Zenoss™
Open Enterprise Management

Zenoss Administration

Version 2.3

Zenoss, Inc.
www.zenoss.com

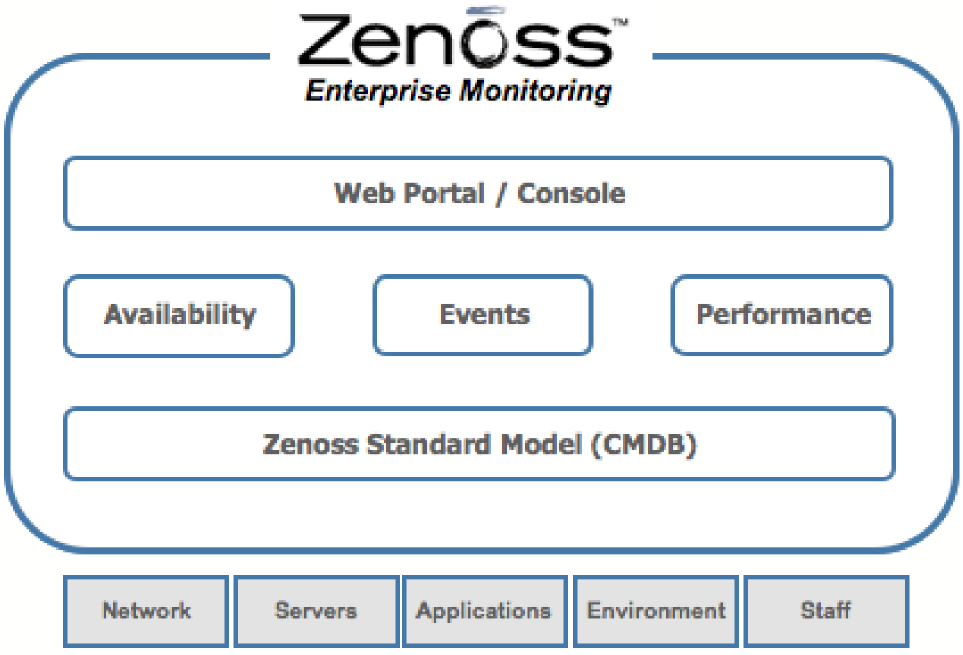
Zenoss Administration for Version 2.3

Chapter 1. Introduction

1. Zenoss Overview

The Zenoss system brings together many types of monitoring and management information. This information is available through a standard web browser. In fact, all aspects of the system are accessed though the web there is no need to edit configuration files. At a high level, Zenoss consists of four major parts:

Figure 1.1. Zenoss High-Level Architecture



1.1. The Zenoss Standard Model

At the heart of Zenoss is the Standard Model. The model is a detailed description of any device Zenoss manages and that device's relationship to your other devices, business units or any other important groupings you define. Because of the high degree of detailed information in the model, there are several ways the model's information is populated. The primary way information is added to the model is through a process Zenoss calls "auto-discovery". Auto-discovery is defined as Zenoss using one of the available transports to discover the services, interfaces etc on a device. From this information, Zenoss builds a model of the device in the system. The model can also be populated by adding and manipulating data associated with the device though the web UI (or through Zenoss' external APIs). Version 2.0 adds discovery locking, which allows auto-discovered information to be tightly integrated with manually added information. The model is used to drive all monitoring elements of the Zenoss system, which will be described throughout the rest of this document.

1.2. Availability Monitoring in Zenoss

Availability monitoring in Zenoss consists of the system running tests against the IT infrastructure to determine if it is currently functioning properly. These tests are typically run external to the monitored system. Examples tests include: ping tests, process tests, and service tests.

1.3. Zenoss Event Management System

The Zenoss Event Management System is a consolidation of status information from all parts of the Zenoss system as well as any monitored external systems. When a Zenoss monitoring daemon detects a failure or threshold breach, events are generated. This is similar to most other monitoring systems available. Zenoss does more in that it also incorporates event imports from other parts of the IT infrastructure. These include Syslog and SNMP Traps. It is one thing to bring the events into a single repository but an event management system must do more. As events are received, Zenoss runs them through a set of rules that augment the information contained therein and integrates all of this information into the Zenoss Model.

