Chapter No. 5
"Device Management"
In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.5 "Device Management"

A synopsis of the book’s content

Information on where to buy this book

About the Author

**Michael Badger** is a technical writer with a BS in Technical and Professional Communication from the Pennsylvania College of Technology/Penn State. He has been helping users understand, troubleshoot, and use technology for the better part of 15 years. In the 1990's, he rose through the ranks at the industry leading internet service provider, MindSpring, to manage a technical support call center in Dallas, TX. He later found himself supporting and writing about Win4Lin, a Windows virtualization solution for Linux. Today, he prefers to fill a generalist's role with a focus on automated web application testing and writing—always looking to learn the next cool application or technology. For fun, he prefers to be outside in the wilds of Central Pennsylvania fishing, hiking, and hunting.
Zenoss Core Network and System Monitoring

Regardless of the size of your organization, information technology (IT) plays an increasingly important role in day-to-day business, which implies we have incentives to manage the servers, routers, workstations, printers, and other systems attached to our networks. *Zenoss Core Network and System Monitoring: A Step-by-Step Guide for Beginners* provides a narrowly focused guide that helps users set up an environment to manage their IT assets regardless of systems administration background or lack thereof.

We use step-by-step examples with ample screen captures to demonstrate Zenoss Core's capabilities that you can easily apply to your environment. The book keeps the emphasis on using Zenoss Core through its web interface. Advanced users will be able to identify ways in which they can customize the system to do more, while less advanced users will appreciate the ease of use Zenoss provides.

If you work through each chapter in sequence, you will start with installation and finish with monitoring solution that can be deployed on your network. Each chapter builds on the knowledge gained from the previous chapter. However, each chapter can stand on its own, allowing you to pick and choose the features you want to explore.

**What This Book Covers**

*Chapter 1*—Introduction: Provides an overview of Zenoss Core's network and systems management capabilities.

*Chapter 2*—System Architecture: Discusses the underlying components and how they fit together to form Zenoss Core.

*Chapter 3*—Installation and Setup: Details step-by-step instructions for each of the three installation methods—As a virtual appliance, from a binary installer, or compiled from source. Information on how to prepare servers to be monitored is also covered.

*Chapter 4*—Zenoss Dashboard: Introduces the web interface's navigation and organization properties. The dashboard holds the key to the rest of the book. From Chapter 4 onwards, the emphasis is on using the dashboard.

*Chapter 5*—Device Management: Walks through the process of discovering and modeling devices to build an inventory of the network. In Zenoss, everything is viewed as a device, and without devices, we have nothing to monitor.

For More Information:

Chapter 6—Status and Performance Monitors: Describes how to set up monitoring so that we know the operational status of our devices and components, such as file systems, interfaces, and processes.

Chapter 7—Event Management: Provides an in-depth review of how Zenoss Core generates events and how we can manage them from the Event Console.

Chapter 8—System Reports: Takes us on a tour of Zenoss Core's included reporting features. The reports aggregate system-wide data to provide real-time and historical status views about devices, events, and performance.

Chapter 9—Settings and Administration: Documents how to manage users, define alerting rules, and customize event views. Includes information about general Zenoss Core administration, including backups and updates.

Chapter 10—Extend Zenoss: Extend Zenoss Core with ZenPacks, Nagios plugins, and command line utilities.

Chapter 11—Technical Support: The place to start when things go wrong. Outlines the vibrant community support resources and provides a synopsis of how to troubleshoot Zenoss Core.

Appendix A—Event Attributes: A table of available event fields that are used to describe and process events.

Appendix B—TALES and Device Attributes: Provides a list of the device and event attributes available to the Templating Attribute Language Expression Syntax (TALES).
Device Management

Based on the work done in Chapter 4, Zenoss is now monitoring all the devices we automatically added to our inventory and if we look around the web interface, we may notice that some devices have events associated with them. At any moment, we can get the up/down status for each device, but we'll to continue to build a more detailed model of our networks.

We'll start this chapter by fine-tuning our device inventory through manually adding devices to our inventory. Then we'll take a look at the main device status view and perform some routine device administration tasks. The second half of the chapter demonstrates the available monitoring protocols that Zenoss uses to model the devices. Device modeling builds relationships between devices and inventories the services, processes, and hardware on each device.

We'll continue to demonstrate features using the Mill Race network, but feel free to substitute your own devices in the examples given in this chapter. By the time we finish Chapter 5, we'll have a detailed model of our networks that we will continue to build upon in later chapters.

Add Devices

In Chapter 4, we auto-discovered the devices on our networks, but sometimes we don't want to add all the available devices on the network to the inventory or it may be that all our devices may not be found. To compensate for both these scenarios, Zenoss allows us to add one device at a time to the device inventory.

