slow HTTP requests that the server has processed:

- requestProcessingTime
- maxRequestUri
- maxTime
- processingTime

---- com.quest.nitro:service=Derivation

This is the derivation service. Its most useful attributes are:

- DerivationRuletteCount—the number of derivation rulettes.
- ComplexDerivationDefinitionCount—the number of derivation definitions.
- EvaluationCount—the number of derivation evaluations made. From release 5.2.3, this is a delta value for the previous 30 seconds. Prior to release 5.2.3, it was a cumulative count from when the server started, so it was not as useful.

This service also provides more detailed information for each derivation definition, and each derivation rulette, including the number of evaluations, the last time it was evaluated, the result of last evaluation, and so on.

Derivation Related Issues

This service is most often the cause of diagnostic snapshots that are 500 to 600 MB in size. Large file sizes indicate you may have an issue with an excessive number of derivations.

Identifying and Resolving Derivation Related Issues

Search the file for: `with .* derivation rulettes`. The results should resemble the following:

```
DATA_DRIVEN with 115 derivation rulettes

DATA_DRIVEN with 115 derivation rulettes

DATA_DRIVEN with 187230 derivation rulettes

DATA_DRIVEN with 97 derivation rulettes

DATA_DRIVEN with 115 derivation rulettes
```

The third line (`187230 derivation rulettes`) indicates an issue because the number of derivation rulettes is significantly larger than any other rulettes listed. Locate the complex derivation definition (located above the `with .* derivation rulettes` section in the file), and make a note of the topology type and metric name. For example:

```
Complex derivation definition: DBSS_Total_Elapsed_Time_Per_Exec (null) :
DerivationCalculation for DBSS_Top_Sql
```

where `DBSS_Top_Sql` is the topology type and `DBSS_Total_Elapsed_Time_Per_Exec` is the metric name.

If there is evidence of derivation problems, contact Quest Support with this information for further assistance and to determine if the issue has been resolved in the latest version of the affected cartridge.

---- com.quest.nitro:service=Topology

This is the topology service.

The most useful part of this service is in the extra information, which lists all topology types and, for each type, the number of instances, the number of instance versions, the maximum versions, and the effective instance versions. This information helps you determine if the topology is too large, or if there is a topology churn. Look for a high number of versions of instances, as well as a high maximum versions.

The topology table has the following six columns:

1. Topology Type—name of the topology object type.
2. Num Instance Version—number of versions of all instances of this topology type combined.
3. Max Version—version number of the single most changing instance.
4. Num Effective—number of active instances of this object.
5. Num Recent Versions—number of new versions of all instances in the last seven days.
6. Num Recent Instances—number of new instances created in the last seven days.

Topology Related Issues

Topology churn is defined as the constant changing and creation of new versions of existing topology objects. Each time a property is updated on an instance, a new version of that instance is created. Topology churn can cause high CPU usage as the Management Server propagates the changes across the rest of the topology model.

Topology growth is defined as the continuous creation of new instances of a type of topology object. Topology growth can cause high CPU usage as models and rulettes are updated, as well as increased JVM heap usage. The entire topology model is stored in memory, so as the number of objects added increases, so does the heap usage.

Identifying and Resolving Topology Churn and Growth

If the values in columns five and six in the table above are greater than 5000, examine the highest numbers and work your way down the list. Resolving issues with the higher ones can sometimes resolve other churn issues, since topology changes to one object can cause changes in other objects.

