# Table of Contents

**Introduction to this Guide** ........................................................................................................................................ 7
- About Quest Software ........................................................................................................................................ 7
- Contacting Quest Software ............................................................................................................................. 7
- Contacting Quest Support .............................................................................................................................. 7

**Overview** ......................................................................................................................................................... 9
- Characteristics of Poor Performance in Foglight ............................................................................................. 9
- Parameters that Affect Foglight Performance ............................................................................................... 10
- Critical Areas that Can Affect Foglight Performance .................................................................................. 10
- Identifying Performance Problems ................................................................................................................ 10

**Hardware and Operating System Tuning** ....................................................................................................... 11
- Physical Memory ........................................................................................................................................... 11
- CPU ............................................................................................................................................................. 12
- I/O Throughput ........................................................................................................................................... 12
- File Handle Limits ....................................................................................................................................... 13
- Running on Virtual Hardware ...................................................................................................................... 13
  - Performance Challenges .......................................................................................................................... 13
  - CPU ......................................................................................................................................................... 14
  - Memory .................................................................................................................................................. 14
  - Shares .................................................................................................................................................... 15
  - Disk ........................................................................................................................................................ 15

**Management Server Tuning** ............................................................................................................................. 16
- Alarm Limit ................................................................................................................................................... 16
- Topology Limit ............................................................................................................................................ 17
- Number of CPUs Used for Thread Pools .................................................................................................... 18
- Dashboard Default Timeout ......................................................................................................................... 18

**Java Virtual Machine Tuning** .......................................................................................................................... 19
- Foglight JVM Configuration ......................................................................................................................... 20
- JVM Options ............................................................................................................................................... 20
- Common Symptoms and Tuning Resolutions ............................................................................................ 23
- JVM Code Cache ......................................................................................................................................... 24