To add a single device, select Add Device from the navigation panel. The Add Device page is divided into multiple sections for general device information, Attributes, and Relations as shown in the following screenshot. We can be as detailed as we want to be when we add the device manually. However, at a minimum, we should enter a Device Name, Device Class Path, and Discovery Protocol.
Device Management

The **Device Name** identifies the IP address or resolvable hostname, while the device class sets the monitoring properties we want our device to inherit by default. If the device is not SNMP-enabled, select **None**, otherwise Zenoss will not add the device.

We’ll continue monitoring our Mill Race location by adding a new Linux server with the following configuration:

Device Name: 192.168.1.110
Device Class: /Server/Linux
Discovery Protocol: None
OS Manufacturer: Ubuntu
Location: /Mill Race/Second Floor
System: /Development
Group: /Developers/Software Testers

The **Add Device Options** table lists the available configuration information we can set when adding a device manually.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Name</td>
<td>Enter either an IP address or resolvable host name to identify the device.</td>
</tr>
<tr>
<td>Device Class Path</td>
<td>Select the appropriate device classifications: For example: /Server/Linux.</td>
</tr>
<tr>
<td>Discovery Protocol</td>
<td>Choose either SNMP or None depending on whether or not you monitor the device with SNMP.</td>
</tr>
<tr>
<td>SNMP Community</td>
<td>Enter the community string of the device. The most common default is public.</td>
</tr>
</tbody>
</table>

**Attributes**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP Port</td>
<td>The default port for SNMP communication is 161.</td>
</tr>
<tr>
<td>Tag Number</td>
<td>If the device has a tag number, such as a service tag number, enter the value.</td>
</tr>
<tr>
<td>Serial Number</td>
<td>Record the manufacturer's serial number.</td>
</tr>
<tr>
<td>Production State</td>
<td>Select the current state of the device: For example: Production, maintenance, decommissioned.</td>
</tr>
<tr>
<td>Priority</td>
<td>Highest, high, normal, low, lowest, trivial.</td>
</tr>
<tr>
<td>Rack Slot</td>
<td>Record the physical rack location of the device.</td>
</tr>
<tr>
<td>Comments</td>
<td>Use the comments to enter device-specific information, including description, device users, or who is responsible for the device.</td>
</tr>
</tbody>
</table>

**Relations**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW Manufacturer</td>
<td>Select a manufacturer name from the list. For example: Cisco or Linksys.</td>
</tr>
<tr>
<td>HW Product</td>
<td>Select a product from the list. The HW Product lists gets populated based on the HW Manufacturer selection.</td>
</tr>
<tr>
<td>OS Manufacturer</td>
<td>Select a manufacturer name from the list. For example: Microsoft or Fedora Core.</td>
</tr>
</tbody>
</table>
Device Management

### Add Device Options

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS Product</td>
<td>Select a product from the list. The OS Product list gets populated based on the HW Manufacturer selection.</td>
</tr>
<tr>
<td>Location Path</td>
<td>Select the location of the device. Create a new location by typing the name in the New Location field and clicking Add.</td>
</tr>
<tr>
<td>Systems</td>
<td>Select a system organizer. Create a new system by typing the name in the New System field and clicking Add.</td>
</tr>
<tr>
<td>Groups</td>
<td>Select a group organizer. Create a new group by typing the name in the New Device Group field and clicking Add.</td>
</tr>
<tr>
<td>Status Monitor</td>
<td>Select a status monitor to define how often the device availability is monitored. The default is localhost. Create a new status monitor by typing the name in the New Status Monitor field and clicking Add. Refer to Chapter 6 for configuration information.</td>
</tr>
<tr>
<td>Performance Monitor</td>
<td>Select a performance monitor to define how often device performance data is collected. The default is localhost. Create a new performance monitor by typing the name in the New Performance Monitor field. Refer to Chapter 6 for configuration information.</td>
</tr>
</tbody>
</table>

After we enter the configuration information for the device, click the Add button. If Zenoss encounters an error while adding the device, the error will be printed in the status window. Check the add device properties and try again. If Zenoss successfully adds the device, the **Status** window displays a log indicating device's properties as shown in the following screenshot.
Chapter 5

The Add Device Status page provides a hyperlink at the bottom of the page that says, "Navigate to device 192.168.1.110." If we click on the device name, the Device Status page is displayed.

Device Status

The Device status page displays an overview of our device and contains the same information we encountered on the Add Device page. As we look at the Device Status table for 192.168.1.110 as shown in the following screenshot, we can determine several important monitoring statistics in one glance.

In our example, the device name and IP address are the same, but they do not need to be the same. If the host has multiple CNAMEs or interfaces, we can specify a name other than the name we used to find the device, via DNS resolution. We may find that we want to implement a custom naming scheme for devices. Regardless of what we name the device, Zenoss uses the IP address to monitor, not the name.