For example, consider the sample rows of the topology table below:

```
| DBO_Alert_Log | 76 | 2 | 38 | 76 | 38 |
```

This is an example of a good model. There are 38 instances (column 4), with a maximum of 2 versions (column 3), for a total of 76 versions (column 2). The numbers are in balance.

```
| DBO_Datafile | 5816 | 2 | 2908 | 2 | 1 |
```

This is also an example of a good, stable model. Even though the numbers are higher, 2908 x 2 = 5816, so the numbers are in balance. Additionally, in the last 7 days, there was only 1 new object, with 2 changes. There is no large growth or churn in this example.

```
| DBO_Undo_Activity_Info | 393761 | 16810 | 39 | 0 | 0 |
```

This is an example of a model that was bad but has become good. There are 393761 total versions in history, but no new changes (0) in the last 7 days.

```
| HostNetwork | 238231 | 4472 | 846 | 234543 | 42 |
```

This is a bad topology model. A large number (234543) of new versions have been created in the past 7 days.

```
| VMWESXServerPhysicalDisk | 28652 | 3 | 3530 | 10590 | 5295 |
```

This is also a bad model. In the past 7 days, 5295 new instances have been created. Column 4 indicates that some stale object cleanup has been done, but unless the root cause is found, the instances will keep being created.

If there is evidence of topology problems, contact Quest Support with this information for further assistance and to determine if the issue has been resolved in the latest version of the affected cartridge.

---- com.quest.nitro:service=DataCacheEviction

This service lists metrics that are being held in the JVM waiting to be written to the database permanently. This information is located in the Cache Policies section of the diagnostic snapshot.

If many (thousands) of metrics are held in memory for long periods of time, they cannot be cleaned up by a garbage collector (GC) because they are active/live objects. Therefore, a large portion of memory is used simply by data that should be written into the database instead. This lead to JVM heap exhaustion, and performance problems.

The following is an example of the Cache Policies section of the diagnostic snapshot:

```
Cache Policies:

cbc82b6a-1f8c-4fa8-a88a-fcb07af2854e:file_physical_io_pct -  age:259200000
granularity:300000 cached duration:259500000 num values:123 delay:192764

c25e1d2-c20b-400e-b87c-b9749c28899a:DBO_File_Avg_Read_Time_Ms -  age:259200000
granularity:300000 cached duration:259500000 num values:122 delay:43089

1a6c1537-3050-499d-9e93-1f702b1ab77f:file_read_time -  age:259200000
granularity:300000 cached duration:259500000 num values:140 delay:11026

1279f9c8-c72d-4deb-9080-73eeed73d70a:DBO_Datafile_File_Write_Requests_Rate -
age:259200000 granularity:300000 cached duration:259500000 num values:118
delay:12308

71d83485-c441-45e9-9cc6-ccd3bb510f71:file_physical_writes -  age:259200000
granularity:300000 cached duration:259500000 num values:136 delay:37018
```

Each line can be broken down as follows:

`cbc82b6a-1f8c-4fa8-a88a-fcb07af2854e`—topology object ID

`file_physical_io_pct`—name of the metric

`age:259200000`—length of time the metric is kept in memory (in ms)

`granularity:300000`—rawness of the metric value (in ms)

`num values:123`—number of values of this metric on this object

`delay:19276`—length of time the metric has been in memory

You can search the diagnostic snapshot for the metric name, and locate its parent topology in the XML schema. For example, for the metric detailed above:

```
<property name='file_physical_io_pct' type='Metric' is-many='false' is-
containment='true' unit-name='count'>
<annotation name='UnitEntityName' value='percents'/>
</property>
```

This metric is contained in the following XML tag:

```
<type name='DBO_Datafile_IO_Activity'
extends='DBO_Instance_Alarm_Object'>
```

This indicates that the `file_phyiscal_io_pct` metric is part of the DBO topology.

Contact Quest Support with this information for further assistance and to determine if the issue has been resolved in the latest version of the affected cartridge.

Analyzing a Performance Report

The Support Bundle contains a Management Server Performance Report (*PerfReport.pdf*), which can be helpful in diagnosing issues related to server performance.