**Backend Database Tuning** .............................................................................................................................. 26
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Database Configuration Settings</td>
<td>26</td>
</tr>
<tr>
<td>Monitoring and Managing Database Size</td>
<td>26</td>
</tr>
<tr>
<td>Backup and Recovery Recommendations</td>
<td>26</td>
</tr>
<tr>
<td>init.ora Configuration</td>
<td>26</td>
</tr>
<tr>
<td>Database Maintenance Recommendations</td>
<td>27</td>
</tr>
<tr>
<td>MySQL Tuning</td>
<td>27</td>
</tr>
<tr>
<td>Case</td>
<td>27</td>
</tr>
<tr>
<td>Oracle Tuning</td>
<td>29</td>
</tr>
<tr>
<td>How the Management Server Uses the Database</td>
<td>29</td>
</tr>
<tr>
<td>Index Management</td>
<td>29</td>
</tr>
<tr>
<td>Oracle’s Database Analysis</td>
<td>30</td>
</tr>
<tr>
<td>Memory</td>
<td>30</td>
</tr>
<tr>
<td>Block Size</td>
<td>30</td>
</tr>
<tr>
<td>Oracle Striping</td>
<td>31</td>
</tr>
<tr>
<td>Oracle Tablespaces</td>
<td>31</td>
</tr>
<tr>
<td>Microsoft SQL Server Tuning</td>
<td>34</td>
</tr>
<tr>
<td>High Availability (HA) Tuning</td>
<td>35</td>
</tr>
<tr>
<td>Tuning Connection Issues in an HA Implementation</td>
<td>35</td>
</tr>
<tr>
<td>Managing Hosts with Multiple Network Interfaces</td>
<td>36</td>
</tr>
<tr>
<td>JDK with IPv6 on Linux</td>
<td>37</td>
</tr>
<tr>
<td>Management Server Automatically Restarted</td>
<td>37</td>
</tr>
<tr>
<td>Other JGroup Related Issues and Information</td>
<td>37</td>
</tr>
<tr>
<td>Agent Tuning</td>
<td>39</td>
</tr>
<tr>
<td>Topology Changes and Topology Churn</td>
<td>39</td>
</tr>
<tr>
<td>Canonical Data Transformations (CDTs)</td>
<td>40</td>
</tr>
<tr>
<td>Agent Weight / Environment Complexity</td>
<td>41</td>
</tr>
<tr>
<td>Large Topologies</td>
<td>41</td>
</tr>
<tr>
<td>Sampling Frequency</td>
<td>42</td>
</tr>
<tr>
<td>XML-HTTP Agent Adapter</td>
<td>42</td>
</tr>
<tr>
<td>Dropped Agent Manager Log Messages</td>
<td>43</td>
</tr>
<tr>
<td>Java EE Technologies</td>
<td>43</td>
</tr>
<tr>
<td>Appendix: Performance Health Check</td>
<td>44</td>
</tr>
<tr>
<td>Is the Server Getting the Right Amount of Memory</td>
<td>44</td>
</tr>
<tr>
<td>Management Server Memory Checks</td>
<td>44</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>45</td>
</tr>
<tr>
<td>Is the Server Getting Enough CPU?</td>
<td>45</td>
</tr>
<tr>
<td>Management Server CPU is Healthy if</td>
<td>45</td>
</tr>
<tr>
<td>Management Server CPU Checks</td>
<td>45</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Foglight: Performance Tuning Field Guide</td>
<td></td>
</tr>
<tr>
<td>Server Log</td>
<td>52</td>
</tr>
<tr>
<td>Diagnostic Snapshot</td>
<td>53</td>
</tr>
<tr>
<td>Is the Database too Slow?</td>
<td>46</td>
</tr>
<tr>
<td>The Database is Healthy if</td>
<td>46</td>
</tr>
<tr>
<td>Database Checks</td>
<td>47</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>47</td>
</tr>
<tr>
<td>Is the Database Growth Reasonable?</td>
<td>48</td>
</tr>
<tr>
<td>Database Growth is Healthy if</td>
<td>48</td>
</tr>
<tr>
<td>Database Checks</td>
<td>48</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>48</td>
</tr>
<tr>
<td>Is There Too Much Data to Process?</td>
<td>48</td>
</tr>
<tr>
<td>Data Volume is Healthy if</td>
<td>49</td>
</tr>
<tr>
<td>Data Volume Checks</td>
<td>49</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>49</td>
</tr>
<tr>
<td>Is the Model Stable?</td>
<td>49</td>
</tr>
<tr>
<td>The Model is Healthy if</td>
<td>49</td>
</tr>
<tr>
<td>Model Checks</td>
<td>49</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>49</td>
</tr>
<tr>
<td>Are Too Many Alarms Firing?</td>
<td>50</td>
</tr>
<tr>
<td>The Number of Alarms is in a Healthy State if</td>
<td>50</td>
</tr>
<tr>
<td>Alarms Checks</td>
<td>50</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>50</td>
</tr>
<tr>
<td>Is the Business Logic Properly Tuned?</td>
<td>50</td>
</tr>
<tr>
<td>Business Logic is Healthy if</td>
<td>50</td>
</tr>
<tr>
<td>Business Logic Checks</td>
<td>50</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>50</td>
</tr>
<tr>
<td>Are There Too Many User Requests?</td>
<td>51</td>
</tr>
<tr>
<td>User Activity Is Healthy If</td>
<td>51</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>51</td>
</tr>
<tr>
<td>Appendix: Analyzing a Support Bundle</td>
<td>52</td>
</tr>
<tr>
<td>Extended Browser Interface Checks</td>
<td>46</td>
</tr>
<tr>
<td>Possible Actions</td>
<td>46</td>
</tr>
<tr>
<td>Memory Consumption</td>
<td>47</td>
</tr>
<tr>
<td>Topology Sync</td>
<td>52</td>
</tr>
<tr>
<td>Topology Limits</td>
<td>52</td>
</tr>
<tr>
<td>Threads, Deadlocks, and Overall CPU Usage</td>
<td>53</td>
</tr>
<tr>
<td>Useful Information</td>
<td>54</td>
</tr>
<tr>
<td>---- jboss.system:type=ServerInfo</td>
<td>54</td>
</tr>
<tr>
<td>---- jboss.jca:service=ManagedConnectionPool,name=jdbc/nitrogen</td>
<td>54</td>
</tr>
<tr>
<td>---- jboss.web:type=RequestProcessor,*</td>
<td>54</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>---- com.quest.nitro:service=Derivation</td>
<td>54</td>
</tr>
<tr>
<td>---- com.quest.nitro:service=Topology</td>
<td>55</td>
</tr>
<tr>
<td>---- com.quest.nitro:service=DataCacheEviction</td>
<td>57</td>
</tr>
<tr>
<td>Analyzing a Performance Report</td>
<td>58</td>
</tr>
<tr>
<td>Server Rule Information</td>
<td>58</td>
</tr>
<tr>
<td>System-wide Topology Changes</td>
<td>58</td>
</tr>
<tr>
<td>JVM Memory Usage</td>
<td>59</td>
</tr>
<tr>
<td>Management Server Garbage Collectors</td>
<td>61</td>
</tr>
<tr>
<td>JDBC Connection Pool</td>
<td>62</td>
</tr>
<tr>
<td>Derivation Rulette</td>
<td>63</td>
</tr>
<tr>
<td>Index</td>
<td>64</td>
</tr>
</tbody>
</table>
Introduction to this Guide