The Device Status table lists the number of events by severity and color code. The Device Severities table lists Zenoss's severity:

<table>
<thead>
<tr>
<th>Color</th>
<th>Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>Critical</td>
</tr>
<tr>
<td>Orange</td>
<td>Error</td>
</tr>
<tr>
<td>Yellow</td>
<td>Warning</td>
</tr>
<tr>
<td>Blue</td>
<td>Information</td>
</tr>
<tr>
<td>Grey</td>
<td>Debug</td>
</tr>
</tbody>
</table>

For More Information:
Device Management

The **Device Status** page also lists important statistics of the device. The **Availability** and **Uptime** values are automatically calculated, and the **Production State** and **Priority** values can be changed via the device's Edit page. We can lock the device to prevent Zenoss from removing or updating the device configuration. The **Last Change**, **Last Collection**, and the **First Seen** values provide a quick way to verify the modeling history of the device by listing the last time Zenoss detected a change with the device configuration and the last time the device was modeled.

In the **Device Status** page, we also see a list of **Component Types** and the **Status** of each monitored component. As we build our monitoring solution, the components we monitor will change per device, but common components include SNMP, ipServices, Windows event logs, and syslogs.

If we look closely at the previous screen shot that shows the status of 192.168.1.110, we notice that the SNMP component displays an error condition. This indicates that our device does not have SNMP installed or is not configured correctly. Refer to the Server Setup section in Chapter 3 for help in SNMP configuration.
Up to this point, we have only added devices to our inventory, so why do we see an error message for SNMP anyway? When we added the device, we set the class /Server/Linux, which implies that the device uses the modeling properties defined in the class. The /Server/Linux class uses the SNMP monitoring by default. We'll talk more about modeling our devices in the modeling section of this chapter.

Like other pages, Zenoss provides context-aware menus that allow us to manage our device from the Device Status page. When we click on the page menu, three submenus display: **More**, **Manage**, and **Run Commands**. As we work through this chapter, we will cover many of the available menu options, but the following series of screen shots provide a quick view of each menu.
Device Management

From the page menu on the Device Status page, we can perform several administration-related tasks, including reset IP address, rename, and lock the device configuration.

Lock Or Unlock Device

Zenoss automatically polls the devices in our inventory and remodels the devices when it finds changes. We can lock the device's configuration from being updated by Zenoss. We can also lock the device from being deleted from the inventory.

To change the lock status of a device:

1. From the Device Status page menu, select Manage > Lock.
2. Select from the following choices as shown in the following screenshot:
   - Send event when actions are blocked by a lock
   - Lock from deletion and updates
   - Lock from deletion
   - Unlock
3. The device status page displays after we choose a locking option.
If we lock the device, the lock status displays a padlock icon on the Device Status page.

**Rename A Device**

Zenoss automatically detects and populates the device name, but we can name the device as anything we want. We'll change the name of our 192.1.168.110 device:

1. From Device Status page menu, select Manage > Rename Device.
2. Enter the new name (e.g., Coyote) in the ID field of the Rename Device dialog.
3. Click OK to save the change.

On the Device Status page, the device information updates to reflect the new name, Coyote as shown in the following screenshot. Even the breadcrumb navigation changes to reflect the name.

The device name will not be updated by the Zenoss modeling process.
Device Management

**Reset IP Address**

If the IP address of a device changes, we need to update Zenoss to reflect the correct configuration. To change the IP address of our newly named Coyote:

1. From Device Status page menu, select **Manage > Reset IP**.
2. Enter the new resolvable host name or IP address in the IP field of the **Reset IP** dialog box (shown in the following screenshot) or leave it blank to allow Zenoss to lookup the IP based on the device name.
3. Click **OK** to save the change.

![Reset IP dialog box](image)

**Push Changes**

After we make changes to the device, we can "push" the changes live right away instead of waiting for Zenoss to remodel the device. From the Device Status page menu, select **Manage > Push Changes**. Zenoss confirms the action with a status message as shown in the following screenshot.
Device List

Up to this point, we have been administering our devices on a per device level, which is acceptable if we only want to make a few changes to one or two devices. If we want to mass update our device properties, we use the **Device List** view.

To display a list of devices, select **Device List** from the navigation panel (Device List is shown in the previous screenshot).

The **Device List** table divides into columns for **Device ID**, **IP**, **Class**, **Production State**, **Event**, and **Locks**, which provides succinct synopsis of the state of our devices.

The device names and classes are hyperlinks that take you to the device's status page and the class' summary pages. The **Event** page displays two squares per device. The squares with the red borders display the critical events, and the squares with no borders display error events. **Events** are listed as the number of acknowledged events over the total number of events.