Server Rule Information

Figure 1. Server Rule Information section of the performance report as a starting point for diagnosis

Server Rule Information

Rule Name	Alarm Count		
Agent Health State	0	1529	0
Catalyst Data Service Discarding Data	0	0	2
Catalyst Database Space Checking	0	0	0
Catalyst Free Database Space Checking	0	0	0
Catalyst License Monitoring	0	0	0
Catalyst Topology Size Limit	0	0	0
Foglight Garbage Collector Check	2	4270	3436
Foglight Memory Usage Check	0	0	0
Idle Agents	0	0	0
Remote Agent Managers State per Host	0	0	0
Remote Agent Managers State per Host [Group Customized]	0	0	456

Check the following:

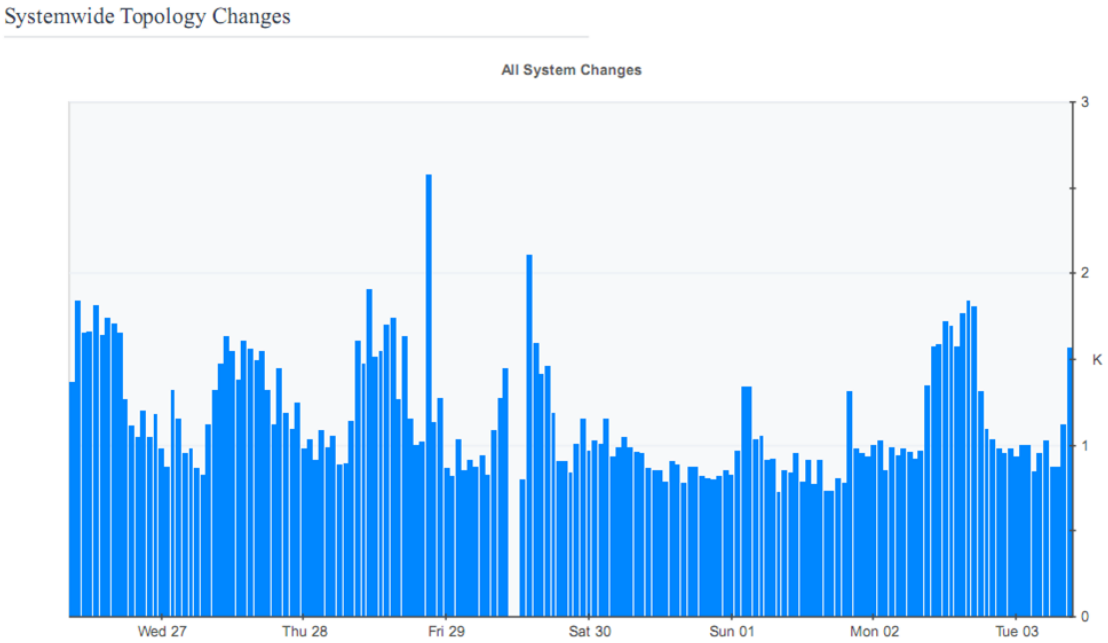
- Catalyst Topology Size Limit: When fired, this rule indicates that something is exceeding topology limits. It is a good indicator for topology growth.
- Foglight® Garbage Collector (GC) Check: Indicates JVM Heap usage problems. Large numbers of GC indicate the GC is working hard but doing little. Few or no alarms indicate the server is in a good state; hundreds of alarms indicate an issue.

- Foglight Memory Usage Check: Usually an indication of extreme JVM Heap Problems. When memory passes over a particular threshold, this alarm is fired.

System-wide Topology Changes

Topology changes should be considered over a 7-day span.

Figure 2. System-wide topology changes



Some cartridges generate new topology objects regularly and this is considered normal behavior. Such cartridges can produce hundreds of changes over a 7-day span.

If thousands of changes are generated, however, there may be an issue. In such cases, review the diagnostic snapshot to determine if there are Topology Related Issues.

JVM Memory Usage

The Support Bundle contains a Foglight® Management Server Performance Report (*PerfReport.pdf*). In the Performance Report, under the “Management Server Java Virtual Machine Memory” heading, there are a number of JVM memory charts.