1.4. Zenoss Performance Monitoring System

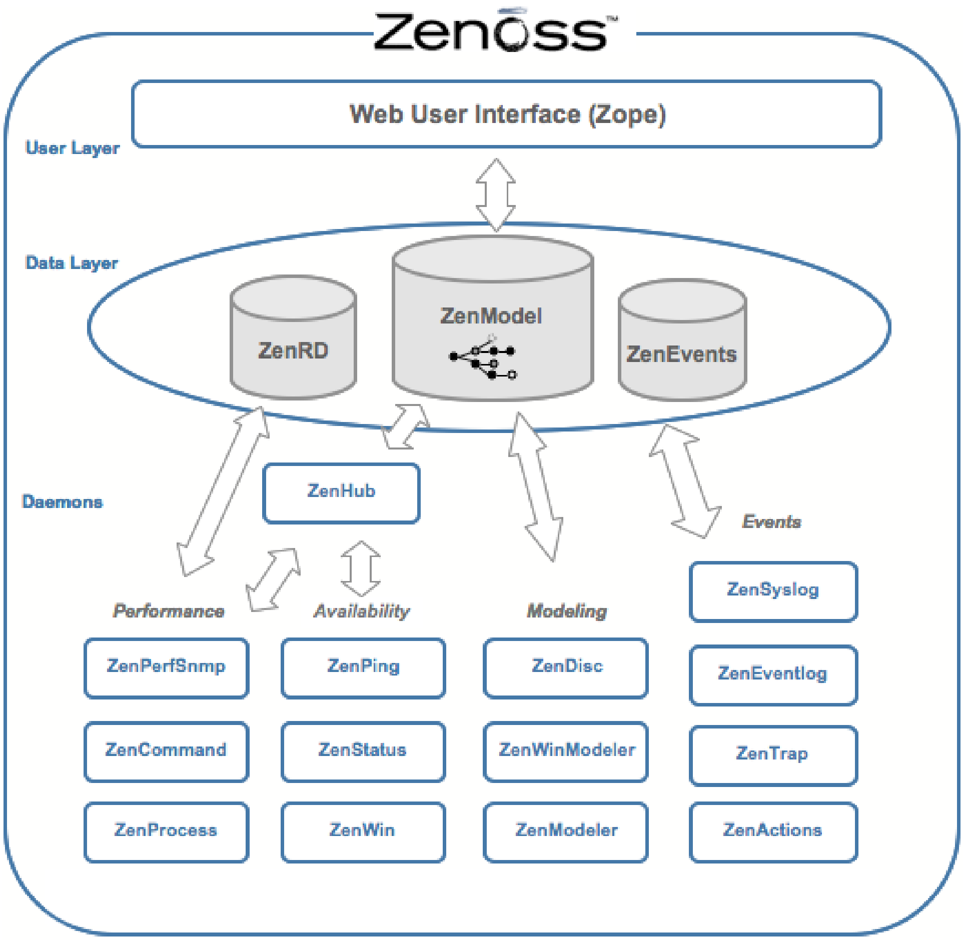
The Zenoss Performance Management System tracks important IT resource information and tracks these changes over time. It is critical to know how much disk space is available, what the CPU load is, or how long a web page takes to download. Zenoss can collect information using SNMP, custom scripts (ZenCommands) or XML-RPC. Performance information is integrated with the Zenoss Model so that resource usage is shown in the context of other Zenoss information.

Chapter 2. Detailed Architecture

1. Zenoss Detailed Architecture

The following diagram shows a more detailed view of the architecture of the Zenoss system.

Figure 2.1. Zenoss Detailed Architecture



The diagram shows the 3 distinct layers of the Zenoss system: The User Layer, the Data layer, and the Collection and Control Layer where the Zenoss Daemons reside and perform their duties.

2. User Layer

The User Layer is manifested as a Web Console/Portal. This layer consists of the Graphical User Interface (GUI), which allows the user access to the following pieces of information:

Dashboard	Events	Locations
Devices	Manufacturers	Reports
Services	Systems	Users
Networks	Groups	Administration

The User Layer Interacts with the Data layer and translates the information for display in the GUI.

3. Data Layer

The Data Layer is where all of the system information is stored. This layer consists of the Zenoss Daemons as well as zeoctl and zopectl to run the heart of the system. Zeoctl is the back-end object database that stores the configuration model, and zopectl controls the zope web application development environment used to display the console.

Daemon	Description
ZenrRRD	ZenRRD gathers Time Series Data and acts as an RRDtool.
Zenevents	Zenevents interacts with the MySQL Events Database.
Zenmodel	Zenmodel Is the United Configuration Model of the Zope object database.
Zenhub	Broker of information between the data layer and the collection daemons.

4. Collection and Control Services Layer

The services that collect the data and feed it to the Data Layer come from the daemons associated with the Collection and Control Services Layer. These daemons can be broken down into five distinct areas: Automated Modeling, Availability Monitoring, Event Collection, Performance Monitoring, or Automated Response. The daemons that fall under each layer are detailed below.

4.1. Automated Modeling Daemons

Daemon	Description
Zendisc	Zendisc is a subclass of zenmodeler and it goes out to discover new network resources. It walks the routing table to discover the network topology and then pings all discovered networks to find active IPs and devices.
ZenwinModeler	ZenWinModeler is used for the auto-discovery of Windows Services (WMI) running on a windows box.
ZenModeler	ZenModeler is a configuration collection and configuration daemon. It is used for high-performance, automated model population using SNMP, SSH, and Telnet to collect its information. Zenmodeler works against devices that have been loaded into the DMD.

4.2. Availability Modeling Daemons

Daemon	Description
Zenping	Zenping is the ping status monitoring (ICMP) for Zenoss. Zenping does the high-performance asynchronous testing of the ICMP status.
Zenstatus	Zenstatus performs active TCP connection testing of remote daemons.
Zenprocess	Zenprocess enables process monitoring using SNMP host resources MIB.