If we have a large inventory, selecting a device from a list of entries becomes cumbersome, so we can sort the table by **Device ID**, **IP**, **Class**, or **Production State**. Click on the column heading to change the sort order and note the white triangle that shows whether the column is sorted in ascending or descending order. Click the column heading again to reverse the sort order.
If we know the name of the device we want to find, Zenoss provides a global search box that we can use to search by device name or IP. The Device/IP Search box is right-aligned at the top of the page and to the right of the Zenoss Core logo.

If Zenoss finds a device matching the search criteria, it automatically opens the Device Status page. If multiple devices match the search criteria, Zenoss displays a search results page, so that we can select the correct device.

The Device List table also has a search box, but it’s more flexible and allows us to search by the Device Name, IP, Production state, and Class. As an example, enter the search term "linux" and press enter. The list of devices changes to reflect all devices that contain Linux in the name, Production state, or Class (as shown in the following screenshot).

![Device List screenshot](image)

The Device List view not only displays the list of devices, it also allows us to mass update a group of devices by setting properties such as class, groups, locations, status monitors, performance monitors, and production states.

Let's walk through a quick example and change the location for all our devices:

1. Select All devices in the list.
2. From the page menu, select Set Location.
3. From the Set Location dialog box, choose a new location (for example, Mill Race).
4. Click the Set Location button.
This process eliminates the sub-locations that we set in our initial configuration in Chapter 4 and all the devices are now assigned to Mill Race.

**Delete Devices**

If we physically remove a device from our network, we need to update our Zenoss inventory. Otherwise, Zenoss will continue to monitor a device that no longer exists. We can either set the device's production state to decommissioned or delete the device from the Device List. If we change the production state to decommissioned, the device still displays in the Device List, but Zenoss no longer monitors it.

We remove devices from our inventory from the Device List view with just a few steps:

1. Select the device from the list.
2. From the page menu, select **Delete** Devices.
3. Click **OK** to confirm the delete.

The device will no longer show in inventory and Zenoss will not monitor or model it. However, removing the device from the Device List does not remove the performance data associated with the device. If we add the same device name back into Zenoss, the existing performance data will be available. Zenoss stores the performance data by device name in `$ZENHOME/perf`.

---

For More Information:  
Device Management

Model Devices
When we talk about Zenoss, two related but different words often come up, monitoring and modeling. Monitoring refers to the availability of the device and answers the question, "Is the device accessible?" Modeling defines a relationship between devices and identifies the components available on a device, such as services, interfaces, and file systems.

Zenoss models devices via SNMP, SSH, port scan, and telnet and gathers information via collector plug-ins. Each class has a default set of collector plug-ins that tells Zenoss how to model the devices assigned to the class. We can add or remove collector plug-ins at the device level for individual changes or at the class level for all the devices in the class.

The collector plug-in names reflect the monitoring protocol they are used for. All the SNMP collectors contain "snmp" in the name. The SSH and telnet plug-in names contain "cmd," and the port scan plug-in contains "portscan" in the name.

We'll step through modeling examples for SNMP, SSH, and port scan; however, we'll skip telnet because it's similar to SSH.

SNMP
Zenoss defaults to SNMP monitoring, and as we discussed in Chapter 3, the monitored device needs to have SNMP installed and configured to work properly. If you glossed over the SNMP configuration in Chapter 3, take a moment to review the information now.

Zenoss supports SNMP v1, v2c, and v3. The example commands used in this section to troubleshoot SNMP specify v1.

Test SNMP
If we're unsure of our SNMP setup, we can test it by running the snmpwalk command to retrieve the values of the MIB tree on the monitored device. We'll demonstrate both working and broken SNMP configurations on the Mill Race network.

From the Device List, select the device named Coyote. From the Device Status page menu, choose Run Commands > snmpwalk. A new window opens and we see the results of the snmpwalk command as shown in the following screenshot.
Now we select the device Bobcat from the Device List view. Run the `snmpwalk` command from the Device Status page. This time, we receive a `Timeout` error, which indicates that we have a problem with SNMP on the device Bobcat as shown in the following screenshot.

Assuming that SNMP is properly configured on the device and that the monitored device accepts traffic on port 161, we may need to update the device's community string. To update the community:

1. Select the device from the Device List view.
2. From the Device Status page menu, select More > zProperties.
3. Find the Community field and enter the correct value.
4. Save the changes.

After updating the SNMP community string in the zProperties, we run the `snmpwalk` command again to see if we have fixed the problem.
Device Management

If we continue to encounter problems getting Zenoss to model a device with SNMP, we can try to narrow down the problem by running the following `snmpwalk` command from the monitored devices shell prompt:

```
snmpwalk -v1 -c public localhost system
```

Replace `public` with the correct community string. If the command is successful when using localhost, edit the snmpd configuration file. As root, edit `/etc/default/snmpd` or `/etc/default/snmp` and remove `127.0.0.1` from the following line:

```
SNMPDPORTS = '-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid 127.0.0.1'
```

After editing the `/etc/default/snmpd` file, restart the snmpd service as root. For example:

```
/etc/init.d/snmpd stop
/etc/init.d/snmpd start
```

Retest the `snmpwalk` command to confirm that SNMP is working correctly. Windows users can run `wbemtest` from the command line as outlined in Chapter 3 to test SNMP. If problems remain, consult chapter 11 for a list of Zenoss Core community help resources.