In the JVM memory charts, the JVM heap is divided into New Generation, Old Generation, and Permanent Generation. The New Generation JVM heap is further divided into Eden and Survivor spaces. The charts display the memory utilization for each of these. The charts can help you determine whether or not the entire heap (Xms and Xmx), or just one generation (NewSize and MaxNewSize), is insufficient.

Typically, when objects are first created, they reside in the Eden space. Once objects survive a garbage collection (GC), they are moved into the Survivor space. If the garbage collector determines that the objects in the Survivor space are no longer live objects, then they are moved into the Old Generation. This helps the JVM efficiently manage its memory, because short-lived objects can easily be collected from the Eden space without the need for the garbage collector to scan the entire heap.

Typically, a sawtooth (up, down, up, down) pattern is a normal memory usage pattern for the Management Server. The server generates some garbage in the Old Generation, which is fine. Then, at some point, the JVM recycles the garbage.

A sudden drop in the Old Generation memory usage does not always indicate that a full GC has occurred, because the server now uses the parallel GC which runs in the background. Full GCs (during which the server slows down due to the garbage collection) are usually visible in the GC chart for `ConcurrentMarkSweep`, which is included in *PerfReport.pdf*. If the time or count lines rise above zero for a prolonged period, that indicates that the server is probably running out of memory. The GC becomes intense only when the sawtooth pattern hits 100%.

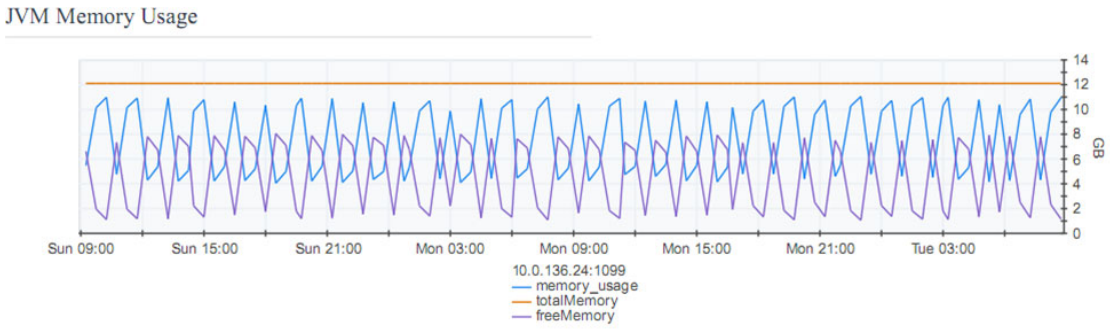
Permanent Generation is used for items like classes that are typically never collected.

If the Min/Max/Used line on the CMS Old Gen chart repeatedly hits the horizontal purple (Committed) line, further diagnosis (for example, matching with other performance metrics at the time) is required. That pattern typically indicates that the JVM was working extra hard to reclaim memory. Check for this situation by opening the Diagnostic Snapshot and searching for “Cache Policies”. This shows the entries that are in the cache and therefore consuming memory. If there is no list of entries, then there were no entries when the snapshot was taken.

To find the actual retention policy, open the Monitoring Policies XML file and search for `lifecycle definitions`.

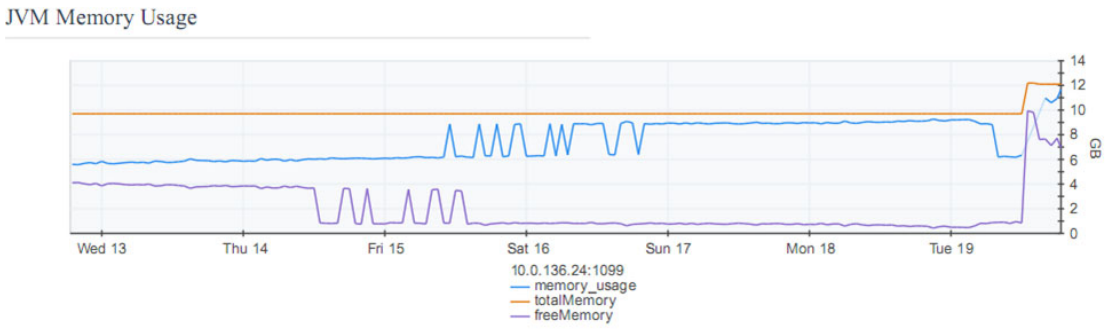
Figure 3. Some examples of JVM Memory Usage from the performance report

Good:



A consistent and regular pattern of memory being used and freed. Garbage collections are performing correctly.