4.3. Event Collection Daemons

Daemon	Description
Zensyslog	Zensyslog is collection of and classification of syslog events.
Zeneventlog	Zeneventlog is used collect (WMI) event log events.
Zentrap	Zentrap collects SNMP Traps. It receives traps and turns them into events.

4.4. Performance Monitoring Daemons

Daemon	Description
ZenperfSNMP	ZenperfSNMP does the high performance asynchronous SNMP performance collection.
ZenperfXMLrpc	ZenperfXMLrpc is used for XML RPC Collection.
Zencommand	Zencommand is used for XML RPC Collection specifically it allows the running of Nagios© and Cactii plugins on the local box or on remote boxes through SSH.

4.5. Automated Response Daemons

Daemon	Description
Zenactions	Zenactions is used for alerts (SMTP, SNPP and Maintenance Windows).

Chapter 3. Key Concepts

1. Classification

Many of Zenoss's hierarchies are used to classify IT entities, things like Devices (computers, routers, switches, etc) or Events (status information sent out by devices). Once an item is properly classified the system understands more about the item. This makes proper classification an important activity in the system. Often classification happens automatically with the ability for manual override later.

2. zProperties

Zenoss allows configuration to be specified using its hierarchical organization system. zProperties are properties applied to devices or groupings that help maintain more specific control in different modules of Zenoss. zProperties can be set at any level of a hierarchy, and values set at lower levels of the hierarchy override those above.

3. Inheritance and Path Navigation

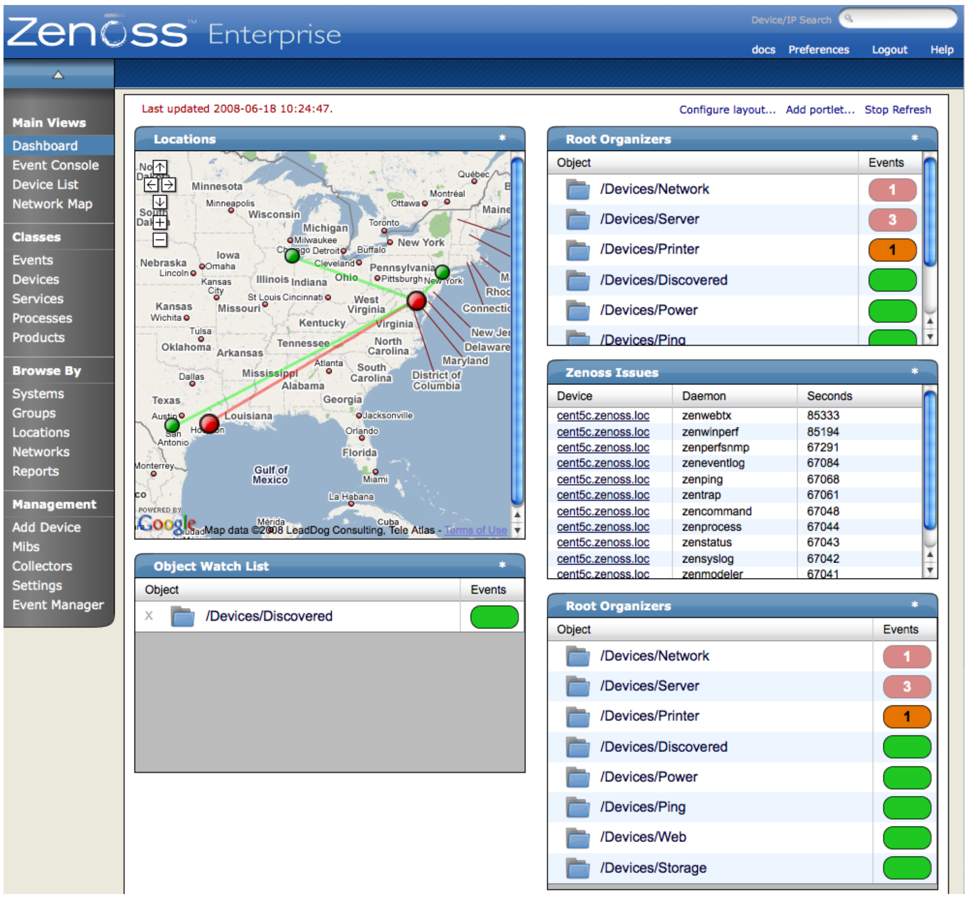
Zenoss uses hierarchies much like a file system does to organize information. These hierarchies are navigated using paths just like a UNIX file system or a web URL. The paths are navigating objects in a database though, and not an actual file system.

Chapter 4. Zenoss Interface and Navigation

1. Zenoss Dashboard

Once you have installed Zenoss and entered the URL for the UI, the Zenoss Dashboard appears. The Dashboard is the primary window into the devices, events and activities within the Zenoss system. The Dashboard shows Systems-level event summaries, devices that currently have events with severity of at least “Error” magnitude, and infrastructure issues along with a navigational bar. There is a search field, where you can enter a machine’s name or partial name or IP address. All or some of a machine's name may be typed in to search for it, as well as an IP address. User information may be accessed through the “Preferences” link in the top right of the screen. Below the Preferences link is the time and date that Zenoss was last updated. Every 60 seconds there is an AJAX call that refreshes the data fields and polls for new data. If the poll fails, it will display “Lost Connection to Zenoss” near the User information area.