Windows Considerations

The Windows SNMP installation is covered in the Server Setup section of Chapter 3, but in order to collect information from WMI, we need to configure the zProperties for the Windows device. Navigate to the Windows device and open the zProperties page by selecting the More > zProperties from the page menu. Scroll to the bottom of the page and make the following changes:

- Set `zWinEventLog` to true.
- Enter the Windows user's password in the `zWinPassword` field.
- Enter the user name with administrative rights in the `zWinUser` field in the following formats:
  - `\user` for local user accounts
  - `DOMAIN\user` for domain user accounts
- Set `zWMIMonitorIgnore` to false.

Save the changes, and Zenoss is ready to model the information on the Windows device. We can force a model by selecting Manage > Model Device from the page menu.
SNMP Collector Plug-ins

The Collector Plug-ins assigned to the device determine how Zenoss models the device. Let's take a look at our example device Coyote and see what collectors are currently assigned. From the Device Status page for Coyote, select More > Collector Plug-ins from the page menu.

A page showing the assigned collector plug-ins displays in the left column of the page with an Add Fields link on the right. When we click on the Add Fields link, a column of unassigned plug-ins appears and the link name changes to Hide Fields as shown in the following screenshot.

The plug-in names are intuitive in that the name suggests the type of information we expect to be modeling. For example, `zenoss.snmp.IpServiceMap` returns a list of active IP services on the device, such as HTTP. The Dell specific plug-ins retrieve more detailed information from Dell devices using OpenManage, and the HP plug-ins provide more information about devices using Insight Management agents.

For More Information:
Device Management

To remove a plug-in from the assigned plug-in list, click on the "x" next to the plug-in name. To assign a plug-in, drag the plug-in name from available list to the assigned list.

To see how our devices are affected, let's remove the `zenoss.snmp.IpServiceMap` and add `zenoss.cmd.df`. After we make the changes to the plug-ins for Coyote, scroll to the bottom of the page and click Save.

Model Device

Zenoss automatically models each device in our inventory every six hours, but we can manually force Zenoss to model the device. From the Device Status page, select Manage > Model Device from the page menu.

Zenoss displays the results of the `zenmodeler` command in the window as shown in the following screenshot.
Zenoss first determines which plug-ins are available and then collects information based on those plug-ins. Notice that no cmd plug-ins are found, which means that the `zenoss.cmd.df` plug-in we added to Coyote will not be collected. After Zenoss models the device, we can review the device overview page to see what component types Zenoss discovered. `IpService` should not be listed.

If we go back to the Collector Plug-ins page for Coyote, we can add the `zenoss.snmp.IpService` plug-in and then model the device again. Now, `IpServices` is displayed in the **Component Type** list as shown in the following screenshot.
Device Management

The Component Type list gets updated as part of the modeling process and so does the OS fields in the Device Information table (the greyed-out fields in the screenshot). If we enter values in these fields during the Add Device step, the values would be overwritten with the SNMP values.

Our example made changes to the device level, which means that if we view the collector plug-ins for the /Server/Linux device class, the original plug-ins are specified. To view the plug-ins for the class:

1. Select Devices from the navigation panel.
2. Select Server from the sub-devices list.
3. Select Linux from the sub-devices list.

Devices automatically inherit any changes we make to the class collector plug-ins the next time Zenoss models the devices.

SSH Modeling

If the monitored device does not support SNMP, or if we need to monitor a device behind a firewall, SSH provides an alternative to SNMP. Unlike SNMP, SSH needs the Zenoss Plug-ins installed on each monitored device and platform support is limited to Linux, Darwin, and FreeBSD. We also need to make sure that the monitored device has an SSH server installed so that the Zenoss system can log in and retrieve information. OpenSSH from openssh.com offers a good cross-platform SSH solution.

The level of modeling provided by the Zenoss Plug-ins varies between platforms. For this reason, we may not achieve the same level of detail as we do with SNMP, but SSH modeling provides more detail than a port scan.

To help us setup our SSH monitoring, Zenoss provides the /Server/Cmd class which is already configured with the command plug-ins we need to monitor via SSH.

SSH Collector Plug-ins

From the navigation panel, select Devices. Navigate to the /Server/Cmd class and click on the zProperties tab. Find the zCollectorPlug-ins field and click on the Edit link. A list of the assigned collector plug-ins is displayed as shown in the following screenshot.
The important thing to note with the command collector plug-ins is the new level of specificity in the name. The `zenoss.cmd.uname` and `zenoss.cmd.df` plug-ins are common to all architectures, while the plug-ins with "linux" in the name work with Linux systems. Mac OS X platforms use the plug-ins with "darwin" in the name.

