Zenoss Core Extended Monitoring

Copyright © 2014 Zenoss, Inc. All rights reserved.

Zenoss and the Zenoss logo are trademarks or registered trademarks of Zenoss, Inc., in the United States and other countries. All other trademarks, logos, and service marks are the property of Zenoss or other third parties. Use of these marks is prohibited without the express written consent of Zenoss, Inc., or the third-party owner.

Amazon Web Services, AWS, Amazon Elastic Compute Cloud, and Amazon EC2 are trademarks of Amazon.com, Inc. or its affiliates in the United States and/or other countries.

Flash is a registered trademark of Adobe Systems Incorporated.

Oracle, the Oracle logo, Java, and MySQL are registered trademarks of the Oracle Corporation and/or its affiliates.

Linux is a registered trademark of Linus Torvalds.

SNMP Informant is a trademark of Garth K. Williams (Informant Systems, Inc.).

Sybase is a registered trademark of Sybase, Inc.

Tomcat is a trademark of the Apache Software Foundation.

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

All other companies and products mentioned are trademarks and property of their respective owners.

Part Number: 03-032014-4.2-v10
About this guide

This guide describes the ZenPacks that are distributed with, or available for, Zenoss Core.

This release includes new ZenPacks that deprecate previous ZenPacks. Information about both the new and deprecated ZenPacks are included, to facilitate transition to the new ZenPacks.

Unlike previous editions of this guide, chapters are ordered by ZenPack name.

Two new appendices are included.

Administering ZenPacks on page 90

This appendix includes procedures for installing, upgrading, and removing ZenPacks.

Zenoss Core daemons on page 91

This appendix includes descriptions of the daemons that are always present in Zenoss Core, and the daemons that are installed when specific ZenPacks are installed.

ZenPack Index

The following table lists the ZenPacks that are included in this release of Zenoss Core. The table includes

- The ZenPack name and a link to its documentation, if any
- A brief description of the ZenPack
- A one-character code indicating the ZenPack type
  - P: Platform ZenPacks add features to Zenoss Core or enhance existing features.
  - M: Monitoring ZenPacks add device modeling and data collection capabilities for specific device types.

Some platform ZenPacks are undocumented because they provide basic, background services.

<table>
<thead>
<tr>
<th>Name and location</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ActiveDirectory) Active Directory on page 6</td>
<td>The ZenPacks.zenoss.ActiveDirectory ZenPack uses WMI to monitor Microsoft Active Directory authentication metrics. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td>(ApacheMonitor) Apache HTTP Server on page 8</td>
<td>The ZenPacks.zenoss.ApacheMonitor ZenPack monitors Apache HTTP Server by collecting metrics through the mod_status module.</td>
<td>M</td>
</tr>
<tr>
<td>(DellMonitor) Dell Hardware on page 13</td>
<td>The ZenPacks.zenoss.DellMonitor ZenPack provides customized modeling of devices running Dell OpenManage agents, and includes identification for proprietary Dell hardware.</td>
<td>M</td>
</tr>
<tr>
<td>(DeviceSearch)</td>
<td>The Zenpacks.zenoss.DeviceSearch ZenPack is undocumented.</td>
<td>P</td>
</tr>
<tr>
<td>(DigMonitor) Dig Monitor on page 15</td>
<td>The ZenPacks.zenoss.DigMonitor ZenPack monitors the response time of DNS lookups.</td>
<td>M</td>
</tr>
<tr>
<td>(DnsMonitor) DNS Monitor on page 17</td>
<td>The ZenPacks.zenoss.DnsMonitor ZenPack monitors the response time of DNS requests.</td>
<td>M</td>
</tr>
<tr>
<td>Name and location</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td><em>(EsxTop)</em> VMware ESX Server on page 19</td>
<td>The ZenPacks.zenoss.EsxTop ZenPack uses the VMware resxtop command to gather performance information about VMware Infrastructure™ ESX™ servers. NOTE: This ZenPack is deprecated; see <em>(vSphere)</em> VMware vSphere.</td>
<td>M</td>
</tr>
<tr>
<td><em>(HPMonitor)</em> HP Monitor on page 25</td>
<td>The ZenPacks.zenoss.HPMonitor ZenPack provides customized modeling of devices running HP Insight Management Agents, and includes identification for proprietary HP hardware.</td>
<td>M</td>
</tr>
<tr>
<td><em>(HttpMonitor)</em> HTTP Monitor on page 27</td>
<td>The ZenPacks.zenoss.HttpMonitor ZenPack monitors the response times of HTTP server connection requests, and determines whether specific content exists on a Web page.</td>
<td>M</td>
</tr>
<tr>
<td><em>(IISMonitor)</em> Microsoft Internet Information Server on page 30</td>
<td>The ZenPacks.zenoss.IISMonitor ZenPack uses Windows Perfmon to collect performance metrics from Microsoft Internet Information Server (IIS). NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see <em>(Microsoft.Windows)</em> Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td><em>(IRCDMonitor)</em> IRCD Monitor on page 32</td>
<td>The ZenPacks.zenoss.IRCDMonitor ZenPack monitors the number of users connected to an Internet Relay Chat (IRC) server.</td>
<td>M</td>
</tr>
<tr>
<td><em>(JabberMonitor)</em> Jabber Monitor on page 34</td>
<td>The ZenPacks.zenoss.JabberMonitor ZenPack monitors the response times of Jabber instant messaging servers.</td>
<td>M</td>
</tr>
<tr>
<td><em>(LinuxMonitor)</em> Linux Monitor on page 38</td>
<td>The ZenPacks.zenoss.LinuxMonitor ZenPack demonstrates how to develop new plugins that collect performance data using Secure Shell.</td>
<td>P</td>
</tr>
<tr>
<td><em>(MSExchange)</em> Microsoft Exchange on page 53</td>
<td>The ZenPacks.zenoss.MSExchange ZenPack uses WMI to monitor Microsoft Exchange and related services. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see <em>(Microsoft.Windows)</em> Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td><em>(MSMQMonitor)</em> Microsoft Message Queueing on page 55</td>
<td>The ZenPacks.zenoss.MSMQMonitor ZenPack uses WMI to automatically discover Microsoft Message Queuing (MSMQ) queues, and monitor the number of messages queued in each. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see <em>(Microsoft.Windows)</em> Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td>Name and location</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>(MSSQLServer) Microsoft SQL Server on page 57</td>
<td>The ZenPacks.zenoss.MSSQLServer ZenPack uses WMI to monitor Microsoft SQL Server and its related services. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td>(MySqlMonitor) MySQL Database Monitor on page 59</td>
<td>The ZenPacks.zenoss.MySqlMonitor ZenPack monitors MySQL database servers through the Python twisted.enterprise.adbapi asynchronous framework.</td>
<td>M</td>
</tr>
<tr>
<td>(NNTPMonitor) NNTP Monitor on page 64</td>
<td>The ZenPacks.zenoss.NNTPMonitor ZenPack monitors the response time of Network News Transfer Protocol (NNTP) servers.</td>
<td>M</td>
</tr>
<tr>
<td>(NtpMonitor) NTP Monitor on page 66</td>
<td>The ZenPacks.zenoss.NtpMonitor ZenPack monitors the difference between the system time a server is using and the time a Network Time Protocol (NTP) server is reporting.</td>
<td>M</td>
</tr>
<tr>
<td>(PySamba)</td>
<td>The Zenpacks.zenoss.PySamba ZenPack provides the Python interface to the Samba C library and executable targets used by other ZenPacks for Windows device monitoring. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.</td>
<td>P</td>
</tr>
<tr>
<td>Zenpacks.zenoss.PythonCollector</td>
<td>This ZenPack provides a Python data source type and polling daemon (zenpython) for use in customized ZenPacks.</td>
<td>P</td>
</tr>
<tr>
<td>ZenPacks.zenoss.WBEM</td>
<td>The ZenPacks.zenoss.WBEM ZenPack provides a WBEM data source type and a WBEMPlugin base modeler plugin, and is used to create customized ZenPacks. This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed.</td>
<td>P</td>
</tr>
<tr>
<td>WindowsMonitor (Microsoft Windows) on page 67</td>
<td>The ZenPacks.zenoss.WindowsMonitor ZenPack uses WMI to monitor the performance of Microsoft Windows servers. NOTE: This ZenPack is included in the Zenoss Core RPM package, and copied to the $ZENHOME/packs directory, but not installed. In addition, this ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.</td>
<td>M</td>
</tr>
<tr>
<td>(XenMonitor) Xen Virtual Hosts on page 75</td>
<td>The ZenPacks.zenoss.XenMonitor ZenPack monitors Xen para-virtualized domains.</td>
<td>M</td>
</tr>
<tr>
<td>(ZenJMX) Java Management Extensions on page 77</td>
<td>The ZenPacks.zenoss.ZenJMX ZenPack adds the zenjmx daemon, which communicates with remote Java Management Extensions (JMX) agents, to collect data from Java-based applications.</td>
<td>P</td>
</tr>
<tr>
<td>(ZenossVirtualHostMonitor)</td>
<td>The Zenpacks.zenoss.ZenossVirtualHostMonitor ZenPack is undocumented.</td>
<td>P</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.ActiveDirectory ZenPack uses WMI to monitor Microsoft Active Directory authentication metrics.

**Note** This ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.

This ZenPack creates a device class for Microsoft Active Directory with appropriate priorities. It also creates a Windows Service class and IP Service class for Active Directory-related services with monitoring enabled.

Use this ZenPack to monitor the following metrics:

- DS Client Binds/Sec
- DS Directory Reads/Sec, Searches/Sec and Writes/Sec
- DS Monitor List Size
- DS Name Cache Hit Rate
- DS Notify Queue Size
- DS Search Sub-operations/Sec
- DS Server Binds/Sec, Server Name Translations/Sec
- DS Threads In Use
- KDC AS Requests, TGS Requests
- Kerberos Authentications
- LDAP Active Threads
- LDAP Bind Time
- LDAP Client Sessions
- LDAP New / New SSL and Closed Connections/Sec
- LDAP Searches/Sec, Writes/Sec
- LDAP Successful Binds
- LDAP UDP Operations/Sec
- NTLM Authentications

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required Zen Packs</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
</tbody>
</table>
Enable Monitoring

All Active Directory services must have a device entry under the /Devices/Server/Windows/Active Directory device class. In addition, verify that your Zenoss Core Windows service account has access to the Active Directory service.

1. Navigate to the device or device class in the Zenoss Core interface.
   - If applying changes to a device class:
     1. Select the class in the devices hierarchy.
     2. Click Details.
   - If applying changes to a device:
     1. Click the device in the device list.
     2. Select Configuration Properties.

2. Verify the credentials for the service account to access the service.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zWinUser</td>
<td>Windows user with privileges to gather performance information.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Password for the above user.</td>
</tr>
</tbody>
</table>

3. Click Save to save your changes.

You will now be able to start collecting the Active Directory server metrics from this device.

4. Navigate to Graphs and you should see some placeholders for graphs. After approximately fifteen minutes you should see the graphs start to become populated with information.

Note: For more information about user credentials and troubleshooting WMI connections, see WindowsMonitor (Microsoft Windows) on page 67.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zenwinperf</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.ApacheMonitor ZenPack monitors Apache HTTP Server by collecting metrics through the `mod_status` module.

The following metrics are collected and graphed for Apache HTTP Server.

- Requests per Second
- Throughput (Bytes/sec and Bytes/request)
- CPU Utilization of the HTTP server and all worker processes or threads
- Slot Usage (Open, Waiting, Reading Request, Sending Reply, Keep-Alive DNS Lookup, and Logging)

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.ApacheMonitor</td>
</tr>
</tbody>
</table>

### Display the Status Page in Apache Version 1.3 or Higher

1. On the Apache server, locate the `httpd.conf` file. Generally, this file is located at `/etc/httpd/httpd.conf` or `/etc/httpd/conf/httpd.conf`; however, other locations are possible depending on your operating system and setup.

   If you cannot locate the configuration file, use your system’s search facilities to locate it. For Windows, use the **Search** button of the Windows Explorer tool. For Unix, try the following command:

   ```bash
   find / -name httpd.conf
   ```

2. Check to see that the following line is not commented out and is available in `httpd.conf` or `/etc/apache/modules.conf`:

   ```bash
   LoadModule status_module /usr/lib/apache/1.3/mod_status.so
   ```

**Note**: You may have to search in alternate locations to find the `mod_status.so` file. Also, the syntax may differ depending on your configuration.
3 Turn the ExtendedStatus option on in the httpd.conf file. This option is typically commented out. You can enable it by uncommenting it or ensuring that it is defined.

```plaintext
#ExtendedStatus on

becomes:

ExtendedStatus on
```

4 Enable the /server-status location in the httpd.conf file. Typically, this option exists but is commented out.

```plaintext
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .example.com
#</Location>

becomes:

```text
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from zenoss.example.com
</Location>
```

**Note**  Your Zenoss Core server or servers must be able to connect to your Apache server. Ensure that it is listed here or is part of the network specified in this chunk of configuration.

To specify multiple servers, separate the entries with spaces. If you specify an IP address range rather than a destination, be sure to add a network mask to the end of the IP address range.

The following example allows a server called externalzenoss.example.com, as well as all servers that start with 192.168.10, in their addresses:

```text
<Location /server-status>
SetHandler server-status
Order deny,allow
Deny from all
Allow from externalzenoss.example.com
</Location>
```

5 Save the httpd.conf file with these changes and verify that the configuration file is correct. This can be accomplished with following command.

```
apachectl -t
```

Correct any issues before restarting Apache.

6 Restart the Web server (httpd). This can be accomplished with following command.

```
apachectl restart
```
Display the Status Page in Apache Version 2.x

1 On the Apache server, find the `httpd.conf` file. This is usually `/etc/apache2/apache2.conf` or `/etc/apache2/conf/httpd.conf`; however, other locations are possible depending on your operating system and setup.

   If you are unsure about where your configuration file is located, use your system’s search facilities to locate this file. Under Windows, use the **Search** button of the Windows Explorer tool. Under Unix, try the following command:

   ```
   find / -name httpd.conf
   ```

2 Verify that the mod_status module is loaded.

   ```
   apache% apachec2ctl -M 2<&1 | grep status
   status_module (shared)
   ```

   The previous output indicates that the module is loaded and no further configuration is necessary. If there is no output, then copy the `mods-available/status.load` to the `mods-enabled` directory, and then run:

   ```
   apache% /etc/init.d/apache2 force-reload
   ```

3 Turn the ExtendedStatus option on in the `httpd.conf` file. This option is typically commented out. You can enable it by uncommenting it or ensuring that it is defined.

   ```
   #ExtendedStatus on
   ```

   becomes:

   ```
   ExtendedStatus on
   ```

4 Enable the `/server-status` location in the `httpd.conf` file. This is another option that typically already exists but is commented out.

   ```
   #<Location /server-status>
   #    SetHandler server-status
   #    Order deny,allow
   #    Deny from all
   #    Allow from .example.com
   #</Location>
   ```

   becomes:

   ```
   <Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from zenoss.example.com
   </Location>
   ```

   **Note** Your Zenoss Core server or servers must be able to connect to your Apache server so you must ensure that it is either listed here or is a part of the network specified in this chunk of configuration.

   To specify multiple servers, separate the entries with spaces. If you would like to specify an IP address range rather than a destination, be sure to add a network mask to the end of the IP address range. The following
example allows a server called externalzenoss.example.com as well as all servers that start with '192.168.10' in their addresses:

```html
<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
  Allow from externalzenoss.example.com 192.168.10.0/24
</Location>
```

5 Save the `httpd.conf` file with these changes and verify that the configuration file is correct. This can be accomplished with following command.