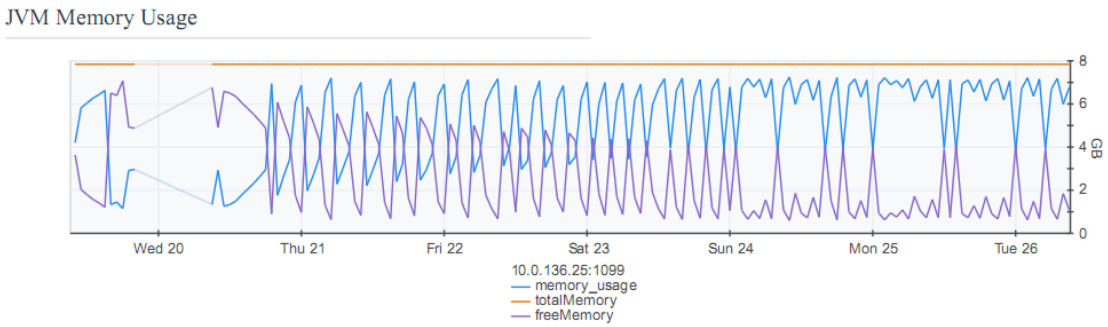
Bad:



A gradual exhaustion of the heap is indicated by slow and steady decline in the free memory, leading to a flat line. Here, no memory is being freed by GC.

Good, becoming Bad:

Figure 4. Gradual exhaustion of the heap



Part of the graph indicates a constant, regular pattern of memory being freed. The warning sign is that less memory is being freed each time in the later cycles.

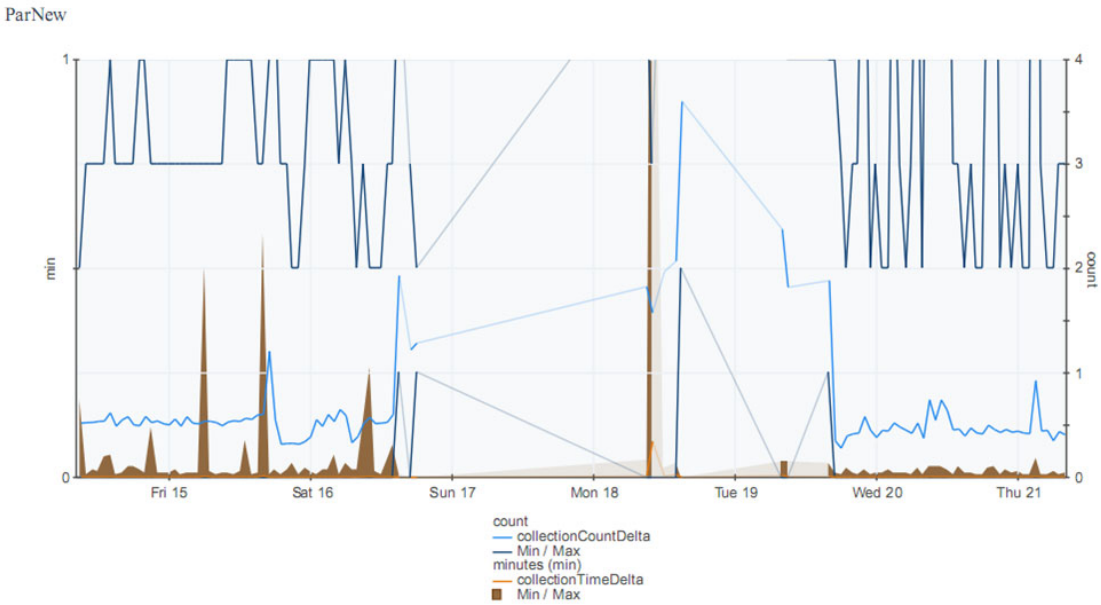
Management Server Garbage Collectors

There are two types of garbage collectors:

1. `ParNew`—short and easy collections done in parallel, these have a low impact on the Foglight® Management Server in normal operation.
2. `ConcurrentMarkSweep`—long and time-consuming collections that cause the JVM to pause everything else. The Management Server can appear to “freeze” during these cycles.

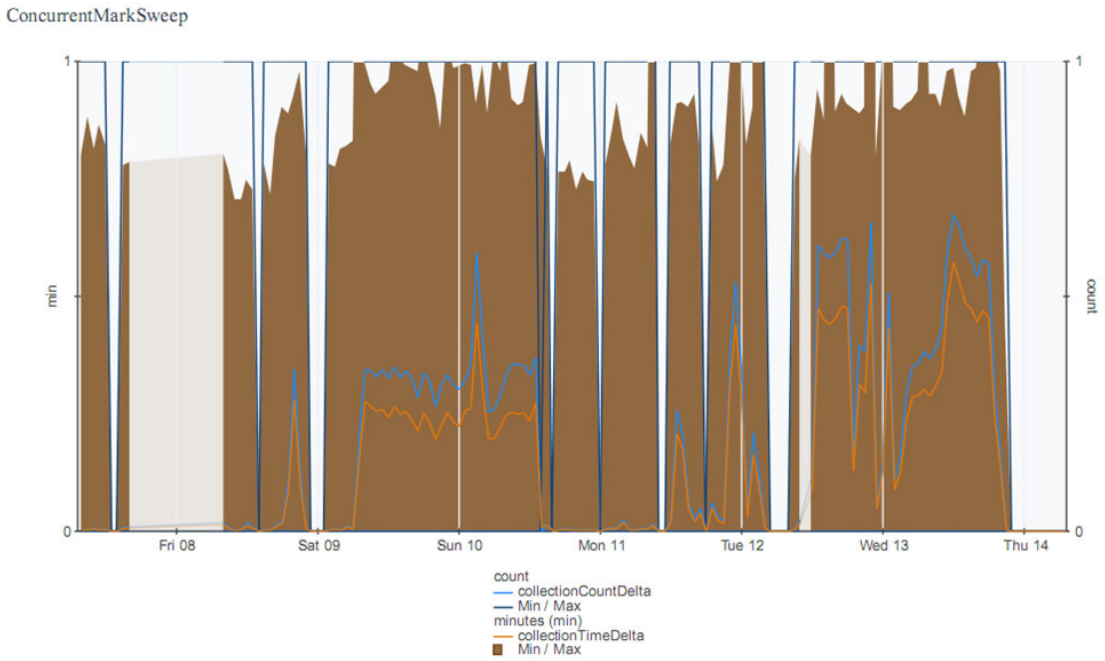
Examine the graphs for indications of issues.

Figure 5. Problems with the ParNew GC



The graph above indicates problems with the `ParNew` GC. The count (blue line) is up to 4, and the time (orange and brown) has increased to minutes.

Figure 6. Problems with the ConcurrentMarkSweep GC



The graph above indicates problems with the `ConcurrentMarkSweep` GC. The time (brown) has increased to minutes and the GC is called frequently.

These two graphs indicate the following issues:

- GCs are frequent and consuming a large portion of CPU time.
- High CPU usage is likely occurring.
- Memory problems are present, which can be resolved by increasing the heap size and/or debugging where the heap is being used. Common sources for excessive heap usage are: derivation rulettes, cached metrics, and topology objects.

NOTE: It is rare that the heap needs to be larger than 8 GB.