The *Performance Tuning Field Guide* is intended for anyone who wants to tune Foglight to improve performance.

The guide provides information about the various things that can have an effect on overall performance and describes the applicable performance-related options.

About Quest Software, Inc.

Established in 1987, Quest Software (Nasdaq: QSFT) provides simple and innovative IT management solutions that enable more than 100,000 global customers to save time and money across physical and virtual environments. Quest products solve complex IT challenges ranging from database management, data protection, identity and access management, monitoring, user workspace management to Windows management. For more information, visit [www.quest.com](http://www.quest.com).

Contacting Quest Software

<table>
<thead>
<tr>
<th>Email</th>
<th><a href="mailto:info@quest.com">info@quest.com</a></th>
</tr>
</thead>
</table>
| Mail        | Quest Software, Inc.  
World Headquarters  
5 Polaris Way  
Aliso Viejo, CA  92656  
USA           |
| Web site    | [www.quest.com](http://www.quest.com) |

Refer to our Web site for regional and international office information.

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a Quest product and have a valid maintenance contract. Quest Support provides unlimited 24x7 access to our Support Portal at [http://www.quest.com/support](http://www.quest.com/support).
From our Support Portal, you can do the following:

- Retrieve thousands of solutions from our Knowledge Base
- Download the latest releases and service packs
- Create, update, and review Support cases

View the Global Support Guide for a detailed explanation of support programs, online services, contact information, policies, and procedures. The guide is available at: [http://www.quest.com/support](http://www.quest.com/support).
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

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 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 44.

2. You can analyze a support bundle to identify problems. For more information, see “Appendix: Analyzing a Support Bundle” on page 52.
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 Chapter 4, “Java Virtual Machine Tuning”).

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

## Physical Memory

<table>
<thead>
<tr>
<th><strong>Motivation</strong></th>
<th>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).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Symptom</strong></td>
<td>The browser interface performance is poor (that is, it responds slowly).</td>
</tr>
</tbody>
</table>
| **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.                                                                               |
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. |

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). |
| 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. |
File Handle Limits

<table>
<thead>
<tr>
<th>Motivation</th>
<th>Foglight requires a minimum number of file handles. The native launcher checks for a minimum number of file handles, but problems can still occur.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>The JVM reports a java.lang.OutOfMemoryError.</td>
</tr>
<tr>
<td>Diagnosis / Verification</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 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 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.
Chapter 2—Hardware and Operating System Tuning

CPU

Before you run the 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.

Note

Ideally, the core CPU count on the physical host should be 2 to 3 times the number of CPUs that you have planned for the VM. Otherwise, CPU scheduling issues might arise.

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 a number of CPUs to be free.

Memory

Before you run the 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

```java
---- 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 53.

 Shares

For the 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 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.
Management Server Tuning

The parts of the 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

<table>
<thead>
<tr>
<th>Motivation</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>The browser interface performance is poor (that is, it responds slowly), especially when displaying the alarms table.</td>
</tr>
<tr>
<td>Diagnosis / Verification</td>
<td>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.</td>
</tr>
</tbody>
</table>
| Tuning | The alarm limit may need to be reduced.  
MBean: `*:service=Alarm`  
Attribute: `MaxAlarms`  
Expected old value: 10000  
New value: 5000 |
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;
```

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.