```
apache2ctl -t
```

Correct any issues before restarting Apache.

6 Restart the webserver (`httpd`). This can be accomplished with following command.

```
apache2ctl restart
```

### Verifying Your Apache Configuration

Once Apache has been configured, you should verify that it is working correctly. To verify your Apache server, point your Web browser to your Apache server at the appropriately modified URL:

```
http://your-apache-server/server-status?auto
```

This is an example of what you might see:

```
Total Accesses: 1
Total kBytes: 2
Uptime: 43
ReqPerSec: 0.232558
BytesPerSec: 47.6279
BytesPerReq: 2048
BusyWorkers: 1
IdleWorkers: 5
Scoreboard: _W________________________
```

If there is a configuration issue, you should see an error message telling you that the page is forbidden.

**Note** Your Zenoss Core server or servers must be able to connect to your Apache server by using HTTP to receive information. This means that the Zenoss Core server must be permitted not only by the Apache configuration settings, but also by any firewalls or proxies between the Zenoss Core server and the Apache server, including any firewall on the Apache server. If there are any proxies, they must be configured to allow the Zenoss Core HTTP traffic through. Consult your network administrator and security officer to verify the firewall configuration and your site’s policies.

Further note that the name or IP address that your server has behind a firewall may be different than the IP address (some form of Network Address Translation (NAT)) or name resolution (the way that the external server resolves names may be through an Internet-visible DNS system rather than an intranet-only DNS system).

### Configure Zenoss Core to Monitor the Web Server

Once the Apache server is configured to allow Zenoss Core to access the extended status, you can add Apache monitoring to the device within Zenoss Core by binding the Apache template to the device.
1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
   The device overview page appears.
3. In the left panel, expand Monitoring Templates, and then select Device.
4. Select Bind Templates from the Action menu.
   
   The Bind Templates dialog appears.
5. Add the Apache template to the list of templates, and then click **Save**.
   
   The Apache template is added. The system can now begin collecting the Apache server metrics from this device.

### Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.DellMonitor ZenPack provides customized modeling of devices running Dell OpenManage agents, and includes identification for proprietary Dell hardware.

The following information is collected through SNMP:

- Hardware Model
- Hardware Serial Number
- Operating System
- CPU Information (socket, speed, cache, voltage)
- PCI Card Information (manufacturer, model)

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.DellMonitor</td>
</tr>
<tr>
<td>On each remote device</td>
<td>The Dell OpenManage SNMP Agent is used to gather information about the device.</td>
</tr>
</tbody>
</table>

**Enable Enhanced Modeling**

To enable modeling:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

   The device overview page appears.
3. Select Modeler Plugins from the left panel.
4. Click **Add Fields** to reveal the list of available plugins.
5. Select the following plugins from the Available fields list, and then drag them to the Plugins list:
   - DellCPUMap
   - DellDeviceMap
- DellPCIMap
6. Remove the following plugins by clicking on the 'X' button located to the right of the plugin.
   - zenoss.snmp.CpuMap
   - zenoss.snmp.DeviceMap
7. Click **Save** to save the updates.
8. Remodel the device using these new plugins by selecting Model Device from the Action menu.

### Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>zenmodeler</td>
</tr>
<tr>
<td>Performance Collector</td>
<td>zenperfsnmp</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.DigMonitor ZenPack monitors the response time of DNS lookups.

To collect data, this ZenPack uses the `check_dig` Nagios plugin, which in turn uses the `dig` command.

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.DigMonitor</td>
</tr>
</tbody>
</table>

**Enable Monitoring**

To enable monitoring by the system:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

   The device overview page appears.
3. Expand Monitoring Templates in the left panel, and then select Device.
4. Select Bind Templates from the Action menu.

   The Bind Templates dialog appears.
5. Add the DigMonitor template to the list of selected templates, and then click OK.

   The DigMonitor template appears under Monitoring Templates.
6. Select the DigMonitor template in the left panel, and change options as needed.

**Table 2: DigMonitor Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Server</td>
<td>Name server against which the <code>dig</code> command should be run. The default is the device host name.</td>
</tr>
<tr>
<td>Port</td>
<td>Port on which the name server is listening. This is normally port 53.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Record Name</td>
<td>Name of the record you want to look up. The default is zenoss.com.</td>
</tr>
<tr>
<td>Record Type</td>
<td>DNS record type (for example, A, MX, CNAME).</td>
</tr>
</tbody>
</table>

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.DnsMonitor ZenPack monitors the response time of DNS requests. This ZenPack uses the `check_dns` Nagios plugin to collect data, which in turn uses the `nslookup` command.

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.DNSMonitor</td>
</tr>
</tbody>
</table>

### Enable Monitoring

To enable monitoring by the system:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
The device overview page appears.
3. Expand Monitoring Templates in the left panel, and then select Device.
4. Select Bind Templates from the Action menu.
   
The Bind Templates dialog appears.
5. Add the DNSMonitor template to the list of selected templates, and then click **OK**.
   
The DNSMonitor template appears under Monitoring Templates.
6. Select the DNSMonitor template in the left panel, and change options as needed.

### Table 3: DNSMonitor Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNS Server</td>
<td>Name server against which the <code>nslookup</code> command should be run. If empty (the default), the default DNS server or servers in <code>/etc/resolve.conf</code> are used.</td>
</tr>
<tr>
<td>Port</td>
<td>Port on which the name server is listening. This is normally port 53.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Host Name</td>
<td>Host name to resolve. The default is the device ID.</td>
</tr>
<tr>
<td>Expected IP Address</td>
<td>IP address to which the host name is expected to resolve.</td>
</tr>
</tbody>
</table>

## Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.EsxTop ZenPack uses the VMware resxtop command to gather performance information about VMware Infrastructure™ ESX™ servers.

**Note**  This ZenPack is deprecated; see *(vSphere) VMware vSphere.*

This ZenPack can be used alone, or with one of the other VMware ZenPacks. When used alone, a basic modeler creates virtual machines under the /Devices/Server/Virtual Hosts/EsxTop device class for any host device that is added and modeled. Otherwise, performance data can be collected for the ESX hosts modeled by the other ZenPacks.

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.ZenossVirtualHostMonitor v2.3.5</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.EsxTop</td>
</tr>
<tr>
<td>Required Software (on collectors)</td>
<td>OpenSSL development package v0.9.7</td>
</tr>
<tr>
<td></td>
<td>VMware vSphere CLI v4.1</td>
</tr>
</tbody>
</table>

### Installing Prerequisite Libraries

The VMware vSphere CLI is required for access to the resxtop command, which enables Zenoss Core to model and gather performance information about individual ESX servers.

Follow these steps to install the CLI and required software:

1. If you have not yet installed it, install the OpenSSL development package. For example, for an RPM-based system, enter:

   ```
   yum install openssl-devel
   ```

2. From your VMware account, download the VMware vSphere CLI.
Note For downloads and documentation, go to:

http://downloads.vmware.com/d/details/vcli41/ZHcqYmRoaCpiZHRAag==

3 Copy the package to each Zenoss Core collector.
4 For each collector:
   a Expand the package file.
   b Run the following command to install the package:
      ```bash
      ./vmware-install.pl
      ```
   c As the zenoss user, run the following command to verify successful installation:
      ```bash
      resxtop --server myESXServer --user userOnRemoteEsxServerAllowedToUseEsxTop -b -n 1 -a
      ```
      The resxtop command prompts for a password.
   d Enter the password for a user with permissions on the remote ESX server.
   e Create a symbolic link from the location that the resxtop command was installed into the $ZENHOME/libexec directory. This allows the check_esxtop command to automatically determine which binary to run. For example:
      ```bash
      cd $ZENHOME/libexec
      ln -s PathToResxtop
      ```
   f Test the check_esxtop command by showing the VMs on the remote server:
      ```bash
      $ZENHOME/ZenPacks/Ze*EsxTop*/Z*/z*/E*/libexec/check_esxtop --server=myEsxserver --user=userOnRemoteEsxServerAllowedToUseEsxTop --password=password --showvms
      ```

**Enabling the ZenPack**

Follow these steps to enable this ZenPack. From the Zenoss Core interface, add a host:

1 From Infrastructure > Devices, navigate to the /Devices/Server/Virtual Hosts/EsxTop device class.
2 From the Add Device menu, select Add a Single Device.
   The Add a Single Device dialog appears.
3 Enter a host name or IP address.
4 De-select the Model Device option.
5 Click Add.
6 Select the newly added device in the list.
   The device overview appears.
7 Click Details, and then select Configuration Properties in the left panel.
8 Enter login credentials for the zCommandUsername and zCommandPassword configuration properties, and then click Save.
9 If the device has an SNMP agent installed, update the ESX device configuration with the appropriate SNMP configuration information, and then add any desired modeler plugins.
10 From the Action menu, select Model device.

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>zenmodeler</td>
</tr>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.FtpMonitor ZenPack monitors the response times of File Transfer Protocol (FTP) server connection requests.

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.FtpMonitor</td>
</tr>
</tbody>
</table>

### Enable Monitoring

To enable monitoring of the device:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

   The device overview page appears.
3. Expand Monitoring Templates in the left panel, and then select Device.
4. Select Bind Templates from the Action menu.

   The Bind Templates dialog appears.
5. Select the FTPMonitor template and move it to the list of selected templates.
6. Click Save.

   The FTPMonitor template appears under Monitoring Templates.
7. Select the FTPMonitor template and change options as needed.

### Table 4: FTPMonitor Basic Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The port to connect to FTP server (default 21)</td>
</tr>
<tr>
<td>Send String</td>
<td>Command string to send to the server</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expect String</td>
<td>A string to expect in server response</td>
</tr>
<tr>
<td>Mismatch</td>
<td>If the expected string does not match the string returned from the remote server, create an event with one of these states: ok, warn, crit (default: warn)</td>
</tr>
<tr>
<td>Quit String</td>
<td>Command to send to the remote server to end the session</td>
</tr>
</tbody>
</table>

**Enable Secure Site Monitoring**

To enable secure site monitoring:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the devices list.

   The device overview page appears.
3. Expand Monitoring Templates in the left panel.
4. Select the FTPMonitor template and change options as needed.

**Table 5: FTPMonitor Secure Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The port to connect to FTP server (default 21).</td>
</tr>
<tr>
<td>Certificate</td>
<td>Minimum days for which a certificate is valid</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Use SSL for the connection</td>
</tr>
</tbody>
</table>

**Tuning for Site Responsiveness**

1. Select Infrastructure from the navigation bar.
2. Click the device name in the devices list.

   The device overview page appears.
3. Expand Monitoring Templates in the left panel.
4. Select the FTPMonitor template and change options as needed.

**Table 6: FTPMonitor Tunables Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Seconds before connection times out (default: 60)</td>
</tr>
<tr>
<td>Refuse</td>
<td>If a TCP/IP connection to the remote service is refused (ie no program is listening at that port) send an event with one of these severity states: ok, warn, crit (default: crit)</td>
</tr>
<tr>
<td>Max Bytes</td>
<td>Close the connection once more than this number of bytes are received.</td>
</tr>
<tr>
<td>Delay</td>
<td>Seconds to wait between sending string and polling for response</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Warning response time (seconds)</td>
<td>Response time to result in a warning status.</td>
</tr>
<tr>
<td>Critical response time (seconds)</td>
<td>Response time to result in critical status</td>
</tr>
</tbody>
</table>

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.HPMonitor ZenPack provides customized modeling of devices running HP Insight Management Agents, and includes identification for proprietary HP hardware.

The following information is collected through SNMP:

- Hardware Model
- Hardware Serial Number
- Operating System
- CPU Information (socket, speed, cache)

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.HPMonitor</td>
</tr>
<tr>
<td>On each remote device</td>
<td>The HP Insight SNMP Management Agent gathers information about the device.</td>
</tr>
</tbody>
</table>

### Enable Enhanced Modeling

To enable enhanced modeling:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
   The device overview page appears.
3. Select Modeler Plugins from the left panel.
4. Click Add Fields to reveal the list of available plugins.
5. Select the following available plugins and drag them to the plugins list:
   - HPCpuMap
   - HPDeviceMap
6. Remove the following plugins by clicking the 'X' button to the right of the plugin:
- zenoss.snmp.CPUMap
- zenoss.snmp.DeviceMap

7 Click **Save**.

8 Remodel the device using the new plugins. To do this, select Model Device from the Action menu.

### Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>zenmodeler</td>
</tr>
<tr>
<td>Performance Collector</td>
<td>zenperfsnmp</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.HttpMonitor ZenPack monitors the response times of HTTP server connection requests, and determines whether specific content exists on a Web page.

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.HttpMonitor</td>
</tr>
</tbody>
</table>

### Enable Monitoring

Follow these steps to enable monitoring:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   The device overview page appears.
3. Expand Monitoring Templates, and then select Device from the left panel.
4. Select Bind Templates from the Action menu.
   The Bind Templates dialog appears.
5. Add the HttpMonitor template to the list of selected templates, and then click **Submit**.

**Note** Prior to Zenoss 2.4, this template was not available. If your version is prior to Zenoss 2.4 you must create the template, data source and graphs manually. For more information, refer to Zenoss Service Dynamics Resource Manager Administration.

The HttpMonitor template is added to the list of monitoring templates. You can now begin collecting Web server metrics from the device.

### Check for a Specific URL or Specify Security Settings

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
The device overview page appears.
3. Expand Monitoring Templates, and then select Device from the left panel.
4. Create a local copy of the template.
5. Select the newly created local template copy.
6. Select the HttpMonitor data source, and then select View and Edit Details from the Action menu.

The Edit Data Source dialog appears.
7. Change data source options as needed, and then click Save.

**Table 7: HTTPMonitor Content Checking Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The port to connect to HTTP server (default 80).</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Use SSL for the connection</td>
</tr>
<tr>
<td>Url</td>
<td>Address of the web page.</td>
</tr>
<tr>
<td>Basic Auth User</td>
<td>If the website requires credentials, specify the username here.</td>
</tr>
<tr>
<td>Basic Auth Password</td>
<td>Password for the user.</td>
</tr>
<tr>
<td>Redirect Behavior</td>
<td>If the web site returns an HTTP redirect, should the probe follow the redirect or create an event? Possible event severities are OK, Warning, and Critical.</td>
</tr>
</tbody>
</table>

**Check for Specific Content on the Web Page**

This procedure allows Zenoss Core to create an event if content at the web page does not match the expected output.

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

The device overview page appears.
3. Expand Monitoring Templates, and then select Device from the left panel.
4. Create a local copy of the template.
5. Select the newly created local template copy.
6. Select the HttpMonitor data source, and then select View and Edit Details from the Action menu.

The Edit Data Source dialog appears.
7. Change data source options as needed, and then click Save.

**Table 8: HTTPMonitor Content Checking Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Expression</td>
<td>A Python regular expression to match text in the web page.</td>
</tr>
<tr>
<td>Case Sensitive</td>
<td>Is the regular expression case-sensitive or not?</td>
</tr>
</tbody>
</table>
Option | Description
---|---
Invert Expression | If you would like to test to see if the web page does **not** contain content matched by a regular expression, check this box.

**Tuning for Site Responsiveness**

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

   The device overview page appears.
3. Expand Monitoring Templates, and then select Device from the left panel.
4. Create a local copy of the template.
5. Select the newly created local template copy.
6. Select the HttpMonitor data source, and then select View and Edit Details from the Action menu.

   The Edit Data Source dialog appears.
7. Change data source options as needed, and then click **Save**.

**Table 9: HTTPMonitor Tunables Data Source Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout (seconds)</td>
<td>Seconds before connection times out (default: 60)</td>
</tr>
<tr>
<td>Cycle Time (seconds)</td>
<td>Number of seconds between collection cycles (default: 300 or five minutes)</td>
</tr>
</tbody>
</table>

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.IISMonitor ZenPack uses Windows Perfmon to collect performance metrics from Microsoft Internet Information Server (IIS).

**Note** This ZenPack is deprecated; see *Microsoft Windows* on page 40.

No agent need be installed on IIS servers to collect the following metrics.

- Connections Attempts
- Throughput (Bytes & Files)
- Requests (GET, HEAD, POST, CGI, ISAPI)
  - Standard: GET, HEAD, POST, CGI, ISAPI
  - WebDAV: PUT, COPY, MOVE, DELETE, OPTIONS, PROPFIND, PROPPATCH, MKCOL
  - Other: SEARCH, TRACE, LOCK, UNLOCK

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.IISMonitor</td>
</tr>
</tbody>
</table>

**Enable Monitoring**

All IIS servers must have a device entry in an organizer below the /Devices/Server/Windows/WMI device class. In addition, verify that your Zenoss Core Windows service account has access to the IIS service.

1. Bind the IIS template to the /Devices/Server/Windows/WMI class. To do this:
   a. Select the device class in the devices hierarchy.
   b. Click **Details**.
   c. Select Device_WMI under Monitoring Templates.
   d. Select Bind Templates from the Action menu.
The Bind Templates dialog appears.

1. Move IIS (/Server/Windows/WMI) from the Available area to the Selected area, and then click **Save**.

2. Navigate to the device or device class in the Zenoss Core interface.

- If applying changes to a device class:
  1. Select the class in the devices hierarchy.
  2. Click **Details**.

- If applying changes to a device:
  1. Click the device in the device list.
  2. Select Configuration Properties.

3. Verify the credentials for the service account to access the service.

### Table 10: IIS Configuration Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zWinUser</td>
<td>Windows user with privileges to gather performance information.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Password for the above user.</td>
</tr>
</tbody>
</table>

4. Click **Save** to save your changes.

You will now be able to start collecting the IIS server metrics from this device.

5. Navigate to Graphs and you should see some placeholders for graphs. After approximately fifteen minutes you should see the graphs begin to be populated with information.

---

**Note** For more information about user credentials and troubleshooting WMI connections, see *WindowsMonitor (Microsoft Windows)* on page 67.

---

**Daemons**

### Table 11: Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zenwinperf</td>
</tr>
</tbody>
</table>
(IRCDMonitor) IRCD Monitor

The ZenPacks.zenoss.IRCDMonitor ZenPack monitors the number of users connected to an Internet Relay Chat (IRC) server.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.IRCDMonitor</td>
</tr>
</tbody>
</table>

Enable Monitoring

To enable monitoring:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
   The device overview page appears.
3. Expand Monitoring Templates in the left panel, and then select Device.
4. Select Bind Templates from the Action menu.
   
   The Bind Templates dialog appears.
5. Move the IrcdMonitor template from the Available list and move it to the Selected list.
6. Click Save.
   
   The IrcdMonitor template is added.
7. Click the new template in the left panel and change options as needed.

Table 12: IRC Basic Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>Specifies the port to connect to the IRC server (default 6667).</td>
</tr>
<tr>
<td>warning_num</td>
<td>Creates a warning event when this number of users are seen.</td>
</tr>
</tbody>
</table>
### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>critical_num</td>
<td>Creates a critical event when this number of users are seen.</td>
</tr>
</tbody>
</table>

### Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
(JabberMonitor) Jabber Monitor

The ZenPacks.zenoss.JabberMonitor ZenPack monitors the response times of Jabber instant messaging servers.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.JabberMonitor</td>
</tr>
</tbody>
</table>

Enable Monitoring

To enable monitoring:

1. Select Infrastructure from the navigation bar.
2. Click the device in the device list.

   The device overview page appears.
3. Expand Monitoring Templates in the left panel, and then select Device.
4. Select Bind Templates from the Action menu.

   The Bind Templates dialog appears.
5. Move the Jabber template from the Available list to the Selected list, and then click Save.

   The Jabber template is added. The system can begin collecting Jabber server metrics from the device.
6. Select the newly added template and change options as needed.

Table 13: Jabber Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout (seconds)</td>
<td>Seconds before connection times out (default: 60)</td>
</tr>
<tr>
<td>Port</td>
<td>The port on which the Jabber server is listening. Typically this is port 5223.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Send String</td>
<td>string to send to the server : default</td>
</tr>
<tr>
<td></td>
<td><code>&lt;stream:stream to='${dev/id}'</code></td>
</tr>
<tr>
<td></td>
<td><code>xmlns:stream='http://etherx.jabber.org/streams'</code></td>
</tr>
<tr>
<td>Expect String</td>
<td>String to expect in server response.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;stream&gt;</code></td>
</tr>
</tbody>
</table>

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
(LDAPMonitor) LDAP Monitor


The response time unit of measurement is milliseconds.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.LDAPMonitor</td>
</tr>
</tbody>
</table>

Enable monitoring for a device

Note: The LDAPServer template must be bound to the device class or device you want to monitor.

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
   The device overview page appears.
3. Select Configuration Properties from the left panel.
4. Modify configuration property values as needed for your environment. Check with your LDAP administrator for more information.

Table 14: LDAPServer Configuration Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zLDAPBaseDN</td>
<td>The Base Distinguished Name for your LDAP server. Typically this is the organization's domain name (for example, dc=foobar, dc=com)</td>
</tr>
<tr>
<td>zLDAPBindDN</td>
<td>The Distinguished Name to use for binding to the LDAP server, if authentication is required</td>
</tr>
</tbody>
</table>
Table 15: LDAPServer Basic Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port</td>
<td>The port to connect to LDAP server (default 389)</td>
</tr>
<tr>
<td>Base Distinguished Name</td>
<td>Defaults to <code>${here/zLDAPBaseDN}</code></td>
</tr>
<tr>
<td>Bind Password</td>
<td>Defaults to <code>${here/zLDAPBindPassword}</code></td>
</tr>
<tr>
<td>Use SSL</td>
<td>Use SSL for the connection</td>
</tr>
</tbody>
</table>

**Note** If your LDAP servers require SSL or a custom port, select the ldap data source, and then change the Use SSL and Port fields as needed.

10 Validate your configuration by running `zencommand`. Verify that the `check_ldap` or `check_ldaps` command correctly connects to your LDAP server:

```
zencommand run -v10 -d yourdevicenamehere
```

Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.LinuxMonitor ZenPack demonstrates how to develop new plugins that collect performance data using Secure Shell.

This ZenPack demonstrates the Secure Shell (SSH) features, and enables modeling and monitoring several types of device components for devices placed in the `/Server/SSH/Linux` device class by running commands and parsing the output. Parsing of command output is performed on the Zenoss Core server or on a distributed collector. The account used to monitor the device does not require root access or special privileges.

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.4 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.LinuxMonitor</td>
</tr>
</tbody>
</table>

**Set Linux Server Monitoring Credentials**

All Linux servers must have a device entry in an organizer below the `/Devices/Server/SSH/Linux` device class.

**Note** The SSH monitoring feature will attempt to use key-based authentication before using a configuration properties password value.

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.

   The device overview page appears.

3. Select Configuration Properties from the left panel.
4. Verify the credentials for the service account.
Add a Linux Server

The following procedure assumes that credentials have been set.

1. Select Infrastructure from the navigation bar.
2. Select Add a Single Device from the Add Device list of options.

The Add a Single Device dialog appears.
3. Enter the following information in the dialog:

Table 17: Adding Linux Device Details

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name or IP</td>
<td>Linux host to model.</td>
</tr>
<tr>
<td>Device Class</td>
<td>/Server/SSH/Linux</td>
</tr>
<tr>
<td>Model Device</td>
<td>Select this option unless adding a device with a user name and password different than found in the device class. If you do not select this option, then you must add the credentials (see ) and then manually model the device.</td>
</tr>
</tbody>
</table>

4. Click Add.

Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>zenmodeler</td>
</tr>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.Microsoft.Windows ZenPack monitors Microsoft Windows systems and services through the Windows Remote Management (WinRM) and Windows Remote Shell (WinRS) interfaces.

Monitoring data is collected from the Windows Management Instrumentation (WMI) and Windows Reliability and Performance Monitor (Perfmon) services.

**Note** For Windows systems that support WinRM, this ZenPack replaces the following, deprecated ZenPacks:
- ZenPacks.zenoss.ActiveDirectory
- ZenPacks.zenoss.IISMonitor
- ZenPacks.zenoss.MSExchange
- ZenPacks.zenoss.MSMQMonitor
- ZenPacks.zenoss.MSSQLServer
- ZenPacks.zenoss.PySamba
- ZenPacks.zenoss.WindowsMonitor

However, installing this ZenPack does not automatically migrate Windows systems or disable monitoring through the deprecated ZenPacks. For more information about transitioning to this ZenPack, see *Transitioning to this ZenPack* on page 50.

### Discovery and modeling

The components, properties, and relationships that this ZenPack discovers and updates during modeling. Discovery and modeling are enabled by specifying server addresses, usernames, and passwords.

**Server (Device)**
- Attributes: Name, Contact, Description, Serial Number, Tag, Hardware Model, Total Memory, Total Swap, Operating System, Cluster

**Cluster (Device)**
- Attributes: Name, Contact, Description, Total Memory, Total Swap, Operating System, Member Servers

**Processors**
- Attributes: Name, Description, Model, Socket, Cores, Threads, Clock Speed, External Speed, Voltage, L1 Cache Size, L2 Cache Size and Speed, L3 Cache Size and Speed

**File System**
- Attributes: Mount Point, Status, Storage Device, Type, Block Size, Total Blocks, Total Bytes, Maximum Name Length
Interfaces
Attributes: Name, Description, MAC Address, MTU, Speed, Duplex, Type, Administrative Status, Operational Status, IP Addresses

Network Routes
Attributes: Destination, Next Hop, Interface, Protocol, Type

Process Sets
Attributes: Name, Recent Matches, Process Class

Software
Attributes: Name, Vendor, Installation Date

Services
Attributes: Name, Display Name, Start Mode, Account

Cluster Services
Attributes: Name, Core Group, Owner Node, State, Description, Priority
Relationships: Cluster Resources

Cluster Resources
Attributes: Name, Owner Node, Description, Owner Group, State
Relationships: Cluster Service

IIS Sites
Attributes: Name, Status, App Pool

SQL Server Instances
Attributes: Name
Relationships: SQL Server Databases

SQL Server Databases
Attributes: Name, Version, Owner, Last Backup, Last Log Backup, Accessible, Collation, Creation Date, Default File Group, Primary File Path
Relationships: SQL Server Instance

SQL Server Backups
Attributes: Name, Device Type, Physical Allocation, Status
Relationships: SQL Server Instance

SQL Server Jobs
Attributes: Name, Job ID, Description, Enabled, Date Created, Username
Relationships: SQL Server Instance

Performance Monitoring

Perfmon counter data are collected through the PowerShell Get-Counter cmdlet, within a remote shell (WinRS). To collect data from any other Perfmon counter, simply add the counter to the appropriate Zenoss Core monitoring template.

Note The following Processes metrics (Win32_PerfFormattedData_PerfProc_Process) are collected directly through Windows Management Instrumentation (WMI):
- PercentProcessorTime
- WorkingSet
- WorkingSetPrivate (not available on Windows 2003)
Device metrics

The following metrics are collected from the Devices Perfmon counter.

- \Memory\Available bytes
- \Memory\Committed Bytes
- \Memory\Pages Input/sec
- \Memory\Pages Output/sec
- \Paging File\% Usage
- \Processor\% Privileged Time
- \Processor\% Processor Time
- \Processor\% User Time
- \System\System Up Time

Active Directory metrics

The following metrics are collected from the Active Directory Perfmon counter.

- \NTDS\DS Client Binds/sec
- \NTDS\DS Directory Reads/sec
- \NTDS\DS Directory Searches/sec
- \NTDS\DS Directory Writes/sec
- \NTDS\DS Monitor List Size
- \NTDS\DS Name Cache hit rate
- \NTDS\DS Notify Queue Size
- \NTDS\DS Search sub-operations/sec
- \NTDS\DS Server Binds/sec
- \NTDS\DS Server Name Translations/sec
- \NTDS\DS Threads in Use
- \NTDS\KDC AS Requests
- \NTDS\KDC TGS Requests
- \NTDS\Kerberos Authentications
- \NTDS\LDAP Active Threads
- \NTDS\LDAP Bind Time
- \NTDS\LDAP Client Sessions
- \NTDS\LDAP Closed Connections/sec
- \NTDS\LDAP New Connections/sec
- \NTDS\LDAP New SSL Connections/sec
- \NTDS\LDAP Searches/sec
- \NTDS\LDAP Successful Binds/sec
- \NTDS\LDAP UDP operations/sec
- \NTDS\LDAP Writess/sec
- \NTDS\NTLM Authentications
- \NTDS\DS Client Binds/sec
- \NTDS\DS Directory Reads/sec
- \NTDS\DS Directory Searches/sec
- \NTDS\DS Directory Writes/sec
- \NTDS\DS Monitor List Size
- \NTDS\DS Name Cache hit rate
- \NTDS\DS Notify Queue Size
- \NTDS\DS Search sub-operations/sec
- \NTDS\DS Server Binds/sec
- \NTDS\DS Server Name Translations/sec
- \NTDS\DS Threads in Use
- \NTDS\LDAP Active Threads
- \NTDS\LDAP Bind Time
- \NTDS\LDAP Client Sessions
- \NTDS\LDAP Closed Connections/sec
- \NTDS\LDAP New Connections/sec
- \NTDS\LDAP New SSL Connections/sec
- \NTDS\LDAP Searches/sec
- \NTDS\LDAP Successful Binds/sec
- \NTDS\LDAP UDP operations/sec
- \NTDS\LDAP Writes/sec
- \DirectoryServices(NTDS)\DS Client Binds/sec
- \DirectoryServices(NTDS)\DS Directory Reads/sec
- \DirectoryServices(NTDS)\DS Directory Searches/sec
- \DirectoryServices(NTDS)\DS Directory Writes/sec
- \DirectoryServices(NTDS)\DS Monitor List Size
- \DirectoryServices(NTDS)\DS Name Cache hit rate
- \DirectoryServices(NTDS)\DS Notify Queue Size
- \DirectoryServices(NTDS)\DS Search sub-operations/sec
- \DirectoryServices(NTDS)\DS Server Binds/sec
- \DirectoryServices(NTDS)\DS Server Name Translations/sec
- \DirectoryServices(NTDS)\DS Threads in Use
- \DirectoryServices(NTDS)\LDAP Active Threads
- \DirectoryServices(NTDS)\LDAP Bind Time
- \DirectoryServices(NTDS)\LDAP Client Sessions
- \DirectoryServices(NTDS)\LDAP Closed Connections/sec
- \DirectoryServices(NTDS)\LDAP New Connections/sec
- \DirectoryServices(NTDS)\LDAP New SSL Connections/sec
- \DirectoryServices(NTDS)\LDAP Searches/sec
- \DirectoryServices(NTDS)\LDAP Successful Binds/sec
- \DirectoryServices(NTDS)\LDAP UDP operations/sec
- \DirectoryServices(NTDS)\LDAP Writes/sec

**Exchange metrics**

The following metrics are collected from the Exchange Perfmon counter.

- \MSExchangeIS Mailbox(_Total)\Folder opens/sec
- \MSExchangeIS Mailbox(_Total)\Local delivery rate
- \MSExchangeIS Mailbox(_Total)\Message Opens/sec
- \MSExchangeIS\RPC Averaged Latency
- \MSExchangeIS\RPC Operations/sec
- \MSExchangeIS\RPC Requests
- \SMTP Server(_Total)\Local Queue Length
- \SMTP Server(_Total)\Messages Delivered/sec
- \MSExchangeTransport Queues(_Total)\Active Mailbox Delivery Queue Length
- \MSExchangeTransport Queues(_Total)\Messages Completed Delivery Per Second
Zenoss Core Extended Monitoring

- \MSExchangeIS Mailbox(_Total)\Folder opens/sec
- \MSExchangeIS Mailbox(_Total)\Local delivery rate
- \MSExchangeIS Mailbox(_Total)\Message Opens/sec
- \MSExchangeIS\RPC Averaged Latency
- \MSExchangeIS\RPC Operations/sec
- \MSExchangeIS\RPC Requests

**IIS metrics**

The following metrics are collected from the **IIS** Perfmon counter.

- \Web Service(_Total)\Bytes Received/sec
- \Web Service(_Total)\Bytes Sent/sec
- \Web Service(_Total)\CGI Requests/sec
- \Web Service(_Total)\Connection Attempts/sec
- \Web Service(_Total)\Copy Requests/sec
- \Web Service(_Total)\Delete Requests/sec
- \Web Service(_Total)\Files Received/sec
- \Web Service(_Total)\Files Sent/sec
- \Web Service(_Total)\Get Requests/sec
- \Web Service(_Total)\Head Requests/sec
- \Web Service(_Total)\ISAPI Extension Requests/sec
- \Web Service(_Total)\Lock Requests/sec
- \Web Service(_Total)\Mkcol Requests/sec
- \Web Service(_Total)\Move Requests/sec
- \Web Service(_Total)\Options Requests/sec
- \Web Service(_Total)\Other Request Methods/sec
- \Web Service(_Total)\Post Requests/sec
- \Web Service(_Total)\Propfind Requests/sec
- \Web Service(_Total)\Proppatch Requests/sec
- \Web Service(_Total)\Put Requests/sec
- \Web Service(_Total)\Search Requests/sec
- \Web Service(_Total)\Trace Requests/sec
- \Web Service(_Total)\Unlock Requests/sec

**IIS Sites metrics**

The following metrics are collected from the **IIS Sites** Perfmon counter.

**Note** To use these metrics, install the IIS 6 Management Compatibility module on the servers to monitor.

- \Bytes Received/sec
- \Bytes Sent/sec
- \CGI Requests/sec
- \Connection Attempts/sec
- \Copy Requests/sec
- \Connection Attempts/sec
- \Delete Requests/sec
- \Files Received/sec
- \Files Sent/sec
- \Get Requests/sec
- Head Requests/sec
- ISAPI Extension Requests/sec
- Lock Requests/sec
- Mkcol Requests/sec
- Move Requests/sec
- Options Requests/sec
- Other Request Methods/sec
- Post Requests/sec
- Propfind Requests/sec
- Proppatch Requests/sec
- Put Requests/sec
- Search Requests/sec
- Trace Requests/sec
- Unlock Requests/sec

**SQLServer metrics**

The following metrics are collected from the SQLServer Perfmon counter.

- \SQLServer:Access Methods\Full Scans/sec
- \SQLServer:Buffer Manager\Buffer cache hit ratio
- \SQLServer:Buffer Manager\Free pages
- \SQLServer:Databases(_Total)\Data File(s) Size (KB)
- \SQLServer:General Statistics\User Connections
- \SQLServer:Latches\Latch Waits/sec
- \SQLServer:Locks(_Total)\Average Wait Time (ms)
- \SQLServer:Locks(_Total)\Lock Requests/sec
- \SQLServer:Locks(_Total)\Number of Deadlocks/sec
- \SQLServer:SQL Statistics\Batch Requests/sec

**File Systems metrics**

The following metrics are collected from the File Systems Perfmon counter.

- \Disk Read Bytes/sec
- \% Disk Read Time
- \Disk Write Bytes/sec
- \% Disk Write Time
- \Free Megabytes

**Interfaces metrics**

The following metrics are collected from the Interfaces Perfmon counter.

- \Bytes Received/sec
- \Bytes Sent/sec
- \Packets Received Errors
- \Packets Received/sec
- \Packets Outbound Errors
- \Packets Sent/sec
Event Management

Events are collected from Windows event logs through a WinRM subscription. Information encoded in Windows event classes is used to populate the following, standard Zenoss event fields.

- device
- component
- summary
- severity
- eventClassKey (for mapping specific event types)
- eventKey (for de-duplication and auto-clear fingerprinting)

Events collected in this manner are timestamped with the time from the Windows event log, not the collection time.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>RPM Packages</td>
<td>krb5-workstation</td>
</tr>
<tr>
<td>Windows</td>
<td>WinRM 2.0. (Windows Server 2003 with SP1 and all subsequent releases of Windows Server include WinRM 2.0.)</td>
</tr>
<tr>
<td>management</td>
<td></td>
</tr>
<tr>
<td>IIS management</td>
<td>To enable the IIS Sites metrics, install the IIS 6 Management Compatibility module on the Windows systems to monitor.</td>
</tr>
</tbody>
</table>

Installing Kerberos authentication

To use this ZenPack, Kerberos authentication must be installed on all Zenoss Core hosts (master, collector, and hub hosts).

1. Log in to a Zenoss Core host as root, or as a user with superuser privileges.
2. Determine whether the Kerberos authentication package is installed.
   
   rpm -qa | grep -i krb5-workstation

   - If the command returns a result, Kerberos authentication is installed. Discontinue this procedure.
   - If the command does not return a result, proceed to the next step.
3. Install Kerberos authentication.
   yum -y install krb5-workstation

Adding a Windows device

Follow these steps to add a Windows device through the Zenoss Core user interface.

1. Log in to Zenoss Core user interface as a user with ZenManager or Manager privileges.
2. Navigate to the INFRASTRUCTURE page.
3. In the left column, select the /Server/Microsoft/Windows class, and then click DETAILS.
4. Select Configuration Properties, and then provide values for the zWinRMUser and zWinRMPassword properties.
5. In the left column, click SEE ALL.
6. From the the Add menu, select Add a Single Device.
7. Complete the form with the information for the Windows device to add.
The value of the **Name or IP** field must be resolvable and accessible from the collector host specified in the **Collector** field.

8 Click **ADD**.

**Adding multiple Windows devices**

Follow these steps to add multiple Windows devices with the `zenbatchload` command.

1 Log in to the Zenoss Core master host as `zenoss`.
2 Create a text file with content similar to the following example.