Figure 4.1. Zenoss Dashboard



Last updated 2008-06-18 10:24:47.

Map data ©2008 LeadDog Consulting, Tele Atlas - Terms of Use

Root Organizers

Object	Events
/Devices/Network	1
/Devices/Server	3
/Devices/Printer	1
/Devices/Discovered	🟩
/Devices/Power	🟩
/Devices/Ping	🟩

Zenoss Issues

Device	Daemon	Seconds
cent5c.zenoss.loc	zenwebtx	85333
cent5c.zenoss.loc	zenwinperf	85194
cent5c.zenoss.loc	zenperfsnmp	67291
cent5c.zenoss.loc	zeneventlog	67084
cent5c.zenoss.loc	zenping	67068
cent5c.zenoss.loc	zentrap	67061
cent5c.zenoss.loc	zencommand	67048
cent5c.zenoss.loc	zenprocess	67044
cent5c.zenoss.loc	zenstatus	67043
cent5c.zenoss.loc	zensyslog	67042
cent5c.zenoss.loc	zenmodeler	67041

Object Watch List

Object	Events
X /Devices/Discovered	🟩

Root Organizers

Object	Events
/Devices/Network	1
/Devices/Server	3
/Devices/Printer	1
/Devices/Discovered	🟩
/Devices/Power	🟩
/Devices/Ping	🟩
/Devices/Web	🟩
/Devices/Storage	🟩

2. Left Navigation Menu

The Left Navigation menu consists of the three main sections: the Main Views, Classes, Browse By, and a Management Section.

Hiding the Left Navigation Menu

You can use the triangle above the left navigation menu to hide the menu, and then you can use the pin icon to “pin” the menu into place and the left navigation menu will remain visible.

2.1. Main Views

The Main Views section of the Navigational Bar consists of the following views: The Dashboard, which returns you to the main view; The Event Console, which will show a list of all of the current events in the Event Database; The Device List, which shows you a list of all of the devices in the Zenoss system; and the Network Map link which shows you a graphical representation of the devices in your network.

2.2. Classes

The Classes area contains the following links: The Events link will take you to the event management tabs where you can monitor event status, events, history, zProperties, event transforms and also track any changes made to events. The Devices link allows you to manage sub-devices and a summary of events by severity. You can also look at events sorted by severity followed by device name, history of device events, PerfConfig, zProperties for Devices, and recent changes made to the Devices. The Services link allows you show service Classes, administer commands on a Service basis and also access zProperties and track any changes made to Services monitoring. Processes Classes allows you to create new process groupings)here called Sub-folders) and also add processes to monitor. You can also set a Sequence or order in which to gather process monitoring information. You can also use the Process Class to run commands against processes, or access process zProperties. The Products link will show you a list of all the manufacturers of devices in the Zenoss database.

2.3. Browse By

Use the "Browse by" areas to see data based on any of the logical groupings Zenoss allows you to create. Either "Systems", a more generic "Groups", the physical location-related "Locations", a networking based "Networks", or you can see and define "Reports". Browsing by Systems allows you to see network statuses broken down into the system groupings you have created. Show performance data based on these systems and also see Events and the Event history for these systems. You can also Use the Administration tab to define commands, create maintenance windows or define administered objects on a Systems basis. Browsing by Groups allows you access to the same kind of data accessed when Browsing by System (only here on a "Group" basis) with the notable exception of tracking Performance data on a Group basis. Browsing by Locations refer to seeing data related to devices grouped by physical locations. Here devices can only be in one location but multiple nested sub-locations. A device can reside in several Groups in the groupings area, but should only reside in one physical location at a given level of a hierarchy. Nesting in sub-locations is perfectly acceptable, though. For example, a device can reside in the Location "Maryland", and also in sub-location Annapolis, but not simultaneously in the sub-location "Bethesda". Even though both are sub-locations you can define under "Maryland". You can also see a cool Google-based map of the physical locations of your devices and see status that way. Browsing by Networks, will show you the devices and sub-networks, based on IP address groupings. You can also see and access zProperties based on IP address groupings and see any changes made to any Networks. The Reports option allows you to see and define the Reports available within the Zenoss system.

2.3.1. Management Menu

2.3.2. Adding New Devices

Use the Add Device left navigation menu to add new devices into Zenoss. For more information about adding devices see the "Adding Devices" section in this guide.