If we did not have any any OS X systems to monitor, then we could remove all the Darwin-based plug-ins from the `/Server/Cmd` class or, if we know that we don't want to monitor the memory usage for any of our devices, we can remove that plug-in.

We'll leave the collector plug-ins as they are for the `/Server/Cmd` class and change our test device Coyote to use SSH instead of SNMP.

**Zenoss Plug-ins**

Zenoss will monitor and retrieve some data using the SSH modeler even if we do not install the Zenoss plug-ins, but the device model will be incomplete. For example, file systems will be detected along with the size of each drive., but the usage statistics will not be reported. Zenoss also generates warning events if it cannot find the `zenplugin.py` command on the monitored system.
Device Management

The monitored system needs a Python environment installed. This can be installed using your distribution's package manager. If you have setuptools installed, you can install the Zenoss-Plug-ins package from the Cheese Shop (http://pypi.python.org/pypi/) with the following command as root:

```
$ sudo easy_install Zenoss-Plugins
```

We can also build the Zenoss Plug-ins package from source:

2. Extract the plug-in file.
3. From the plug-in source directory, run the following commands as root:
   
   - python setup.py build
   - python setup.py install

The setuptools procedure installs `zenplugin.py` to `/usr/bin`, which is important because we need to configure the device zProperties to look for the plug-ins in the correct location.

To ensure that the plug-in file is working correctly, run the following command on the monitored device, which is Coyote in our example:

```
zenplugin.py -list-plugins
```
The command outputs the detected platform and the supported plug-ins as shown in the following screenshot.

```
mike@coyote:~$ zenplugin.py --list-plugin
platform 'linux2' supports the following plugins:
   process
   mem
   disk
   cpu
   io
mike@coyote:~$
```

**Model Device**

In order to get Zenoss to model Coyote, we need to tell Zenoss how to connect. First, we'll change the class to /Server/Cmd because it's already configured with the plug-ins we need to use. Second, we'll configure the zProperties so that Zenoss can log in to the device and run system commands.

Navigate to Coyote's Device Status page, and from the page menu, select **Manage > Change Class**. Select /Server/Cmd and confirm the selection by clicking **OK**.

Next open the zProperties. From the page menu, select **More > zProperties**. Make the following changes:

- Set zCommandUsername to the SSH login on the monitored device.
- Set the user's password in zCommandPassword.
- Change zCommandPath to /usr/bin.
- Set zSnmpMonitorIgnore to true.

When we type the password in zCommandPassword, it will be in clear text, but after we save the zProperties, the password will be starred out. After we have all the changes entered, click **Save**.

Find the zCollectorPlug-ins field and click on the **Edit** link to display the collector plug-ins page. Verify the plug-ins listed are for the /Server/Cmd class only. If not, remove the SNMP plug-ins and add the cmd plug-ins. Save any changes.
Device Management

Now, let's model the device. From the page menu, select Manage > Model Device.

Port Scan Modeling

Sometimes, the only option we have to model our devices is with a port scan. A port scan tries to guess which services are running on a device by connecting to various ports. Port scans provide the least detailed model and may raise security alerts on your network. Consult the security administrators before port scanning devices on the network.

Zenoss creates a separate device class in /Server/Scan to handle these devices. There is only one plug-in available, and it is named zenoss.portscan.IpServiceMap. As the name implies, it returns a list of services running on the device.

We go through the same steps to model a device with port scan as we do for SNMP and SSH.
OS Tab

After Zenoss models the devices, it populates the operating system (OS) tab with its findings. From the device's Status page, click on the OS tab.

We'll discuss each of the sections in more detail in Chapter 6, but we see that Zenoss has detected the Interfaces, IP Services, File Systems, and Routes for our server, Fox as shown in the previous screenshot. Those groupings should sound familiar, as we've seen various implementations of those collector plug-ins for SNMP and SSH plug-ins.

[91]
Device Management

Hardware Tab

With the exception of the port scan, the Zenoss models include information about a device's memory and CPU. We can access the Hardware tab by clicking the tab labeled Hardware from the Device Status page.

![Hardware Tab](image)

If we monitor a Windows system, we can gather hard disk information by adding the `zenoss.snmp.InformantHardDiskMap` collector plug-in to the device.

Device zProperties

In the course of the chapter, we have changed several zProperties at the device or class level to define how we monitor our devices. The following table lists the available zProperties and a description of each.