```
/Devices/Server/Microsoft/Windows
FQDN-or-IP-address zWinRMUser="Administrator", zWinRMPassword="Password"
FQDN-or-IP-address zWinRMUser="Administrator", zWinRMPassword="Password"
FQDN-or-IP-address zWinRMUser="Administrator", zWinRMPassword="Password"
```

Replace `FQDN-or-IP-address` with the fully-qualified domain name or IP address of the Windows host to add, and replace `Password` with the Administrator's password for each host.

3 Load the devices into Zenoss Core. Replace `Filename` with the name of the file created in the preceding step. `zenbatchload Filename`

**Configuring systems for monitoring**

You may configure the Windows systems to monitor with this ZenPack collectively or individually.

- To configure systems collectively, use Windows Group Policy.

  **Note** This option may require configuration steps on each Windows system to monitor.

- To configure systems individually, use the procedure in this section.

  **Note** To enable the IIS Sites metrics, install the IIS 6 Management Compatibility module on the Windows systems to monitor.

**Authentication and transport options**

For authentication and transport, you may configure one of the options in the following list. This ZenPack supports all four options.

**Basic authentication, HTTP transport**

This option is the least secure and is not recommended. Usernames, passwords, and payloads are transmitted in clear text.

**Basic authentication, HTTPS transport**

This option encrypts usermames, passwords, and payloads at the transport layer, with SSL. The Windows systems to monitor must be configured to support HTTPS individually.

**Kerberos authentication, HTTP transport**

This option encrypts usernames and passwords (at the application layer) but payloads are not encrypted.

**Kerberos authentication, HTTPS transport**

This option is the most secure and is recommended. Usernames, passwords, and payloads are all encrypted, first at the application layer, and again at the transport layer. The Windows systems to monitor must be configured to support HTTPS individually.
NTLMv2 is not supported, and cannot be configured to work with this ZenPack.

Configuring systems with Windows Group Policy

Windows Remote Management (WinRM)

The Computer Configuration\Policies\Administrative Templates\Windows Components\Windows Remote Management category includes the WinRM Client and WinRM Service sub-categories. For WinRM Client, no policy changes are required. The default policies provide the needed support.

The following table shows the WinRM Service policies to set for the desired authentication and transport option.

Table 18: WinRM Service policy options

<table>
<thead>
<tr>
<th>Authentication/Transport option</th>
<th>Basic/HTTP</th>
<th>Basic/HTTPS</th>
<th>Kerberos/HTTP</th>
<th>Kerberos/HTTPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow remote server management through WinRM</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Enabled</td>
</tr>
<tr>
<td>Allow unencrypted Traffic</td>
<td>Enabled</td>
<td>Disabled</td>
<td>Enabled</td>
<td>Disabled</td>
</tr>
<tr>
<td>Allow Basic authentication</td>
<td>Enabled</td>
<td>Enabled</td>
<td>Disabled</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

For more information about configuring HTTPS, see Configuring HTTPS on page 49.

Windows Remote Shell (WinRS)

Configure the following policies in the Computer Configuration\Policies\Administrative Templates\Windows Components\Windows Remote Shell category.

- Policy: Allow Remote Shell Access
  Choose the default setting, which allows remote shell connections.

- Policy: Specify maximum number of processes per Shell
  Set the value to 4294967295.
  
  **Note** This is the maximum value. The default (5) is inadequate, because Zenoss Core opens concurrent requests for each WQL query and set of Perfmon counters. If the maximum value is inappropriate for your environment, 50 should be adequate.

- Policy: Specify maximum number of remote shells per user
  Set the value to 2147483647.
  
  **Note** This is the maximum value. The default (5) is inadequate, because Zenoss Core opens concurrent requests for each WQL query and set of Perfmon counters. If the maximum value is inappropriate for your environment, 50 should be adequate.

- Policy: Specify Shell Timeout
  Set the value to 7200000.
Configuring HTTPS

Currently, HTTPS must be configured on each Windows system to monitor. Zenoss is testing several options for automating the task, but none are available for production use.

To successfully encrypt the payload between Zenoss Core and Windows clients, you must install a Server Authentication certificate on the client system. Once the correct certificate is installed, change the client’s zWinScheme property to HTTPS and the zWinRMPort property to 5986.

If the preceding steps are unsuccessful, verify that the appropriate Service Principal Name (SPN) record exists for Kerberos authentication. Log in to any Active Directory system and enter the following command. Replace Hostname with the hostname portion of the client system’s fully-qualified domain name (FQDN).

```bash
setspn -l Hostname
```

If the preceding command returns a hostname record that does not start with HTTPS/, use the following command to create the record. Replace FQDN with the fully-qualified domain name of the client system.