2.3.3. Adding and Managing MIBS

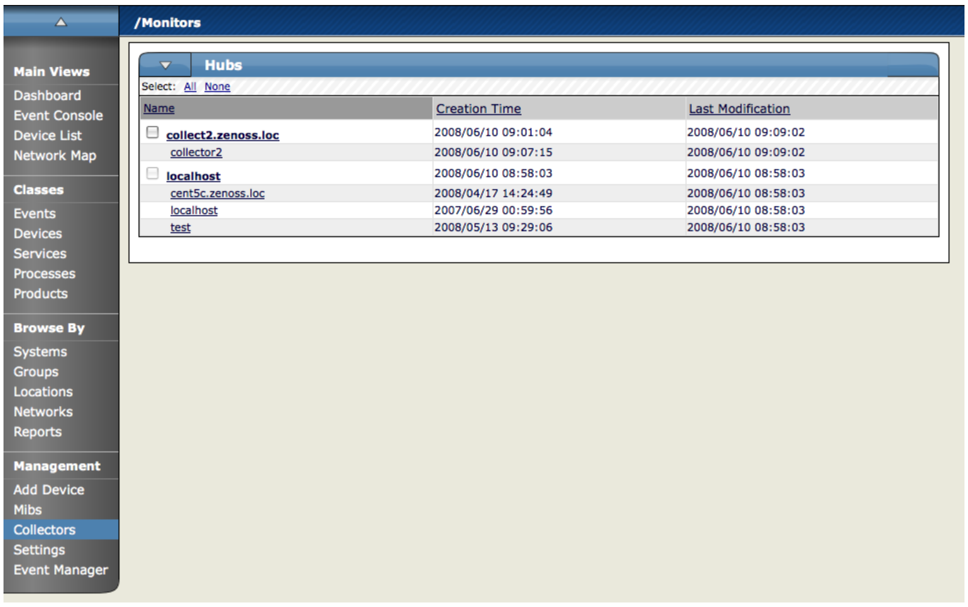
You can use the Mibs link in the left navigation menu to add and manage MIBs in Zenoss. The preferred method of MIB loading is still to do it from the command line using zenmib (as decribed in the section called "Loading and Registering MIBs with Zenoss". This tab is especially useful for checking MIBs you have loaded through the command line. You can, however use the Mibs link to create and build your own Mibs. To add a MIB, select the Mibs link and from the table menu and select Add Mib. You can add all of the

You can also use this table menu to delete MIBS and move them into different sub folders.

2.3.4. Adding Additional Collectors and Scaling Zenoss

You can add additional collectors to Zenoss to allow for better performance and scaling when the device count gets too large for one collector. This number is somewhere between 1000 and 1200 devices. When a Zenoss collector is collecting data from more than that number of devices, it becomes harder for the collector to complete the collection cycle in the given cycle interval time. The default collector is localhost, but you can add another instance of Zenoss on another server to act as a collector. In prior versions of Zenoss, these collectors were known as Performance Monitors and Status Monitors.

Figure 4.2. Collectors Tab



/Monitors

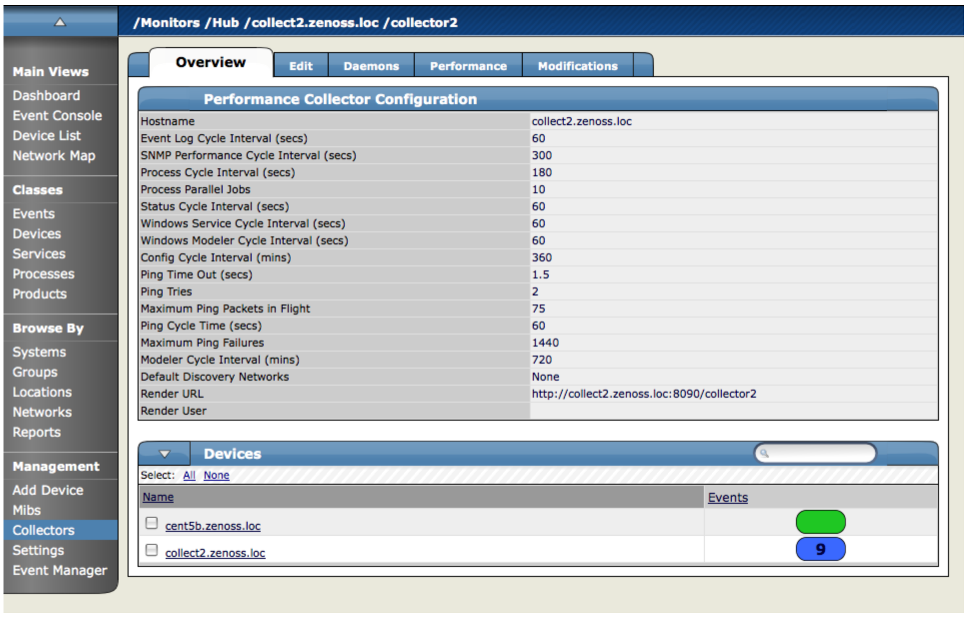
Hubs

Select: All None

Name	Creation Time	Last Modification
collect2.zenoss.loc	2008/06/10 09:01:04	2008/06/10 09:09:02
collector2	2008/06/10 09:07:15	2008/06/10 09:09:02
localhost	2008/06/10 08:58:03	2008/06/10 08:58:03
cent5c.zenoss.loc	2008/04/17 14:24:49	2008/06/10 08:58:03
localhost	2007/06/29 00:59:56	2008/06/10 08:58:03
test	2008/05/13 09:29:06	2008/06/10 08:58:03

If you click the name of a collector you will see all of the attributes of a given collector. You can also see all of the devices that this collector will monitor.