---

**Topology Limit**

<table>
<thead>
<tr>
<th><strong>Motivation</strong></th>
<th>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.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Symptom</strong></th>
<th>The browser interface performance is poor (that is, it responds slowly). The server memory usage is high. The server may be overloaded.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Diagnosis / Verification</strong></th>
<th>Examine the topology instance counts shown in a support bundle.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Tuning</strong></th>
<th>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 <code>foglight.limit.instances</code> registry variable.</th>
</tr>
</thead>
</table>

| **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. |
Number of CPUs Used for Thread Pools

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.
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 Oracle’s Java Hotspot Virtual Machine (JVM). While there are other vendors of JVMs, Oracle’s 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.

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**  
The value must be enclosed in quotation marks and followed by a semicolon.

---

**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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-XX:+UseConcMarkSweepGC</code></td>
<td>This switch enables a concurrent low-pause garbage collector.</td>
</tr>
<tr>
<td><strong>Default</strong></td>
<td>Enabled, since release 5.2.3.</td>
</tr>
<tr>
<td><strong>Recommendation</strong></td>
<td>Do not change this setting unless advised by Quest Support.</td>
</tr>
</tbody>
</table>

#### -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`

**Note**  
-Xmx and -Xms should be set to the same value.
Default

32-bit: 256m < 75% avail < 1024m
64-bit: 512m < 75% avail < 4096m*

Note There was no maximum prior to release 5.5.4.

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).

Note Be careful not to configure the heap to be smaller than Foglight requires.

-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.


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)

**Note** This has been the NewRatio default since release 5.5.2. These are the defaults for all three switches because it has been proven that `-XX:NewRatio` alone does not set the young object space correctly in all situations.

**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:

-XX:s=512k
-XX:s=1m
Default

- Xss=256k

Recommendation

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

Note

Foglight utilizes hundreds of threads, which is a multiplying factor in the overall tax on memory.

Common Symptoms and Tuning Resolutions

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

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Diagnosis/Verification</th>
<th>Tuning</th>
</tr>
</thead>
</table>
| • 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. |
Chapter 4—Java Virtual Machine Tuning

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.

<table>
<thead>
<tr>
<th>Symptoms</th>
<th>Diagnosis/Verification</th>
<th>Tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• On a 32-bit Windows operating system, the server logs show java.lang.OutOfMemory errors.</td>
<td>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.</td>
<td>Consider reducing the memory allocated by Foglight using the <code>-Xmx1G</code> switch. Alternatively, consider reducing the amount of memory allocated per thread using the <code>-Xss=512k</code> or <code>-Xss=256k</code> switch.</td>
</tr>
<tr>
<td>• On a 32-bit Windows operating system, the server logs show java.lang.OutOfMemoryError:unable to create new native thread.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 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.</td>
<td>Foglight’s launcher process utilizes Java DLLs, instead of running <code>java.exe</code> as <code>run.bat</code> does. Internal memory parameters such as ‘size of contiguous free memory’ or ‘memory layout’ can lead to related problems and/or limitations.</td>
<td>Customers who plan to have memory requirements that are greater than 800 MB should consider running Foglight on a 64-bit operating system.</td>
</tr>
<tr>
<td>• On a 32-bit Windows operating system, the server output displays a ‘Could not reserve enough space for object heap’ message.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
According to our analysis, when the code cache is filling up, there is no significant change in CPU utilization. When the code cache is filling up, the JVM may simply be waiting to exchange code in the cache with newer compiler code, to achieve the best possible performance.

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:**

1. Open `<foglight_home>/config/server.config`.
2. Edit the `server.vm.option0` line to read:
   ```
   server.vm.option0 = "-XX:ReservedCodeCacheSize=128m";
   ```
3. Restart the Management Server.
Backend Database Tuning

The 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

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 Guide.

Backup and Recovery Recommendations

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

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>
<thead>
<tr>
<th>Motivation</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>The browser interface performance is poor (that is, it responds slowly).</td>
</tr>
<tr>
<td>Diagnosis / Verification</td>
<td>The diagnostic snapshot for the Management Server shows that it is overloaded.</td>
</tr>
<tr>
<td>Tuning</td>
<td>Increase the <code>innodb_buffer_pool_size</code> to 1 GB or more (up to 80 percent of the free memory). The default is 512 MB.</td>
</tr>
</tbody>
</table>


**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 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 31.
- The Management Server alarm tables are frequent inserts (alarm_* , tablespace group A). For more information, see “Oracle Tablespaces” on page 31.
- The Management Server topology tables are less frequent inserts/updates (topology_* , tablespace group A). For more information, see “Oracle Tablespaces” on page 31.

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 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.