```bash
setspn -s HTTPS/FQDN Hostname
```

Configuring systems individually

You may configure the Windows systems to monitor individually, without Windows Group Policy.

**Note** If you choose the default WinRM configuration, supply Kerberos authentication settings in zProperties. The Kerberos authentication process requires a Key Distribution Center (KDC), which acts as both an authentication server and a ticket granting server. Microsoft Active Directory is a KDC, so the zWinKDC value must be set to the IP address of the Active Directory server, and Zenoss Core collectors must be able to send TCP/IP packets to it. The value of zWinRMUserName must include a fully qualified domain name (for example, jsmith@zenoss.com) and zWinRMPassword must include the password of the user account.

Follow these steps to enable this ZenPack to monitor a Windows system.

1. Log on to the Windows system to enable, and open the Windows Firewall with Advanced Security utility, or its equivalent.
2. Open port 5985 (for HTTP) or 5986 (for HTTPS) for WinRM, and then close the utility.
3. Start a command window as Administrator.
4. Configure the system to accept WS-Management requests from other systems.
   ```bash
   winrm quickconfig
   ```
5. Configure the maximum number of concurrent operations per user.
   ```bash
   winrm s winrm/config/service @{MaxConcurrentOperationsPerUser="4294967295"}
   ```
   **Note** This is the maximum value. The default (5) is inadequate, because Zenoss Core opens concurrent requests for each WQL query and set of Perfmon counters. If the maximum value is inappropriate for your environment, 50 should be adequate.
6. Configure the maximum number of shells per user.
   ```bash
   winrm s winrm/config/winrs @{MaxShellsPerUser="2147483647"}
   ```
   **Note** This is the maximum value. The default (5) is inadequate, because Zenoss Core opens concurrent requests for each WQL query and set of Perfmon counters. If the maximum value is inappropriate for your environment, 50 should be adequate.
7 Configure the idle timeout.
   `winrm s winrm/config/winrs @{IdleTimeout="7200000"}`

8 Optional: Configure HTTP. The default is HTTPS.
   `winrm s winrm/config/service @{AllowUnencrypted="true"}

   For more information about configuring HTTPS, see Configuring HTTPS on page 49.

9 Optional: Configure Basic Authentication. The default is Kerberos.
   `winrm s winrm/config/service/auth @{Basic="true"}

### Transitioning to this ZenPack

This ZenPack provides a superset of the functionality provided in earlier Windows monitoring ZenPacks, with better performance and reliability.

**Note**  The new and old Windows monitoring ZenPacks co-exist gracefully—no automatic migration is attempted, and all history is preserved.

Zenoss recommends transitioning to this ZenPack as soon as possible in all but the following situations:

- You must continue to monitor Windows systems that do not support WinRM 2.0.

  At some point, Zenoss will discontinue support for the deprecated ZenPacks, so moving older Windows systems to a version that supports WinRM 2.0 is encouraged.

- You must maintain an unbroken history of monitoring, using exactly the same metrics, gathered in exactly the same way.

  Testing or regulatory requirements may prevent an immediate migration. However, no monitoring history is discarded when this ZenPack is enabled. You may maintain the RRD files created with the deprecated ZenPacks alongside the new ones, and use them to examine the complete monitoring history.

To determine whether the deprecated Windows monitoring ZenPacks are installed, log in to the browser interface, and navigate to ADVANCED > ZenPacks. Look for the following ZenPacks:

- ZenPacks.zenoss.ActiveDirectory
- ZenPacks.zenoss.IISMonitor
- ZenPacks.zenoss.MSExchange
- ZenPacks.zenoss.MSMQMonitor
- ZenPacks.zenoss.MSSQLServer
- ZenPacks.zenoss.PySamba
- ZenPacks.zenoss.WindowsMonitor

### Moving Windows systems to this ZenPack

1 Log in to Zenoss Core user interface as a user with ZenManager or Manager privileges.

2 Navigate to the INFRASTRUCTURE page.

3 In the left column, select the /Server/Windows/WMI device class.

4 In the device list, select a Windows system to remove from the class.

5 From the bottom of the left column, click the Action button, and then select Delete Device....

6 In the Delete Device dialog, click SUBMIT.

Add the deleted Windows device to the /Server/Microsoft/Windows class. For more information, see Adding a Windows device on page 46.
Using this ZenPack with the deprecated ZenPacks

You may use this ZenPack with the deprecated Windows monitoring ZenPacks

- by adding some Windows systems to the /Server/Microsoft/Windows device class and leaving other systems in the /Server/Windows/WMI device class
- by adding a subset of this ZenPack, such as the Windows Shell data source, to devices in the /Server/Windows/WMI device class

To add a subset of this ZenPack to a device, follow these steps.

1. In the /Server/Windows/WMI device class, select the device to modify.
2. Verify that the following properties are configured.
   - **zWinUser**
     - Use the DOMAIN\Username format required for DCOM/RPC collection.
   - **zWinPassword**
     - The password for the zWinUser account.
   - **zWinRMUser**
     - Use the username@example.com format required for WinRM collection.
   - **zWinRMPassword**
     - The password for the zWinRMUser account.
3. Create a monitoring template containing a Windows Shell datasource, and then bind it to the device.

Installed items

This ZenPack installs the following items.

- **Device classes**
  - /Server/Microsoft
  - /Server/Microsoft/Cluster
  - /Server/Microsoft/Windows

- **Configuration properties**
  - zWinRMUser
  - zWinRMPassword
  - zWinRMPort
  - zDBInstances
  - zDBInstancesPassword
  - zWinKDC
  - zWinKeyTabFilePath
  - zWinScheme
  - zWinPerfmonInterval

- **Modeler plugins**
  - zenoss.winrm.CPUs
  - zenoss.winrm.FileSystems
  - zenoss.winrm.IIS
  - zenoss.winrm.Interfaces
  - zenoss.winrm.OperatingSystem
  - zenoss.winrm.Processes
zenoss.winrm.Routes
zenoss.winrm.Services
zenoss.winrm.Software
zenoss.winrm.WinCluster
zenoss.winrm.WinMSSQL

Datasource types
Windows EventLog
Windows IIS Site
Windows Perfmon
Windows Process
Windows Service
Windows Shell

Monitoring templates
Device (in /Server/Microsoft)
FileSystem (in /Server/Microsoft)
ethernetCsmacd (in /Server/Microsoft)
OSProcess (in /Server/Microsoft)
OSProcess-2003 (in /Server/Microsoft)
WinService (in /Server/Microsoft)
Active Directory (in /Server/Microsoft)
Active Directory 2008 (in /Server/Microsoft)
Active Directory 2008R2 (in /Server/Microsoft)
IIS (in /Server/Microsoft)
IISADMIN (in /Server/Microsoft)
IISSites (in /Server/Microsoft)
MSExchangeIS (in /Server/Microsoft)
MSExchangeIS 2007 (in /Server/Microsoft)
MSSQLServer (in /Server/Microsoft)
WinDatabase (in /Server/Microsoft)
Cluster (in /Server/Microsoft)
ClusterService (in /Server/Microsoft/Cluster)
ClusterResource (in /Server/Microsoft/Cluster)
The ZenPacks.zenoss.MSExchange ZenPack uses WMI to monitor Microsoft Exchange and related services.

**Note**  This ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.

The ZenPack enables users to view graphs based on MS Exchange Performance Counters and to monitor processes related to MS Exchange.

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.MSExchange</td>
</tr>
</tbody>
</table>

### Enable Monitoring

All MS Exchange services must have a device entry under the /Devices/Server/Windows/MSExchange device class. In addition, verify that your Zenoss Core Windows service account has access to the MS Exchange service.

1. Navigate to the device or device class in the Zenoss Core interface.
   - If applying changes to a device class:
     1. Select the class in the devices hierarchy.
     2. Click **Details**.
   - If applying changes to a device:
     1. Click the device in the device list.
     2. Select Configuration Properties.
2. Verify the credentials for the service account to access the service.
### Table 19: MS Exchange Configuration Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zWinUser</td>
<td>Windows user with privileges to gather performance information.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Password for the above user.</td>
</tr>
</tbody>
</table>

3. Click **Save** to save your changes.

You will now be able to start collecting the MS Exchange server metrics from this device.

4. Navigate to Graphs and you should see some placeholders for graphs. After approximately fifteen minutes you should see the graphs start to become populated with information.

**Note** For more information about user credentials and troubleshooting WMI connections, see *WindowsMonitor (Microsoft Windows)* on page 67.

### Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zenwinperf</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.MSMQMonitor ZenPack uses WMI to automatically discover Microsoft Message Queueing (MSMQ) queues, and monitor the number of messages queued in each.

**Note**  This ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.

The following description of Microsoft Message Queuing (MSMQ) can be found on Microsoft's MSMQ product page.

“Microsoft Message Queuing (MSMQ) technology enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline. MSMQ provides guaranteed message delivery, efficient routing, security, and priority-based messaging. It can be used to implement solutions for both asynchronous and synchronous messaging scenarios.”

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.MSMQMonitor</td>
</tr>
</tbody>
</table>

**Configuration**

To monitor MSMQ queues, set up proper credentials so that Zenoss Core can remotely monitor the target Windows servers. For more information, refer to WindowsMonitor (Microsoft Windows) on page 67.

This ZenPack supports two approaches to enable MSMQ queue monitoring, as detailed in the next sections.

**Automatically Monitor Queues on All Servers**

The easiest way to configure Zenoss Core to monitor your queues is to enable queue discovery for the entire Server/Windows device class. Within 12 hours Zenoss Core will have automatically discovered all of the queues available to be monitored and begun monitoring how many messages are in each queue and creating threshold events if they exceed 10,000 messages.

Perform the following steps to enable queue discovery for all Windows servers.
1. Navigate to the **/Server/Windows** device class.
2. Click Details.
3. Select Modeler Plugins from the left panel.
4. Click Add Fields.
5. Drag `zenoss.wmi.MSMQQueueMap` from the available fields to the list of plugins.
6. Click Save.
7. Wait about 12 hours for all Windows servers to be remodeled.

**Monitor Queues on Specific Servers**

If you do not want Zenoss Core automatically monitoring queues on all of your Windows servers and would rather point it to specific servers you can do so by performing the following steps on each server you're interested in.

1. Navigate to the device.
2. Select Modeler Plugins from the left panel.
3. Click Add Fields.
4. Drag `zenoss.wmi.MSMQQueueMap` from the available fields to the list of plugins.
5. Click Save.
6. Select Model Device from the Action menu.

**Fine-Tuning Queue Monitoring**

By default Zenoss Core will automatically monitor all queues on a server that is running the MSMQ services. Each queue will also have a default 10,000 maximum threshold applied to it. This means that an event will be created when the number of messages in a single queue exceeds 10,000.

**Note** By default queues with names beginning with **tcp** will not be discovered. You can change this behavior with the `zMSMQIgnoreQueues` property. This property is a regular expression and any queues that match it will not be discovered.

You can change the maximum messages threshold on a per-queue basis by changing the **Queues Messages Threshold** property. Leaving this value blank will have the result of no threshold being applied.

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td><code>zenmodeler</code></td>
</tr>
<tr>
<td>Performance Collector</td>
<td><code>zenwinperf</code></td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.MSSQLServer ZenPack uses WMI to monitor Microsoft SQL Server and its related services.

Note  This ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.

The ZenPack enables users to view graphs based on Microsoft SQL Server Performance Counters and to monitor processes related to SQL Server.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.MSSQLServer</td>
</tr>
</tbody>
</table>

Enable Monitoring

All MS SQL Server services must have a device entry under the /Devices/Server/Windows/MSSQLServer device class. In addition, verify that your Zenoss Core Windows service account has access to the MS SQL Server service.

1. Navigate to the device or device class in the Zenoss Core interface.
   - If applying changes to a device class:
     1. Select the class in the devices hierarchy.
     2. Click Details.
   - If applying changes to a device:
     1. Click the device in the device list.
     2. Select Configuration Properties.

2. Verify the credentials for the service account to access the service.
Table 20: MS SQL Server Configuration Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zWinUser</td>
<td>Windows user with privileges to gather performance information.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Password for the above user.</td>
</tr>
</tbody>
</table>

3 Click Save to save your changes.

You will now be able to start collecting the MS SQL Server server metrics from this device.

4 Navigate to Graphs to see placeholders for graphs. After approximately fifteen minutes, the graphs start to become populated with information.

Note For more information about user credentials and troubleshooting WMI connections, see WindowsMonitor (Microsoft Windows) on page 67.

Collecting Information from Non-Default Microsoft SQL Server Instances

The default Microsoft SQL Server instance is SQLServer. The monitoring template delivered with this ZenPack uses this default instance to gather performance metrics. If you use a non-default SQL Server instance, then Zenoss Core does not automatically find and gather information about it.

To enable Zenoss Core to monitor a non-default instance, you must override the monitoring template:

1 From Infrastructure > Devices, click the device on which you want to override the template.
2 Under Monitoring Templates, select the MSSQLServer template.
3 From the Action menu, select Override Template Here.

The Override Templates dialog appears.
4 Select the MSSQLServer template in the list, and then click Submit.

The template redisplay in the left panel, now identified as "Locally Defined."

5 For each of the data sources in the Data Sources area, perform these steps:

a Double-click the data source to edit it.
b In the Perf Counter field, change the text "\SQLServer:" to "\MyInstance:" (where MyInstance is the name of the Microsoft SQL Server database instance name.
c Click Save.
6 Remodel the device.

Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zenwinperf</td>
</tr>
</tbody>
</table>
(MySqlMonitor) MySQL Database Monitor


**Note** Release 4.2.5 of Zenoss Core includes version 3.x of this ZenPack, which collects data through a customized modeler plugin. If you have upgraded Zenoss Core from an earlier release, see *Migrating from version 2.x to 3.x* on page 62.

This ZenPack provides the following features.

- Discovery of MySQL entities
- Monitoring of MySQL Server and Database components
- Event management and monitoring for certain MySQL states
- Integration with Zenoss Service Dynamics Service Impact (Service Impact)

**Discovery**

The following entities are discovered through the zMySQL.ConnectionString property you provide. The attributes, tags and collections are updated during remodeling, which defaults to every 12 hours.

**Servers**
- Attributes: Percentage of full table scans, Slave status, Master status, Number of databases
- Tags: Name
- Collections: Databases

**Databases**
- Attributes: Number of tables, Default character set, Default collation
- Tags: Name

All discovered values are valid as of the most recent modeling time.

**Performance Monitoring**

Server and database metrics are collected every 5 minutes by default. The Average statistic is collected, and the graphed value is per second for anything that resembles a rate.
Server metrics

<table>
<thead>
<tr>
<th>Component</th>
<th>Component state</th>
<th>Event severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aborted clients</td>
<td>Aborted connects</td>
<td>Bytes received</td>
</tr>
<tr>
<td>Com alter db</td>
<td>Com alter table</td>
<td>Com call procedure</td>
</tr>
<tr>
<td>Com commit</td>
<td>Com create db</td>
<td>Com create table</td>
</tr>
<tr>
<td>Com delete multi</td>
<td>Com delete</td>
<td>Com drop db</td>
</tr>
<tr>
<td>Com drop user</td>
<td>Com execute sql</td>
<td>Com flush</td>
</tr>
<tr>
<td>Com insert</td>
<td>Com purge</td>
<td>Com repair</td>
</tr>
<tr>
<td>Com rollback</td>
<td>Com select</td>
<td>Com update multi</td>
</tr>
<tr>
<td>Connections</td>
<td>Data size</td>
<td>Handler commit</td>
</tr>
<tr>
<td>Handler read first</td>
<td>Handler read key</td>
<td>Handler read last</td>
</tr>
<tr>
<td>Handler read prev</td>
<td>Handler read rnd next</td>
<td>Handler read rnd</td>
</tr>
<tr>
<td>Handler savepoint</td>
<td>Handler update</td>
<td>Handler write</td>
</tr>
<tr>
<td>Key read requests</td>
<td>Key reads</td>
<td>Key writes</td>
</tr>
<tr>
<td>Open files</td>
<td>Open streams</td>
<td>Open tables</td>
</tr>
<tr>
<td>Select full range join</td>
<td>Select range check</td>
<td>Select range</td>
</tr>
<tr>
<td>Size</td>
<td>Threads connected</td>
<td>Uptime</td>
</tr>
</tbody>
</table>
Integration with Service Impact

This ZenPack includes custom state providers for services running on MySQL hosts. Custom state providers enable specialized options for defining state triggers in Zenoss Service Dynamics Service Impact (Service Impact).

The following relationships are automatically included in any Service Impact services that contain one or more of the explicitly mentioned components.

- Server failure affects related device
- Database failure affects related server

The preceding relationships follow the default policy, in which a node inherits the worst state of the ancestor nodes to which it is related. For example, a server failure implies that all related databases are also failed.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.MySqlMonitor</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.PythonCollector, version 1.1 or higher</td>
</tr>
<tr>
<td>Link for remote Zenoss DataStore</td>
<td>To monitor Zenoss DataStore with this ZenPack, see Enabling Zenoss DataStore monitoring on page 62.</td>
</tr>
</tbody>
</table>

Adding a MySQL Server instance

To monitor a MySQL Server instance through a user account other than root, grant SELECT privileges to the account, before adding the instance to Zenoss Core. For more information, refer to MySQL GRANT syntax.

Follow these steps to monitor a MySQL Server instance with Zenoss Core.

1. Log in to Zenoss Core browser interface as a user with ZenManager or Manager privileges.
2. Navigate to the INFRASTRUCTURE > Devices page.
3. In the left column, select the device class of the MySQL Server host.
4. In the device list, click the host's entry, to display its details page.
5. Configure the zMySQLConnectionString property.
   a. In the left column, select Configuration Properties.
   b. In the properties list, double-click the zMySQLConnectionString property.
c  In the **Edit Config Property** dialog, enter a valid MySQL Server user account and password, and the port on which the MySQL Server instance listens for connection requests.

**Note**  Zenoss Core version 4.1.x does not support the customized dialog for MySQL Server credentials. Create a JSON list instead.

d  Click **SUBMIT**.

6  Add the MySQL plugin.
   a  In the left column, select **Modeler Plugins**.
   b  In the list of available plugins, select **MySQL Collector**.
   c  Click the right arrow button.
   d  At the bottom of the page, click **Save**.

### Enabling Zenoss DataStore monitoring

Follow these steps to enable this ZenPack to monitor a Zenoss DataStore instance that is not located on the Zenoss Core master host.

1  Log in to the remote Zenoss DataStore host as **zenoss**.
2  Create a symbolic link to the Zenoss DataStore socket file.