<table>
<thead>
<tr>
<th>zProperty</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zCollectorClientTimeout</td>
<td>Set the timeout of the client collector in seconds. The default is 180.</td>
</tr>
<tr>
<td>zCollectorDecoding</td>
<td>Specify the character encoding. The default is latin-1.</td>
</tr>
<tr>
<td>zCollectorLogChanges</td>
<td>Set to true to log changes and false not to log changes to the collector.</td>
</tr>
<tr>
<td>zCollectorPlug-ins</td>
<td>Click the Edit link to open the collector plug-in selection page.</td>
</tr>
<tr>
<td>zCommandCommandTimeout</td>
<td>Time in seconds to wait for a command to finish. The default is 15.</td>
</tr>
<tr>
<td>zCommandCycleTime</td>
<td>Specifies a time in seconds to cycle through zCommands. The default is 60.</td>
</tr>
<tr>
<td>zCommandExistanceTest</td>
<td>Test to see if a command exists on the monitored device. The default is 'test -f %s'.</td>
</tr>
<tr>
<td>zCommandLoginTimeout</td>
<td>Wait for the specified seconds for a login prompt. The default is 10.</td>
</tr>
</tbody>
</table>
### zProperty Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zCommandLoginTries</td>
<td>Attempt to log in the number of specified times. The default is 1.</td>
</tr>
<tr>
<td>zCommandPassword</td>
<td>Enter the password for the user's shell account on the monitored device.</td>
</tr>
<tr>
<td>zCommandPath</td>
<td>Enter the path of the <code>zenplugin.py</code> command. The default is <code>/opt/zenoss/libexec</code>; however, the default installation path for <code>zenplugin.py</code> is <code>/usr/bin</code></td>
</tr>
<tr>
<td>zCommandPort</td>
<td>The port the monitored system uses for SSH connections. The default is 22.</td>
</tr>
<tr>
<td>zCommandProtocol</td>
<td>Specify the protocol (telnet, ssh) to use. The default is SSH.</td>
</tr>
<tr>
<td>zCommandSearchPath</td>
<td>Specify all the paths to search for the commands.</td>
</tr>
<tr>
<td>zCommandUsername</td>
<td>Enter the user log in name for the monitored device.</td>
</tr>
<tr>
<td>zDeviceTemplates</td>
<td>Enter the templates by name to use to display information. The default is device.</td>
</tr>
<tr>
<td>zFileSystemMapIgnoreNames</td>
<td>Enter the names of the files system to ignore. For example: <code>/boot</code>.</td>
</tr>
<tr>
<td>zIcon</td>
<td>Each device class has a default icon that can be changed as necessary.</td>
</tr>
<tr>
<td>zIfDescription</td>
<td>Displays the interface description on the Interfaces table of the OS tab. Select either true or false. The default is false.</td>
</tr>
<tr>
<td>zInterfaceMapIgnoreNames</td>
<td>Enter the names of the interfaces to ignore. For example: <code>lo</code>.</td>
</tr>
<tr>
<td>zInterfaceMapIgnoreTypes</td>
<td>Enter the type of interfaces to ignore. For example: <code>local loopback</code>.</td>
</tr>
<tr>
<td>zIpServiceMapMaxPort</td>
<td>Specify the maximum port number to port scan. The default is 1024.</td>
</tr>
<tr>
<td>zKeyPath</td>
<td>Specify the path to the user's public key file for use with public-key authentication.</td>
</tr>
<tr>
<td>zLinks</td>
<td>Enter HTML markup or TALES expressions to display a link for the device. For example, you can create a link to a router's administration console that will display on the Device Status page.</td>
</tr>
</tbody>
</table>
### Device Management