Figure 4.3. Individual collector attributes



/Monitors /Hub /collect2.zenoss.loc /collector2

Performance Collector Configuration

Hostname	collect2.zenoss.loc
Event Log Cycle Interval (secs)	60
SNMP Performance Cycle Interval (secs)	300
Process Cycle Interval (secs)	180
Process Parallel Jobs	10
Status Cycle Interval (secs)	60
Windows Service Cycle Interval (secs)	60
Windows Modeler Cycle Interval (secs)	60
Config Cycle Interval (mins)	360
Ping Time Out (secs)	1.5
Ping Tries	2
Maximum Ping Packets in Flight	75
Ping Cycle Time (secs)	60
Maximum Ping Failures	1440
Modeler Cycle Interval (mins)	720
Default Discovery Networks	None
Render URL	http://collect2.zenoss.loc:8090/collector2
Render User	

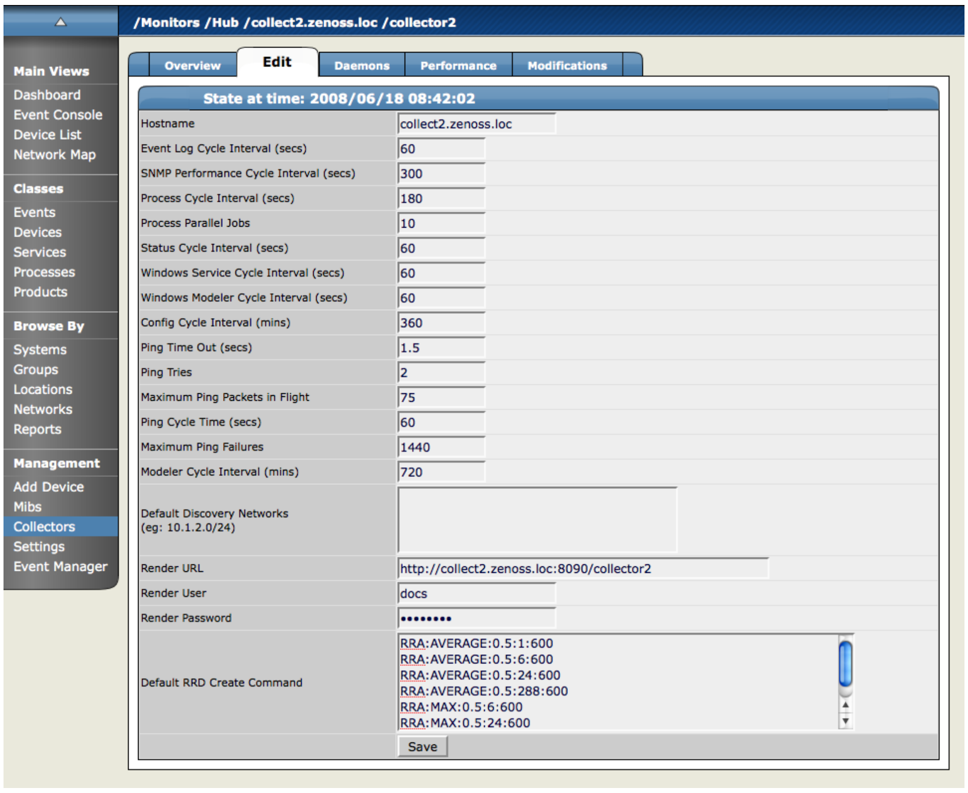
Devices

Select: All None

Name	Events
cent5b.zenoss.loc	🟩
collect2.zenoss.loc	9

If you click the Edit tab you can change any of the following settings to better optimize this collectors performance against the devices you want it to poll.

Figure 4.4. Individual Collector Edit tab



State at time: 2008/06/18 08:42:02

Hostname	collect2.zenoss.loc
Event Log Cycle Interval (secs)	60
SNMP Performance Cycle Interval (secs)	300
Process Cycle Interval (secs)	180
Process Parallel Jobs	10
Status Cycle Interval (secs)	60
Windows Service Cycle Interval (secs)	60
Windows Modeler Cycle Interval (secs)	60
Config Cycle Interval (mins)	360
Ping Time Out (secs)	1.5
Ping Tries	2
Maximum Ping Packets in Flight	75
Ping Cycle Time (secs)	60
Maximum Ping Failures	1440
Modeler Cycle Interval (mins)	720
Default Discovery Networks (eg: 10.1.2.0/24)	
Render URL	http://collect2.zenoss.loc:8090/collector2
Render User	docs
Render Password	••••••••
Default RRD Create Command	RRA:AVERAGE:0.5:1:600 RRA:AVERAGE:0.5:6:600 RRA:AVERAGE:0.5:24:600 RRA:AVERAGE:0.5:288:600 RRA:MAX:0.5:6:600 RRA:MAX:0.5:24:600

- Event Log Cycle Interval (secs) - The Event Log Cycle interval tells the controller The default is 60 secs.