**Important** Adding or altering indices, or modifying the database content that the Management Server manages in any other way, is not supported.
# Oracle's 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. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>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.</td>
</tr>
<tr>
<td>Diagnosis / Verification</td>
<td>The DBA should use Oracle analysis tools to check for strategy effectiveness.</td>
</tr>
</tbody>
</table>
| 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 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 29.
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

<table>
<thead>
<tr>
<th>Note</th>
<th>You should refer to Oracle's documentation for more detailed information on tablespaces.</th>
</tr>
</thead>
</table>

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.

The 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).

<table>
<thead>
<tr>
<th>Note</th>
<th>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.</th>
</tr>
</thead>
</table>

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:

\[
\text{UndoSpace} = UR \times UPS + \text{overhead}
\]

where:

- \text{UndoSpace} is the number of undo blocks
- \text{UR} is the UNDO_RETENTION value in seconds
- \text{UPS} is the number of undo blocks per second, or the transaction rate
- \text{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:
   ```xml
   <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:
   ```xml
   <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;
```

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

The 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.
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**

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:

   ```
   ```

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 restart_monitor.config file, find the line beginning with
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

Issue: 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
```


### 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 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](https://jira.jboss.org/browse/JGRP-798)

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

[https://jira.jboss.org/browse/JGRP-614](https://jira.jboss.org/browse/JGRP-614)
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

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>
<thead>
<tr>
<th>Motivation</th>
<th>The server is optimized to handle stable topology models. Therefore, it is expected that the Management Server is configured to minimize topology changes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Data is being dropped by the Data Service.</td>
</tr>
</tbody>
</table>
Theoretically, CDTs can be a performance bottleneck. Unfortunately, it is not easy to tune them. In most cases, a cartridge update is required.

**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

The 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 \textit{aggregateAgentWeight}. It is available from the EnvComplexityEstimator service in the Management Server Metrics dashboard: \texttt{Dashboards > Foglight > Servers > Management Server Metrics > \langle\texttt{CatalystServer}\rangle > Services > com.quest.nitro:service=EnvComplexityEstimator > aggregateAgentWeight}.

This metric is derived from the number and types of connected agents according to: 
\texttt{<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.

\textbf{Note} \quad 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>
<thead>
<tr>
<th>Motivation</th>
<th>On occasion, agents may send so much observation data that the resulting topology model becomes too large and causes performance problems in the server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>Java EE Technologies agents are the most likely to cause this type of situation, if they are not carefully pre-configured.</td>
</tr>
<tr>
<td>Symptom</td>
<td>The server is in overload condition. Agent data is being dropped. The browser interface performance is poor (that is, it responds slowly).</td>
</tr>
<tr>
<td>Diagnosis/ Verification</td>
<td>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.</td>
</tr>
<tr>
<td>Note</td>
<td>Acceptable count ranges will differ depending on the amount of resources allocated to the Management Server.</td>
</tr>
</tbody>
</table>
Chapter 7—Agent Tuning

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>
<thead>
<tr>
<th>Motivation</th>
<th>The XML-HTTP agent adapter rejects large XML submissions to protect the server from memory overload.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symptom</td>
<td>An agent that uses the XML-HTTP protocol complains that the server is rejecting its data.</td>
</tr>
<tr>
<td>Diagnosis/Verification</td>
<td>Check the agent logs for specific error messages indicating that the data was rejected due to size constraints.</td>
</tr>
<tr>
<td>Tuning</td>
<td>Open web.xml, which is located in <code>&lt;foglight_home&gt;/cartridge/XmlHttpAgentAdapter-*.car</code>, component <code>XmlHttpAgentAdapter-5_2_4/WebApp-foglight-xml-agent-adapter.war</code>, file <code>WEB-INF/web.xml</code>, and adjust the <code>maxContentLength</code> parameter.</td>
</tr>
<tr>
<td>Note 1</td>
<td>You will need to unzip the <code>.car</code> file, then unzip the <code>.war</code> file, edit the <code>web.xml</code> file, and then re-package the cartridge manually (using zip).</td>
</tr>
</tbody>
</table>
Dropped Agent Manager Log Messages

The default settings for the 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 Cartridge for Java EE Technologies documentation.

Note 2
After you modify the cartridge, you will have to delete <foglight_home>/state/cartridge.exploded/XmlHttpAgentAdapter-*, and then restart the Management Server.
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?