```
 ln -s /var/lib/zends/zends.sock /tmp/mysql.sock
```

### Migrating from version 2.x to 3.x

Perform this procedure if one of the following statements is true.

- You have upgraded Zenoss Core to release 4.2.5, or to a more recent release.
- You have manually updated this ZenPack from version 2.x to version 3.x, or to a more recent version.

**Note**  Version 3.x uses a customized modeler plugin to collect data, and data collected through version 2.x of this ZenPack is incompatible. The version 2.x data is preserved. New RRD files are created for the new data.

1  Log in to Zenoss Core browser interface as a user with ZenManager or Manager privileges.
2  Navigate to the **INFRASTRUCTURE > Devices** page.
3  From the device list, select the MySQL Server host.
4  From the action menu at the bottom of the left column, select **Bind Templates**.
5  In the **Bind Templates** dialog, select **MySQL (/Server)** from the **Selected** column.
6  Click the left arrow button to move the template to the **Available** column.
7  Click **SAVE**.
8  From the action menu at the bottom of the left column, select **Model Device**.

### Installed items

This ZenPack adds the following items to Zenoss Core.

**Modeler Plugins**
- MySQLCollector

**Monitoring Templates**
- MySQLServer(in /Server)
- MySQLDatabase (in /Server)

**Component Types**
- MySQLServer (on related device)
MySQLDatabase (on MySQLServer)
(NNTPMonitor) NNTP Monitor

The ZenPacks.zenoss.NNTPMonitor ZenPack monitors the response time of Network News Transfer Protocol (NNTP) servers.

The response time unit of measurement is milliseconds.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.NNTPMonitor</td>
</tr>
</tbody>
</table>

Enable Monitoring

To enable monitoring for a device:

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
   
   The device overview page appears.
3. Expand Monitoring Templates, and then select Device from the left panel.
4. Select Bind Templates from the Action menu.
   
   The Bind Templates dialog appears.
5. Add the NNTPMonitor template to the list of selected templates, and then click **Submit**.
   
   The NNTPMonitor template is added to the list of monitoring templates.
6. Select the template and change options as needed.
7. Validate your configuration by running `zencommand` and observing that the `check_nntp` or `check_nntps` command correctly connects to your NNTP server:

```
zencommand run -v10 -d yourdevicenamehere
```
## Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
The ZenPacks.zenoss.NtpMonitor ZenPack monitors the difference between the system time a server is using and the time a Network Time Protocol (NTP) server is reporting.

**Prerequisites**

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.NtpMonitor</td>
</tr>
</tbody>
</table>

**Enable Monitoring**

The NTPMonitor template must be bound to the device class or device you want to monitor.

1. Select Infrastructure from the navigation bar.
2. Click the device name in the device list.
3. The device overview page appears.
4. Expand Monitoring Templates, and then select Device from the left panel.
5. Select Bind Templates from the Action menu.
   - The Bind Templates dialog appears.
6. Add the NTPMonitor template to the list of selected templates, and then click **Submit**.
   - The NTPMonitor template is added to the list of monitoring templates. You can now start collecting the NTP server metrics from this device.

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
WindowsMonitor (Microsoft Windows)

The ZenPacks.zenoss.WindowsMonitor ZenPack uses WMI to monitor the performance of Microsoft Windows servers.

**Note**  This ZenPack is deprecated; see (Microsoft.Windows) Microsoft Windows on page 40.

The WinPerf data source uses a Windows performance counter (rather than an SNMP OID) to specify the value to collect. For more information on Windows Management Instrumentation (WMI), please see this Microsoft Technet Article.

### Table 21: Windows Monitoring Daemons

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zenwin</td>
<td>Watches Windows services and reports on status.</td>
</tr>
<tr>
<td>zeneventlog</td>
<td>Watches the Windows event log and generates events.</td>
</tr>
<tr>
<td>zenwinperf</td>
<td>Collects performance data.</td>
</tr>
</tbody>
</table>

### Prerequisites

Supported OS versions are:

- Windows XP
- Windows 2000
- Windows 2003
- Windows Vista
- Windows 2008

### Table 22: Windows Performance Monitoring Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.WindowsMonitor,</td>
</tr>
</tbody>
</table>
Defining Windows Credentials

A connection to a Windows device cannot be established without a valid set of credentials. The `zWinUser` and `zWinPassword` configuration properties can be set for each device or for an entire device class.

**Note** The user needs to be a member of the local administrators or of the domain administrators group unless the steps in are followed.

To set these configuration properties:

1. Navigate to the device or device class in the Zenoss Core interface.
   - If applying changes to a device class:
     1. Select the class in the devices hierarchy.
     2. Click Details.
   - If applying changes to a device:
     1. Click the device in the device list.
     2. Select Configuration Properties.

2. Edit appropriate configuration properties for the device or devices.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zWinUser</td>
<td>Windows user with privileges to gather performance information. Like all Windows credentials, the domain should be specified in the <code>zWinUser</code> entry. Use <code>\username</code> for an account that is not in the domain but only on the local computer.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Password for the above user.</td>
</tr>
</tbody>
</table>

3. Click **Save** to save your changes.

Add Devices in Zenoss Core

The WindowsMonitor ZenPack includes a /Device/Server/Windows/WMI class that has several device templates bound. SNMP data collection is not used in this class.

To move a device to the /Device/Server/Windows/WMI class:

1. Select the device row in the devices list.
2. Drag the device to the class in the devices hierarchy.

Monitor Other Performance Counters

To create your own *WinPerf* data sources, follow these steps:
1. Navigate to a new or existing monitoring template, and select New DataSource from the Data Sources table menu.
2. Enter a name for the data source, select WinPerf as the type and then click OK.
3. Enter a Windows performance counter in the Perf Counter field. See Windows Perfmon counters for more details.
4. Click Save. Notice that a data point is created with the same name as the performance counter you selected.
5. Optionally, test the counter by entering a device ID in the Test Device field and clicking the Test button.

Testing Connections from Windows

This procedure verifies that the username/password combination is correct, and that there is no firewall blocking the connection.

1. Run the wbemtest command.
2. Click the Connect… button.
3. In the Namespace field, enter:

\HOST\root\cimv2

4. Enter login information in the User and Password fields.
5. Click the Query field.
6. Enter the following query to return a dialog with a list of services on the device.

```plaintext
select * from win32_service
```

Testing Connections from Zenoss Core

This procedure verifies that the username/password combination is correct, and that there is no firewall blocking the connection. Since this is done from the Zenoss Core server, this test is a better approximation of how successful Zenoss Core will be in connecting to the Windows device.

As the zenoss user on the Zenoss Core server:

```plaintext
wmic -U 'user' //device 'select * from Win32_computerSystem'
```

The wmic command will then prompt you for the password.

Modify Registry Settings for Firewalls in Secure Environments

Note: This procedure is applicable only for environments with firewalls.

The Distributed Component Object Model (DCOM) dynamically allocates one port to each process. You need to decide how many ports you want to allocate to DCOM processes, which is equivalent to the number of simultaneous DCOM processes through the firewall. You must open all of the UDP and TCP ports corresponding to the port numbers you choose. You also need to open TCP/UDP 135, which is used for RPC End Point Mapping, among other things. In addition, you must edit the registry to tell DCOM which ports you reserved. You do this with the HKEY_LOCAL_MACHINES\Software\Microsoft\Rpc\Internet registry key, which you will probably have to create.

To allow remote registry access for the performance data to be read, see Controlling remote Performance Monitor access to Windows NT servers.

The following table shows the registry settings to restrict DCOMs port range to 10 ports.
Table 24: Firewall and Registry Settings for DCOM

<table>
<thead>
<tr>
<th>Registry Key</th>
<th>Type</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ports</td>
<td>REG_MULTI_SZ</td>
<td>Range of port. Can be multiple lines such as: 3001-3010 135</td>
</tr>
<tr>
<td>PortsInternetAvailable</td>
<td>REG_SZ</td>
<td>y</td>
</tr>
<tr>
<td>UseInternetPorts</td>
<td>REG_SZ</td>
<td>y</td>
</tr>
</tbody>
</table>

These registry settings must be established in addition to all firewall settings.

Configuring a Standalone Windows Device for a Non-Administrative Account

Monitoring Windows devices normally requires an account with administrator-level privileges. For the Zenoss Core user who wants to use a non-administrative account, several additional configuration steps must be performed on each Windows device, or by using a Group Policy.

Zenoss Core uses the Windows Management Instrumentation (WMI) feature to collect modeling information. The remote Windows registry API also is used to collect low-level performance monitor ("PerfMon") statistics. Both of these Windows sub-systems use the Microsoft Remote Procedure Call (MS-RPC) interface to connect to the Windows device and gather the appropriate information. MS-RPC handles the authentication on a per-packet or per-session basis, but ultimately the access granted is determined by the sub-systems involved with serving the remote procedure calls.

1. If the Windows firewall is in use, modify it to allow Remote Administration access. This will open the MS-RPC port and others as needed. Enter the following command at the command prompt:

```
netsh firewall set service RemoteAdmin enable
```

2. On Windows XP, Simple File Sharing must be disabled for machines that are not located within a Domain. When this feature is enabled it causes all incoming MS-RPC connections to use the built-in Guest account, rather than the account credentials specified in the incoming call. This option may be found by going to Control Panel, opening the Folder Options applet and then choosing the View tab. In the Advanced Settings list, locate the Use simple file sharing (Recommended) option, and then disable it.
3 Create a local account on the Windows device for monitoring. We assume in the remainder of these steps that this account was named *zenossmon* but any valid account name can be used. Place the account only in the Users group and not in the Power Users or Administrators groups. Optionally, create a new user group for monitoring and use that group instead of the account in the remaining steps.

4 Give the *zenossmon* account DCOM access by running the *dcomcnfg* utility.

**Figure 2: Component Services COM Security Settings**

- In the **Component Services** dialog box, expand **Component Services**, expand **Computers**, and then right-click **My Computer** and click **Properties**.
- In the **My Computer Properties** dialog box, click the **COM Security** tab.
- Under **Access Permissions**, click **Edit Limits**. In the **Access Permission** dialog box, add the *zenossmon* account to the list and ensure that the **Remote Access** checkbox is enabled, then click **OK** to close the dialog.
d Under **Launch and Activation Permissions**, click **Edit Limits**. In the **Access Permission** dialog box, add the **zenossmon** account to the list and ensure that the **Remote Launch** and **Remote Activation** checkboxes are enabled, then click **OK** to close the dialog.

e Click **OK** on the **My Computer Properties** dialog to save all changes.

5 Give the **zenossmon** account permissions to read the WMI namespace by using **WMI Control**.

**Figure 3: WMI Control Properties**

| Open the **Start** menu and right-click on **My Computer**. Select **Manage** from the menu. |
| In the **Computer Management** dialog, expand the **Services and Applications** item and then right-click on **WMI Control**. |
| In the **WMI Control Properties** dialog, click the **Security** tab. |
| Expand the **Root** namespace, select the **CIMV2** namespace folder and then click **Security**. |
| In the **Security for ROOT\CIMV2** dialog, add the **zenossmon** user to the list and ensure the **Enable Account** and **Remote Enable** checkboxes are enabled, then click **OK** to close the dialog. |
| In the **WMI Control Properties** dialog click **OK** to close the dialog and save all changes. |

6 At this point in the process remote access to WMI should be enabled and functioning. Test it by running the following command from the Zenoss Core server:

```
wmic -U ".\zenossmon" /myhostname 'SELECT Name FROM Win32_ComputerSystem'
```

If all is well this command should return the remote system name as the response. If there is any error, carefully recheck the above steps to ensure all access has been properly granted.

7 To gather Windows performance data from PerfMon permissions on the **winreg** registry key must be granted to our monitoring user by using **regedit**.
Figure 4: regedit and the winreg Key

a  Run regedit.

b  Browse to the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control \SecurePipeServers\winreg key.

c  Right-click on the winreg key and choose Permissions.

d  Add the monitoring user to the permissions list and grant only Read permissions.

8 Give the zenossmon account access to read the Windows Event Log.

Once the appropriate changes are made, test that Event Log access works with your zenossmon user. Run the following from your Zenoss Core system:

```
wmic -U '.\zenossmon' //myhostname 'SELECT Message FROM Win32_NTLogEvent WHERE LogFile="Application"'
```

9 If you are using SP1 or newer with Windows Server 2003, then you must allow non-administrative users to access the service control manager to monitor services.

At a command prompt, run the following:

```
sc sdset SCMANAGER D: (A;;CCLCRPC;;AU) (A;;CCLCRPWPRC;;;SY) (A;;KA;;;BA) S: (AU;FA;KA;;;WD) (AU;OIIOFA;GA;;;WD)
```

Note  The above command should be one line.
At this point you should be able to query Windows service status remotely by using the non-administrative account. Test this by running the following command from your Zenoss Core system:

```
wmic -U '.\zenossmon' //myhostname 'SELECT Name FROM Win32_Service'
```

## Tuning Collector Daemon Performance

WindowsMonitor creates several configuration properties that control its behavior. Values for the configuration properties are initially set on the /Devices device class. As with any property, these values can be overridden in other device classes and on individual devices themselves.

<table>
<thead>
<tr>
<th>Table 25: zenwinperf Daemon Configuration Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Property</strong></td>
</tr>
<tr>
<td>zWinPerfCycleSeconds</td>
</tr>
</tbody>
</table>

### Multiple Workers

ZenWinPerf supports multiple workers. This feature allows you to support data collection from more Windows devices without defining additional collectors to host additional ZenWinPerf daemons. The multiple workers feature is enabled by a configuration option:

```
--workers
```


### Enabling the NTLMv2 Authentication Protocol

To enable the NTLMv2 authentication protocol for all Windows devices of a zenwin, zenwinperf, or zenevent log collector, update collector configuration files:

Alternatively, from the command line add:

```
--ntlmv2auth
```

```
# Enable NTLMv2 authentication for Windows
# Devices, default: False
#ntlmv2auth False
```
The ZenPacks.zenoss.XenMonitor ZenPack monitors Xen para-virtualized domains.

With this ZenPack, the zenmodeler can discover guests running on Xen hosts, and Zenoss Core includes screens and templates for collecting and displaying resources allocated to guests.

Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.XenMonitor</td>
</tr>
<tr>
<td></td>
<td>ZenPacks.zenoss.ZenossVirtualHostMonitor</td>
</tr>
</tbody>
</table>

Model Hosts and Guest

For each Xen server, follow this procedure:

1. Optionally, place an SSH key to your Xen server to allow the zenoss user from the Zenoss Core server to log in as root without requiring further credentials.
2. Create the Xen server in the /Servers/Virtual Hosts/Xen device class.

   **Note** If you have this server modeled already, remove the server and recreate it under the Xen device class. Do not move it.

3. Select the Guest menu and ensure that the guest hosts were found during the modeling process.

Daemons

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeler</td>
<td>zenmodeler</td>
</tr>
<tr>
<td>Performance Collector</td>
<td>zencommand</td>
</tr>
</tbody>
</table>
Monitoring with sudo

To configure `sudo` in order to run the `xm` on the Virtual Machine Host, you will need to modify a few things.

- Modify the `CommandPath` property to be blank, otherwise this path will be pre-pended to the `sudo` command.
- Modify the `CommandUsername` and `CommandPassword` configuration properties to be a non-root user with `sudo` access to the `xm` command.
- Modify the `Xen.py` modeler to add the `sudo` command. The modeler can be found under the `$ZENHOME/ZenPacks/ZenPacks.zenoss.ZenossVirtualHostMonitor` directory, under the `modeler/plugins` directory.
- Modify the performance templates.

1. Navigate to the `/Devices/Server/Virtual Machine Host/Xen` device class
2. From the device class click on the **Templates** tab
3. Click on the **VirtualMachine** performance template
4. In the **Data Sources** table, click on the **resources** Data Source
5. Edit the command template to add the `sudo` command to the beginning of the `xm` command
The ZenPacks.zenoss.ZenJMX ZenPack adds the zenjmx daemon, which communicates with remote Java Management Extensions (JMX) agents, to collect data from Java-based applications.

This ZenPack defines a data source named JMX that allows you to query any single or complex-value attribute, or invoke an MBean operation. It also comes with a built-in template named Java that contains MBean information for a few beans built into the JVM.

Note This ZenPack also includes a built-in template named ZenJMX. This template should be used only on devices running Java applications that make information available through JMX. To monitor other Java applications, use the included Java template.

When the zenjmx daemon is started it communicates with its zenhub daemon and retrieves a list of devices that possess JMX data sources. It also spawns a Java process. The zenjmx daemon asynchronously issues queries for each of those devices to the Java process via XML-RPC. The Java process then collects the data from the Java application to be monitored, and returns the results to the zenjmx daemon. Any collection or configuration errors are sent as events to Zenoss Core and appear in the event console.

Also, the zenjmx daemon sends heartbeat data to its zenhub daemon after each collection attempt, to let Zenoss Core know it is still alive and well.

**JMX Background**

The JMX technology is used throughout the Java Virtual Machine to provide performance and management information to clients. Using a combination of JConsole (Oracle’s JMX client that is shipped with the JDK) and JMX, a system operator can examine the number of threads that are active in the JVM or change the log level. There are numerous other performance metrics that can be gleaned from the JVM, as well as several management interfaces that can be invoked that change the behavior of the JVM.

In Java 5, Oracle introduced the Remote API for Java Management Extensions. This enhancement defines an RMI wrapper around a JMX agent and allows for independent client development. The zenjmx daemon accesses remote JMX agents via the Remote API for Java Management Extensions. It currently does not support local connections (provided via the temporary directory) to JMX Agents. JMX also specifies the manner in which various protocols can be used to connect to clients, and send and receive information. The original, most commonly used protocol is RMI. The zenjmx daemon supports RMI and JMXMP connections.
ZenJMX Capabilities

The zenjmx daemon is a full-featured JMX client that works "out of the box" with JMX agents that have their remote APIs enabled. It supports authenticated and unauthenticated connections, and it can retrieve single-value attributes, complex-value attributes, and the results of invoking an operation. Operations with parameters are also supported so long as the parameters are primitive types (Strings, booleans, numbers), as well as the object version of primitives (such as java.lang.Integer and java.lang.Float). Multi-value responses from operations (Maps and Lists) are supported, as are primitive responses from operations.

The JMX data source installed by this ZenPack allows you to define the connection, authentication, and retrieval information you want to use to retrieve performance information. The IP address is extracted from the parent device, but the port number of the JMX Agent is configurable in each data source. This allows you to operate multiple JMX Agents on a single device and retrieve performance information for each agent separately. This is commonly used on production servers that run multiple applications.

Authentication information is also associated with each JMX data source. This offers the most flexibility for site administrators because they can run some JMX agents in an open, unauthenticated fashion and others in a hardened and authenticated fashion. SSL-wrapped connections are supported by the underlying JMX Remote subsystem built into the JDK, but were not tested in the Zenoss labs. As a result, your success with SSL encrypted access to JMX Agents may vary.

The data source allows you to define the type of performance information you want to achieve: single-value attribute, complex-value attribute, or operation invocation. To specify the type of retrieval, you must specify an attribute name (and one or more data points) or provide operation information.

Any numerical value returned by a JMX agent can be retrieved by Zenoss Core and graphed and checked against thresholds. Non-numerical values (Strings and complex types) cannot be retrieved and stored by Zenoss Core.

When setting up data points, make sure you understand the semantics of the attribute name and choose the correct Zenoss Core data point type. Many JMX Agent implementations use inconsistent nomenclature when describing attributes. In some cases the term "Count" refers to an ever-increasing number (a "Counter" data point type). In other cases the term "Count" refers to a snapshot number (a "Gauge" data point type).

Allowable Parameter Types

The following primitive data types are allowed in JMX calls:

- java.lang.Integer
- java.lang.Long
- java.lang.Double
- java.lang.Float
- java.lang.String
- java.lang.Boolean
- int
- long
- double
- float
- boolean

Single Value Attribute Calls

This is the most basic usage scenario. If you are interested in retrieving a single value from an MBean in a JMX Agent, and the attribute returns simple numeric data, you fall into the "single value attribute" category. To define a
single-value attribute call simply provide the fully qualified name of your MBean and then provide the name of the attribute in the **Attribute Name** field of the data source. Lastly, you must define a data point.

Some examples of this include the commonly referenced JDK Threading information:

- **MBean Name:** java.lang:type=Threading
- **Attribute Name:** ThreadCount
- **Data Points:**
  - ThreadCount (type: gauge)

Java uses lots of file descriptors during normal operation. The number of open file descriptors the JVM is working with can be measured using the following information:

- **MBean Name:** java.lang:type=OperatingSystem
- **Attribute Name:** OpenFileDescriptorCount
- **Data Points:**
  - OpenFileDescriptorCount (type: gauge)

There are several other single-value attributes that can be retrieved from the JDK. We recommend using **JConsole** to interactively navigate through the MBean hierarchy to determine which MBeans contain useful information to you. See for additional information on how to inspect the MBeans deployed in an JMX Agent.

### Complex-Value Attribute Calls

If your MBean attribute defines multiple sub-attributes (via CompositeData or Tabular) that you are interested in capturing, then you fall into the category of a "complex-value attribute" call. The JDK contains a few complex-value attributes you might be interested in capturing, including garbage collection statistics that were captured during the copy and mark-sweep compact collection cycles.

To extract data from a complex-value attribute, you must define one or more data points in the data source. The names of the data points are used as keys into the complex-value data structure returned from the MBean attribute. For JMX CompositeData attributes, the data point names are used as a key to map the results. For JMX TabularData, the data point names are used as indexes into the structure to map the result.

The JDK also provides heap memory information via a complex-value attribute. The amount of committed, used, and maximum heap memory can be viewed by setting up a complex-value attribute in Zenoss Core with the following information:

- **MBean Name:** java.lang:type=Memory
- **Attribute Name:** HeapMemoryUsage
- **Data Points:**
  - committed (type: gauge)
  - used (type: gauge)
  - max (type: gauge)

### Example Method Calls

Some management values need to be computed. These situations frequently arise when custom MBeans are deployed alongside an enterprise application. An MBean named "Accounting" might be deployed within an enterprise application that defines operations intended for operators or support staff. These operations might include methods such as "getBankBalance()" or "countTotalDeposits()".
The `zenjmx` daemon can invoke operations, but there are some subtleties in how it sends parameters to a JMX Agent, and interprets the response.

**No parameters, single return value**

In the most basic usage scenario no arguments are passed to the operation and a single value is returned. This usage scenario is very similar to a single-value attribute call, except we're invoking an operation to retrieve the value rather than accessing an attribute. The configuration for this hypothetical usage scenario follows:

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBankBalance()
- Data Points:
  - balance (type: gauge)

**No parameters, multiple values returned in List format**

In this scenario no parameters are passed to an operation, but multiple response values are provided in a List. The values returned are expressed in a `List<Object>`, but they are coerced (but not casted) to doubles prior to being stored in Zenoss Core. This means that returning a numeric value as "1234" will work, but "1,234" will not work. The litmus test is to evaluate if `Double.valueOf(object.toString())` will successfully evaluate.

The `zenjmx` daemon can be configured to read multiple values from an operation's results by defining multiple data points. You must define a data point for each value returned from the operation, and if there is a mismatch between the number of data points you define and the size of the `List<Object>` returned an exception will be generated. The configuration for the `zenjmx` daemon follows:

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalanceSummary()
- Data Points:
  - dailyBalance (type: gauge)
  - annualBalance (type: gauge)

**No parameters, multiple values returned in Map format**

In this scenario no parameters are passed to an operation, but multiple response values are provided in a `Map<String, Object>`. The keyset of the Map contains the names of data points that can be defined, and the values are the values of said data points. When a `Map<String, Object>` is returned you need not capture all of the returned values as data points, and you can instead pick the exact values you are interested in. To choose the values to capture you simply define data points with the same names as Strings in the keyset.

The following configuration demonstrates how to extract specific data points from an operation that returns a `Map<String, Object>`. The key item to note in this configuration is that "dailyBalance" and "annualBalance" must be present as keys in the returned `Map<String, Object>` and their values must be coercible via the `Double.valueOf(object.toString())` idiom.

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalances()
- Data Points:
  - dailyBalance (type: gauge)
  - annualBalance (type: gauge)
Single parameter in polymorphic operation

MBeans are implemented as Java classes and Java permits parameterized polymorphic behavior. This means that multiple methods can be defined with the same name so long as their parameter signatures differ. You can safely define "getBalance(String)" and "getBalance()" and the two exist as separate methods.

In order to properly resolve methods with the same name the caller must provide a Class[] that lists the types of parameters that exist in the method's signature. This resolves the candidate methods to an individual method which can then be invoked by passing an Object[].

The zenjmx daemon allows you to resolve methods of the same name and asks you to provide the fully qualified class names of each parameter in comma delimited format when you set up the data source. Note that primitive types (String, Boolean, Integer, Float) are supported but complex types are not supported, and that you must include the class' package name when providing the information (java.lang.String).

The Object[] of parameter values must line up with Class[] of parameter types, and if there is a mismatch in the number of types and values that are provided an exception will be generated.

The marshaling of values from String to Boolean, Integer, and Float types is provided via the .valueOf() static method on each of those types. That is, if you define an attribute of type java.lang.Integer you must provide a String that can be successfully passed to java.lang.Integer.fromValue(). If you fail to do so an exception is generated.

This example illustrates how to pass a single parameter to a polymorphic operation:

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalances()
- Parameter Types: java.lang.Integer
- Parameter Values: 1234
- Data Points:
  - balance (type: gauge)

Here is another example where we've changed the type of the parameter passed to the method to be a String. Semantically it represents a different type of Account in our example:

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalances()
- Parameter Types: java.lang.String
- Parameter Values: sbb552349999
- Data Points:
  - balance (type: gauge)

Multiple parameters in polymorphic operations

The preceding example describes how polymorphic behavior in Java functions and how method resolution can be provided by identifying the Class[] that represents the parameters passed to a method. The situation where multiple parameters are passed to a polymorphic operation is no different than the situation where a single parameter is passed to a polymorphic operation, except that the length of the Class[] and Object[] is greater than one.

When multiple parameters are required to invoke an operation you must provide the fully qualified class names of each parameter's type in comma delimited format, as well as the object values for each type (also in comma delimited format).

The following example demonstrates a configuration that passes two parameters to an MBean operation. The second parameter passed is a default value to return if no account can be located matching the first parameter.
- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalances()
- Parameter Types: java.lang.String, java.lang.Integer
- Parameter Values: sbb552349999, 0
- Data Points:
  - balance (type: gauge)

There are additional combinations that are possible with polymorphic methods and the values they return, and those combinations are left as an exercise for the reader to explore. The logic for extracting results from multi-value operation invocations follows the same rules as the logic for extracting results from a multi-value attribute read. For additional information on the rules of that logic see the section above on multi-value attributes.

### Special Service URLs

By default, URLs are assembled as:

```
service:jmx:rmi:///jndi/rmi://hostName:portNum/jmxrmi
```

This host name and port points to a registry. After a JMX agent connects to the registry, the registry tells the agent which host and port to use for remote calls.

In some situations, you may want to explicitly provide the registry host and port, as well as the host and port for the remote calls. Use the long form, as in:

```
service:jmx:rmi://127.0.0.1:8999/jndi/rmi://127.0.0.1:8999/jmxrmi
```

### Prerequisites

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Zenoss Core 4.x, Zenoss 2.2 or higher</td>
</tr>
<tr>
<td>Required ZenPacks</td>
<td>ZenPacks.zenoss.ZenJMX</td>
</tr>
<tr>
<td>Other</td>
<td>Oracle JRE Version 5.0 or higher is required.</td>
</tr>
</tbody>
</table>

### Oracle Java Runtime Environment (JRE)

This ZenPack requires Oracle JRE Version 5.0 or higher. Make sure that after you install the JRE you update your PATH such that the `java` executable works. You can test this using the command:

```
$ which java
/usr/java/default/bin/java
```

If the above returns a fully qualified path, then you have successfully installed Java.

If Java is not installed, the `which` will return a message similar to the following:

```
$ which java
/usr/bin/which: no java in (/usr/local/bin:/bin:/usr/bin:/opt/zenoss/bin)
```
To determine which version of Java is installed, run the following command:

```
$ java -version
java version "1.5.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_16-b06-284)
Java HotSpot(TM) Client VM (build 1.5.0_16-133, mixed mode, sharing)
```

**Note** Oracle Java is required. Other Java implementations do not work.

### Example to Monitor a JMX Value

#### Enabling Remote JMX Access

Each application server has a slightly different process for enabling remote JMX Access. You should consult with your application server for specific instructions. This section includes instructions for a few commonly used configurations.

JMX agents can be configured in two ways: remote access and local-only. When configured for remote access a JMX client communicates with the JMX agent via a socket and uses a remote protocol such as Remote Method Invocation (RMI) or JMXMP to access the MBeans. When configured for local-only access the JMX agent periodically dumps serialized MBeans to a temporary directory on the machine. JConsole can be used to access JMX agents in local-only mode as well as in remote mode. The zenjmx daemon can be used only with remote servers via RMI or JMXMP and cannot work with local-only serialized MBeans. This is not a significant limitation because the zenjmx daemon can establish RMI connections to localhost in the same manner that it creates connections to remote hosts.

The JAVA_OPTS environment variable can be used to enable remote access to JVM MBeans. Set it as follows:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=12345
JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote.authenticate=false"
JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote.ssl=false"
export JAVA_OPTS
```

When starting an application pass the JAVA_OPTS variable as an argument to the JVM as follows:

```
java ${JAVA_OPTS} -classpath /path/to/application.jar com.yourcompany.Main
```

You can then use JConsole to connect to localhost:12345. Authentication can be configured by modifying the java.security file as well as java.policy. There are lots of examples available on the Internet that can provide guidance in how to achieve authenticated remote access to JVM MBeans.

### Configure Zenoss Core with a Custom Data Source

Custom JMX data sources allow system administrators to monitor any attribute or operation result accessible via a JMX call. This ZenPack creates a JMX data source and allows you to provide object information, as well as authentication settings, and attribute/operation information. Determining which object and attribute names, as well as which operations to invoke, is the key to customizing this feature.

To configure the system with a custom data source:

1. Select Infrastructure from the navigation bar.
2 Click the device in the device list.

The device overview page appears.

3 Expand Monitoring Templates in the left panel, and then select Device.

4 Select Add Local Template from the Action menu.

The Add Local Template dialog appears.

5 Enter a name for the template (such as JVM Values), and then click Submit.

The template is added.

6 Select the newly created template.

7 Click Add in the Data Sources area.

The Add Data Source dialog appears.

8 Enter a name for the data source (Heap Memory), select JMX as the type, and then click Submit.

The data source is added.

9 Double-click the data source to edit it. Change options as needed, and then click Save.

Table 26: Memory Head Example ZenJMX Data Source Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>RMI or JMXMP. Consult your Java application documentation to determine which JMX Connector protocols it supports.</td>
</tr>
<tr>
<td>JMX Management Port</td>
<td>This is not necessarily the same as the listen port for your server.</td>
</tr>
<tr>
<td>Object Name</td>
<td>The Object Name is also referred to as the MBean name. Enter java.lang:type=Memory</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Enter HeapMemoryUsage</td>
</tr>
</tbody>
</table>

10 Add data points named committed, max, and used:

a Select Add Data Point from the Action menu.

The Add Data Point dialog appears.

b Enter the name of the data point (committed, max, or used) and then click Submit.

11 After adding all data points, add graphs that reference them.

Review to learn how to determine the object name, attribute name, and data points that might be interesting in your application.

Monitor Values in TabularData and CompositeData Objects

The Attribute Path input value on the ZenJMX data source allows you to monitor values nested in the TabularData and CompositeData complex open data objects. Using this value you can specify a path to traverse and index into these complex data structures.

If the result of traversing and extracting a value out of the nested open data is a single numeric value then it is automatically mapped to the datapoint in the data source. However, if the value from the open data is another open data object then the data point names from the datasource are used as indexes or keys to map values out of the open data.
The input value is a dot-separated string that represents a path through the object. Non-bracketed values are keys into CompositeData. Bracketed values are indexes into TabularData.

For TabularData indexes with more than one value, use a comma-separated list with no spaces (for example, [key1,key2]).

To specify a column name (needed only when the table has more than two columns) use curly brackets after the table index.

**Example**

To get the used Tenured Generation memory after the last garbage collection from the Garbage Collector MBean, set the Attribute Name on the datasource to lastGcInfo. Set the Attribute Path to:

```
memoryUsageAfterGc.[Tenured Gen].[value].used
```

The key `memoryUsageAfterGc` is evaluated against the CompositeData returned from the `lastGcInfo` attribute. The evaluation results in a TabularData object. Then, the `[Tenured Gen]` index is evaluated against the TableData, which returns a row in the table.

Since a row in the table can contain multiple columns, the key `value` (in curly brackets) is used to pick a column in the row. Lastly, the key `used` is evaluated against the CompositeData in the column to return the memory value.

In this example, since the index being used for the tabular data is not a multi-value index and so the column name is optional. The Attribute Path can be written as:

```
memoryUsageAfterGc.[Tenured Gen].used
```

**Using JConsole to Query a JMX Agent**

JConsole is a tool built into the JDK that allows system administrators to query a JMX Agent and examine the MBeans deployed within the server. JConsole also allows administrators to view JVM summary information, including the amount of time the JVM has been running, how many threads are active, how much memory is currently used by the heap, how many classes are currently loaded, and how much physical memory exists on the machine.

JConsole also provides a graph that shows memory, thread, and class usage over time. The scale of the graph can be adjusted so that a system administrator can examine a specific period of time, or can zoom out to view a longer range picture of usage. Unfortunately, JConsole can only produce graphs that show usage while JConsole was running. Administrators cannot look back in time to a point where the JVM was running but JConsole was not monitoring the JVM.
The **MBeans** tab along the top of JConsole provides an interactive method for examining MBean values. After clicking on the **MBeans** tab a panel will be displayed with a tree on the left hand side. The tree contains a hierarchical list of all MBeans deployed in the JVM.

The standard JVM MBeans are all in the java.lang and java.util.logging packages. Application server specific MBeans do not follow any standard naming pattern. Some vendors choose to use package names for their MBean names while other vendors choose package-like names (but not fully qualified packages).

To get started expand the java.lang node in the Tree. This will expose several MBeans as well as additional folders. Click on the Memory MBean and observe how the right hand side of the panel is populated with information about the Memory MBean.
MBeans can contain attributes and operations. MBeans can also fire notifications to observers, but that's beyond the scope of this document. The attributes tab lists all of the attributes in the first column and their values (or a clickable attribute type) in the second column. In the case of Memory the HeapMemoryUsage is a Composite attribute, otherwise referred to as a "complex-value attribute" in Zenoss Core. Double click the "javax.management.openmbean.CompositeDataSupport" type and you will see multiple attributes appear. The show the amount of committed, maximum, and used memory sizes for the heap.
The unique name of the MBean can be viewed by clicking on the Info tab. The first value is MBean Name. Its value in the case of Memory is: "java.lang:type=Memory."

**Note** There is no standardized way to name MBeans; application server vendors name them differently.

You can also examine operation information by clicking on the Operations tab. These are methods that JConsole can remotely invoke on an MBean that will result in some value being computed or some state changing in the application. The Threading MBean has several operations that can be invoked that return information. Click on the java.lang package and then click on the Threading operation. Lastly, click on the Operations tab. Methods like "getThreadUserTime" are invocable.
Test the "getThreadUserTime" method by changing the p0 parameter to 1 and clicking the "getThreadUserTime" button. A dialog window will be raised that displays the amount of CPU user time thread #1 has used. Try adjusting the parameter to different values to observe the different CPU times for the threads.

**zenjmx Options**

To display the options supported by the zenjmx daemon, enter the following command:

```
zenjmx help
```

**Memory Allocation**

Use the `--javaheap` option to set the max heap. The value is 512MB.

**ZenJMX Logging**

You can adjust logging levels to reduce the size of ZenJMX log files. In the `log4j.properties` file (in `$ZENHOME/Products/ZenJMX`), update the first line and change DEBUG to INFO, WARN, or ERROR.

**Daemons**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance Collector</td>
<td>zenjmx</td>
</tr>
</tbody>
</table>
Administrating Zen Packs

Installing and upgrading Zen Packs

The ZenPack installation and update procedure requires stopping and starting Zenoss Core.

1. Log in to the Zenoss Core host as **zenoss**.
2. Download the ZenPack or Zen Packs you wish to install or upgrade.
3. Stop Zenoss Core.
   
   **Stopping Zenoss Core** on page 90

4. Start the event server.
   
   
   

5. Install the new Zen Packs.
   
   
   

6. If a ZenPack introduces a new daemon, and you are using `$ZENHOME/etc/daemons.txt`, add the new daemon to the file.

7. Start Zenoss Core.
   
   
   

Removing Zen Packs

The ZenPack removal procedure requires stopping and starting Zenoss Core.

1. Log in to the Zenoss Core master host as **zenoss**.
2. From the list of installed Zen Packs, identify the name of the ZenPack to remove.
   
   
   

3. Stop Zenoss Core.
   
   **Stopping Zenoss Core** on page 90

4. Start the event server.
   
   
   

5. Remove the ZenPack. Replace **Name** with the full name of the ZenPack to remove.
   
   
   

6. If the removed ZenPack provides a daemon, and you are using `$ZENHOME/etc/daemons.txt`, delete the daemon from the file.

7. Start Zenoss Core.
   
   
   

Stopping Zenoss Core

1. Log in to the Zenoss Core master host as **zenoss**.

2. Stop all Zenoss Core daemons.
   
   
   

3. Check for daemons that did not stop.
   
   
   

   - If the command returns no result, Zenoss Core is stopped.
   - If the command returns a result, continue to the next step.

4. Stop the remaining daemons.
   
   
   

Zenoss Core daemons

The daemons that are always part of Zenoss Core.

zenactiond
Runs background jobs such as email notification, database aging, and maintenance window processing.

zencommand
Collects performance data from devices by running commands and parsing the output.

zeneventd
Performs event mappings, transformations, and other data-driven event processing tasks. Forwards processed events to zeneventserver.

zeneventserver
Stores and retrieves events from its database, zenoss_zep. Performs deduplication and clearing; trigger analysis and trigger signal queuing; indexing and storage of the events into Zenoss DataStore; and fanout queuing.

zenhub
zenhubworker
Provides an intermediate connection between collectors and the modeling and event databases. May be run on the master server, on remote servers, or both.

zenjobs
Performs tasks in its queue as scheduled. Large tasks, such as adding a device, may be queued instead of performed immediately. Since version 4.2.2, zenjobs is based on Celery and uses RabbitMQ for job queuing.

zenmodeler
Collects device characteristics through SNMP, SSH, Telnet, and WMI, at scheduled intervals. (Except VMware devices; see zenvmwaremodeler.)

zenperfsnmp
Collects status and performance data from devices through SNMP. Monitoring templates define the data to collect.

zenping
Checks device status and network health with TCP packets. Since version 4.1.1, zenping uses Nmap to build a ping tree and perform Layer 3 event suppression.

zenprocess
Checks the status of processes on modeled devices, at scheduled intervals.

zennrddcached
Provides a queue of writes for RRD data files, to improve the efficiency of collector hosts.

zenstatus
Checks the status of Zenoss Core daemons on localhost. Runs on the Zenoss Core master host and all remote hub and collector hosts.

zensyslog
Collects and classifies syslog events.

zentrap
Collects and parses SNMP traps, resolves OIDs into MIB names, and then forwards the traps to zenhub for additional processing.

zredis
Provides a shared repository for all zenping daemons, and facilitates correlation of "ping down" events.
Daemons from Zen Packs

The daemons that are installed when specific Zen Packs are installed.

<table>
<thead>
<tr>
<th>Daemon</th>
<th>Description</th>
<th>Source ZenPack</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeneventlog</td>
<td>Collects events from Windows Management Instrumentation (WMI) event logs.</td>
<td>WindowsMonitor (Microsoft Windows) on page 67</td>
</tr>
<tr>
<td>zenjmx</td>
<td>Enables monitoring of Java applications by communicating with remote Java Management Extensions (JMX) agents.</td>
<td>(ZenJMX) Java Management Extensions on page 77</td>
</tr>
<tr>
<td>zenpython</td>
<td>Runs Python monitoring code supplied by other Zen Packs.</td>
<td>ZenPacks.zenoss.PythonCollector</td>
</tr>
<tr>
<td>zenwin</td>
<td>Checks the status of Windows services.</td>
<td>WindowsMonitor (Microsoft Windows) on page 67</td>
</tr>
<tr>
<td>zenwinperf</td>
<td>Collects status and performance data from devices through Windows Performance Monitor. Monitoring templates define the data to collect.</td>
<td>WindowsMonitor (Microsoft Windows) on page 67</td>
</tr>
</tbody>
</table>