- SNMP Performance Cycle Interval (secs) - The SNMP Performance Cycle Interval tells the controller at what interval to run the SNMP performance cycle to get performance data from the devices. The default is 300 seconds.

- Process Cycle Interval (secs) - This setting sets the number of seconds to wait before polling the devices for process status information. The default is 180 seconds.

- Process Parallel Jobs - This setting limits the number of parallel jobs this collector is allowed to perform for Zenoss at any given time. The default is 10 jobs.

- Status Cycle Interval (secs) - This setting sets the number of seconds to wait between polling the devices for this collector for their status. The default is 60 seconds.

- Windows Service Cycle Interval (secs) - This setting controls the interval at which the collector polls Windows devices for Service status data. The default is 60 seconds.

- Windows Modeler Cycle Interval (secs) - This setting sets the number of seconds to wait in between polling Windows devices to model them. The default is a 60 second interval.

- Config Cycle Interval (mins) - This setting sets the interval for getting and sending configuration data for the collector and associated devices. The default is 360 minutes.

- Ping Time Out (secs) - This setting sets the time after which an unreturned ping packet will generate another ping try up to the number set in the Ping Tries setting. The default is 1.5 seconds.

- Ping Tries - This setting sets a number of times to attempt to get a ping packet returned from the collector. The default number of tries is 2.

- Maximum Ping Packets in Flight - This setting sets the maximum number of ping packets allowed to be either coming or going from the collector as generated by this central instance of Zenoss. The default number of packets allowed is set to 75.

- Ping Cycle Time (secs) - The number of seconds to wait before sending another ping packet to a device. The default is 60 seconds.

- Maximum Ping Failures - The default maximum ping failures is 1440

- Modeler Cycle Interval (mins) - This is the default interval for modeler to model the devices the collector is responsible for. The default is 720 mins.

- Default Discovery Networks - This where to put the networks you want this collector to discover by default. For example using the 10.1.2.0/24 namespace will give you that namespace as a basis for the other devices.

- Render URL - The render URL is the URL to provide the performance data to the viewing instance of Zenoss.

- RenderServer - The render server is the location of the server on the collector where the performance and status information is served from.

- Render User - If the render server specified above requires a user, you would see/enter the render server user name here. The default is blank.

2.3.5. Editing Other Zenoss Settings

The following settings can be changed in this tab.

- Dashboard Production State Threshold - This setting defines the production state of devices that are allowed to appear on the dashboard. The number here correlates with the State conversion setting below. Any device having a threshold below this threshold will not appear on the dashboard regardless of the event severity.

- Dashboard Priority Threshold - Use this threshold to define what severity of events are allowed to appear on the dashboard. This threshold correlates with the Priority conversions defined below. Any event having a dashboard priority that falls below this threshold will not appear on the dashboard.

- State Conversions - These settings associate a numerical value to each of the defined production states. It is not important what the numbers are as much as the relationship between the numbers. It is also used in the Dashboard Production state threshold above.

  - Production:1000

  - Pre-Production:500

  - Test:400

  - Maintenance:300

  - Decommissioned:-1

- Priority Conversions - These settings associate a numerical value to each of the defined event priorities. It is not important what the numbers are as much as the relationship between the numbers. It is also used in the Dashboard Priority threshold above.
  - Highest:5
  - High:4
  - Normal:3
  - Low:2
  - Lowest:1
  - Trivial:0

- Administrative Roles - Use the Administrative Roles file to define and name administrative roles to be associated with groups of users, they currently do not have differences in permission levels, but are good for grouping users by job function. A decent organizer tool, but it will not differentiate access levels.

- For more information about the email and paging settings found on this tab, see the "Setting SMTP and SNPP Information" section in this guide.

2.4. Hiding the Left Navigation Menu

You can use the triangle above the left navigation menu to hide the menu, and then you can use the pin icon to “pin” the menu into place and the left navigation menu will remain visible.

3. Directory Path

The directory path is where the path on the machine is displayed showing the current location of the data appearing in the UI.

4. Device/IP Search Box

You can use the Search box to find any device using its name (or partial name) or IP address.

5. User Information Area

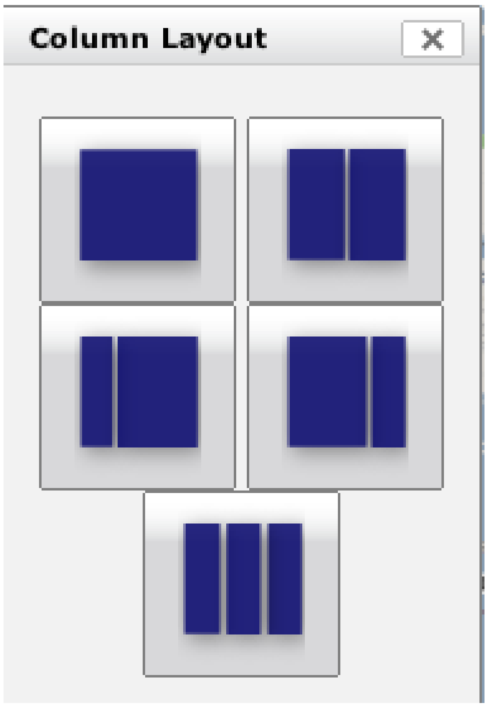
The ID of the current user is shown in the top right of the screen just to the left of the Preferences link. You can also access user setting by clicking the Settings link in the left navigation area in the link or log out using the Log Out link.