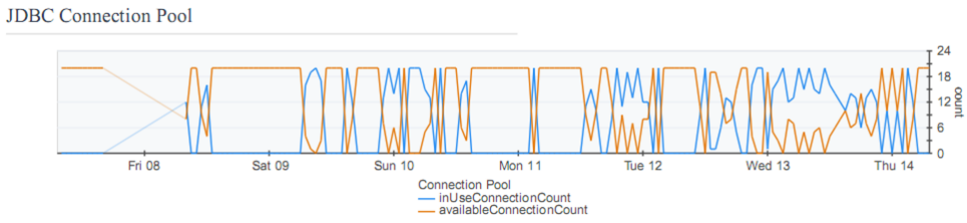
Increasing the heap size will likely seem to resolve an issue, but the root cause of the memory issue will soon grow to the new heap level, causing the issue to reappear.

Proper analysis of heap usage is necessary to ensure the root cause is resolved.

JDBC Connection Pool

The connection pool by itself does not indicate a problem, but can point to possible causes, such as: slow database, inefficient queries, or data intensive dashboards.

Figure 7. JDBC Connection Pool



If the available connections (orange line) flatlines at the bottom, the database connections are used for long periods of time, so no database connections can be made. The following error indicates no available connections:

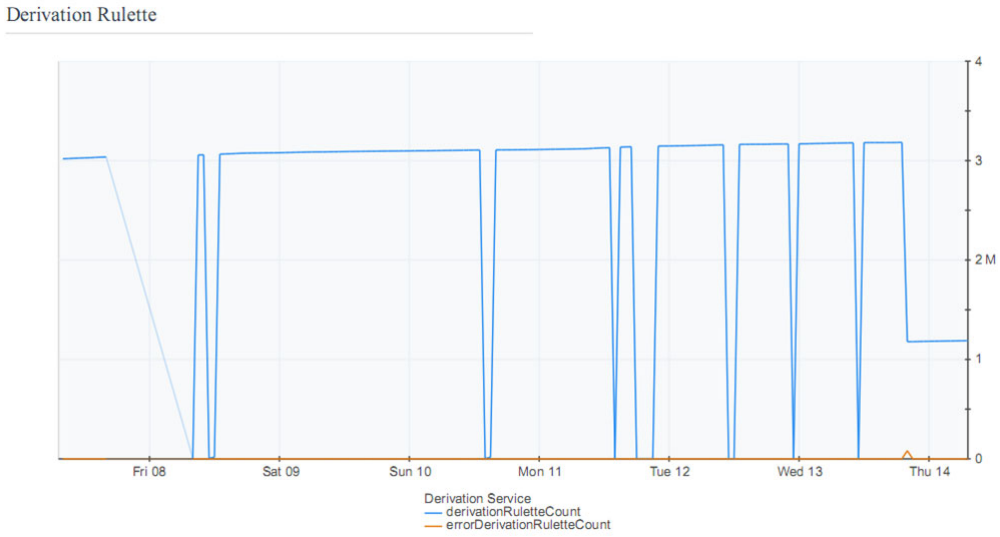
```
blocking timeout ( 30000 [ms] ); - nested
throwable:(javax.resource.ResourceException: No ManagedConnections available within
configured blocking timeout ( 30000 [ms] ))
```

Keep this in mind while investigating other issues.

Derivation Rulette

A derivation rulette is an instance of a derived metric definition that is tied to a particular Topology Object. Derivation rulettes take memory to store. Depending on the Foglight® Management Server version, anywhere from 4k (Management Server versions earlier than 5.5.5) to 1.5k (versions 5.5.5 and later). The more derivation rulettes you have, the more JVM heap is locked and cannot be freed, which leads to JVM heap exhaustion and performance problems.

Figure 8. Derivation rulette



Examine the count to determine whether there is an issue.

- Counts < 100k should not have much of an impact.
- Counts > 100k but < 1 million should be examined
- Counts in the millions can use gigabytes of the heap all by themselves.

Examine the diagnostic snapshot to determine the source of the problematic rulettes.

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.

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.