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

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

  ```sql
  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:

  ```bash
  ./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 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.
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

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:

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

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).
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-9e93-1f702bab77f:file_read_time - age:259200000 granularity:300000 cached duration:259500000 num values:140 delay:11026

1279f9c8-c72d-4deb-9080-73eed73d70a:DBO_Datafile_File_Write_Requests_Rate - age:259200000 granularity:300000 cached duration:259500000 num values:118 delay:12308

```

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:

```xml
<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:

```xml
<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**

The Server Rule Information section of the performance report can provide a starting point for diagnosis.

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.
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 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.

Some examples of JVM Memory Usage from the performance report:

Good:

![JVM Memory Usage Graph](image1)

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

Bad:

![JVM Memory Usage Graph](image2)

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:

![JVM Memory Usage Graph](image)

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 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.

![ParNew Graph](image)

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.
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.
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 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.

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.
Index

A
Agent Manager
log messages
dropped 43
agents
Agent Manager log messages
dropped 43
CDTs 40
environment complexity 41
Java EE 43
large topologies 41
performance tuning 39
sampling frequency 42
topology changes 39
topology churn 39
weight 41
XML-HTTP adapter 42
alarm limit
Management Server 16
alarms, excess 50

B
business logic
performance tuning 50

C
CDTs
agents 40
code cache
Java Virtual Machine 24
console
default timeout 18
CPU
Management Server 45
performance tuning 12
virtual machine 14
CPUs
Management Server 18

data, excess 48
database
backup 26
configuration settings 26
growth 48
init.ora 26
managing 26
monitoring 26
MySQL performance tuning 27
Oracle performance tuning 29
recovery 26
speed 46
diagnostic snapshot 53
disk, virtual machine 15

evironment complexity
agents 41
excess
alarms 50
data 48

F
file handle limits
performance tuning 13

H
hardware
performance tuning 11
high availability
connection issues 35
performance tuning 35
I
I/O throughput
  performance tuning 12

J
Java EE
  agents 43
Java Virtual Machine
  code cache 24
  configuration 20
  performance tuning 20, 23
  resolutions 23
  symptoms 23
JVM memory
  support bundle charts 59
JVM options 20

L
large topologies
  agents 41

M
Management Server
  alarm limit 16
  CPU 45
  CPUs 18
  memory 44
  performance tuning 16
  thread pools 18
  topology limit 17
Management Server on a virtual machine
  performance tuning 13
memory
  Management Server 44
  virtual machine 14
model
  stability 49

O
operating system
  performance tuning 11
overview 9

P
performance
  affecting areas 10
  affecting parameters 9
  identifying problems 10
  poor 9
performance health check 44
performance tuning
  agents 39
  business logic 50
  cpu 12
  file handle limits 13
  hardware 11
  high availability 35
  I/O throughput 12
  Java Virtual Machine 20, 23
  Management Server 16
  Management Server on a virtual machine 13
  MySQL database 27
  operating system 11
  Oracle database 29
  overview 9
  physical memory 11
  SQL Server 34
PerfReport.pdf
  JVM memory charts 59
physical memory
  performance tuning 11

R
ready time
  virtual machine 14

S
sampling frequency
  agents 42
server log
  52
services 54
shares
  virtual machine 15
SQL Server
  tuning 34
support 8
support bundle
  analyzing 52
  diagnostic snapshot 53
  JVM memory charts
    new generation 59
Index

old generation 59
permanent generation 59
new generation
eden space 59
survivor space 59
PerfReport.pdf
JVM memory charts 59
server log 52
services 54
switches 20

T
technical support 8
thread pools
Management Server 18
topology changes
agents 39
topology churn
agents 39
topology limit
Management Server 17
tuning
agents 39
CPU 12
file handle limits 13
hardware 11
high availability 35
I/O throughput 12
Java Virtual Machine 20, 23
Management Server 16
Management Server on a virtual machine 13
MySQL database 27
operating system 11
Oracle database 29
overview 9
physical memory 11
SQL Server 34
tuning business logic 50

U
user requests
too many 51

V
virtual machine 13
running on

CPU 14
disk 15
memory 14
performance challenges 13
ready time 14
shares 15

W
wcf.js
default timeout 18
weight
agents 41

X
XML-HTTP adapter
agents 42