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zLocalInterfaceNames</td>
<td>A regular expression match to identify local interface names. The default expression looks for lo (loopback) and vmnet (Vmware).</td>
</tr>
<tr>
<td>zLocalIpAddresses</td>
<td>A regular expression match to identify local IP address.</td>
</tr>
<tr>
<td>zMaxOIDPerRequest</td>
<td>Specify the number of OIDs Zenoss collects with a single query. The default is 40.</td>
</tr>
<tr>
<td>zPingInterfaceDescription</td>
<td>Find interfaces to ping by device description.</td>
</tr>
<tr>
<td>zPingInterfaceName</td>
<td>Find interfaces to ping by name.</td>
</tr>
<tr>
<td>zPingMonitorIgnore</td>
<td>Select true not to ping the device or false to ping the device.</td>
</tr>
<tr>
<td>zProdStateThreshold</td>
<td>Monitor a service that is higher than the production state listed. Possible values include 1000 (Production), 500 (Pre-Production), 400 (Test), 300 (Maintenance), and -1 (Decommissioned).</td>
</tr>
<tr>
<td>zRouteMapCollectOnlyIndirect</td>
<td>Set to true to collect only the indirect routes. Default is false.</td>
</tr>
<tr>
<td>zRouteMapCollectOnlyLocal</td>
<td>Set to true to collect only the local routes. Default is false.</td>
</tr>
<tr>
<td>zSnmpAuthPassword</td>
<td>Specify SNMP password, if applicable.</td>
</tr>
<tr>
<td>zSnmpAuthType</td>
<td>If using zSnmpAuthPassword, select either MD5 or SHA authentication protocol.</td>
</tr>
<tr>
<td>zSnmpCommunities</td>
<td>List of communities Zenoss tries to collect information for. The defaults are public and private. Enter more as needed.</td>
</tr>
<tr>
<td>zSnmpCommunity</td>
<td>The default community name on the monitored device.</td>
</tr>
<tr>
<td>zSnmpMonitorIgnore</td>
<td>Set whether or not Zenoss should monitor the device with SNMP. Defaults to false.</td>
</tr>
<tr>
<td>zSnmpPort</td>
<td>The SNMP communication port. Defaults to port 161.</td>
</tr>
<tr>
<td>zSnmpPrivPassword</td>
<td>Enter the security user's password.</td>
</tr>
<tr>
<td>zSnmpPrivType</td>
<td>Select either AES or DES encryption.</td>
</tr>
<tr>
<td>zSnmpSecurityName</td>
<td>enter the security user's name.</td>
</tr>
<tr>
<td>zSnmpTimeout</td>
<td>Length of time in seconds that Zenoss waits for a response from the remote SNMP agent. Defaults to 2.5.</td>
</tr>
<tr>
<td><strong>zProperty</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>zSnmpTries</td>
<td>Number of times Zenoss tries to connect via SNMP before reporting a failure.</td>
</tr>
<tr>
<td>zSnmpVer</td>
<td>The version of SNMP. Available options are 1, 2c, and 3. Defaults to 1.</td>
</tr>
<tr>
<td>zStatusConnectTimeout</td>
<td>Specifies the time in seconds for an IP service to respond before the service is marked down. The default is 15.</td>
</tr>
<tr>
<td>zSysedgeDiskMapIgnoreNames</td>
<td>Not used.</td>
</tr>
<tr>
<td>zTelnetEnable</td>
<td>On Cisco routers, send the enable command to enable command collection. Default is false.</td>
</tr>
<tr>
<td>zTelnetEnableRegex</td>
<td>Match the enable prompt with the specified regular expression.</td>
</tr>
<tr>
<td>zTelnetLoginRegex</td>
<td>Match the login prompt with the specified regular expression.</td>
</tr>
<tr>
<td>zTelnetPasswordRegex</td>
<td>Match the password prompt with the specified regular expression.</td>
</tr>
<tr>
<td>zTelnetPromptTimeout</td>
<td>Specify the time in seconds to wait for the login prompt to display.</td>
</tr>
<tr>
<td>zTelnetSuccessRegexList</td>
<td>Match the command prompt with the specified regular expression.</td>
</tr>
<tr>
<td>zTelnetTermLength</td>
<td>Select true to enable telnet terminal length.</td>
</tr>
<tr>
<td>zWinEventLog</td>
<td>Specifies whether or not Zenoss collects the Windows event log. Default is false.</td>
</tr>
<tr>
<td>zWinEventLogMinSeverity</td>
<td>Collect all Windows event logs that match the specified severity. Enter a value between 1 and 5, where 1 is the most severe. The default is 2.</td>
</tr>
<tr>
<td>zWinPassword</td>
<td>Enter the Windows user's password.</td>
</tr>
<tr>
<td>zWinUser</td>
<td>Enter the user name of an account on the monitored Windows system.</td>
</tr>
<tr>
<td>zWmiMonitorIgnore</td>
<td>Set to true to ignore WMI monitoring and set to false to monitor WMI services.</td>
</tr>
<tr>
<td>zFileSystemMapIgnoreTypes</td>
<td>Do not use</td>
</tr>
<tr>
<td>zPythonClass</td>
<td>Do not use</td>
</tr>
<tr>
<td>zXmlRpcMonitorIgnore</td>
<td>Set to true to enable XML/RPC monitoring</td>
</tr>
</tbody>
</table>
Summary

As we see, Zenoss aggregates a large amount of information about our networks. In this chapter, we've learnt how to use classes, plug-ins, and modeling protocols to organize, collect, and display information about our devices. By using the device classes, we can define a hierarchical set of monitoring properties for groups of devices.

The classes allow us to set the collector plug-ins and define a common set of zProperties per device. Exceptions can be made on a per device basis. This is one of Zenoss's core data organization concepts. We can change a device's class or zProperties at any time and Zenoss will apply the changes the next time it models the device.

In Chapter 6, we will review status and performance monitors, and we will also monitor individual device components based on the device models we generated in this Chapter. We'll monitor TCI/IP services, processes, file systems, CPUs, and interfaces.
Where to buy this book


Free shipping to the US, UK, Europe, Australia, New Zealand and India.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.