6. Customizing Zenoss Dashboard Portlets

You can customize the Zenoss Dashboard by selecting and arranging various portlets that display different information about the system. You can also arrange how the portlets appear on the screen.

To set the arrangement of the portlets on the dashboard, from the top right hand side of the Dashboard, select Configure layout. The Column Layout dialog appears.

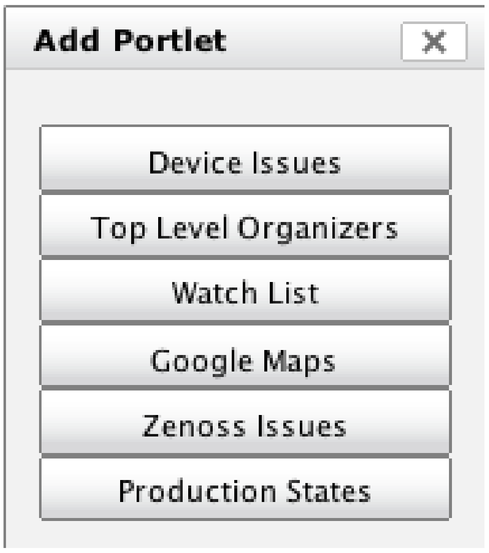
Figure 4.5. Column Layout Dialog



Select the column layout for your dashboard. Once you have selected the column layout, you can then drag and drop the portlets wherever you want on those grids.

To select which portlets you want to see on the dashboard: From the Zenoss Dashboard, in the top right, select Add Portlet. The Add Portlet dialog appears.

Figure 4.6. Add Portlet Dialog



Select the portlets you want to appear on the dashboard from the list. Available portlets include:

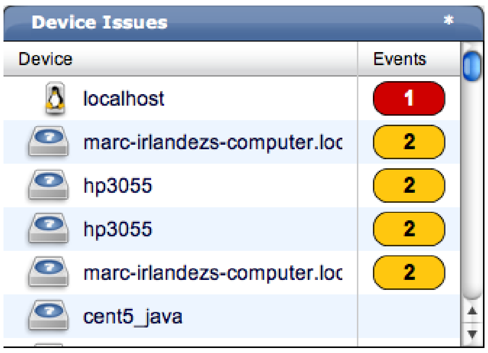
- Device issues
- Top level Organizers
- Watch List

- Google Maps
- Zenoss Issues
- Production States

6.1. Devices Issues Portlet

This section displays the name of the device, the acknowledged status, and amount of Critical and Error severity level events. “Devices” can be clicked on to take you to the Devices event log, which is default sorted by severity. Clicking on the specific device will take you to the device log of that device. The events are shown in fractions of acknowledged/total. If they are acknowledged it shows the user that acknowledged it.

Figure 4.7. Device Issues Portlet

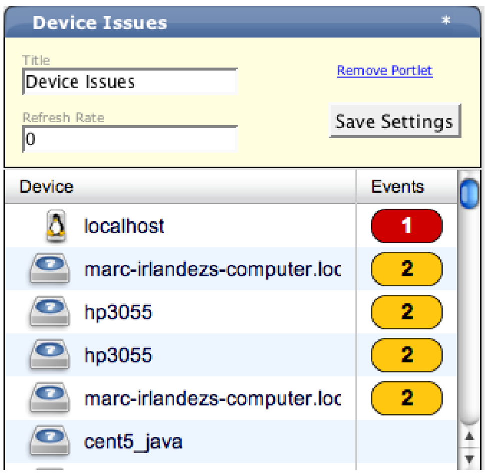

Device Issues

Device	Events
localhost	1
marc-irlandezs-computer.loc	2
hp3055	2
hp3055	2
marc-irlandezs-computer.loc	2
cent5_java	

6.1.1. Configuring the Device Issues Portlet

To configure the information that appears in the Device Issues Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate. This is also how you remove the portlet from the Dashboard.

Figure 4.8. Device Issues Portlet Configuration


Device Issues

Title: Device Issues

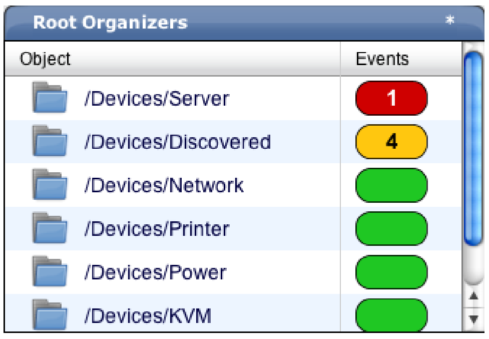
Refresh Rate: 0

Device	Events
localhost	1
marc-irlandezs-computer.loc	2
hp3055	2
hp3055	2
marc-irlandezs-computer.loc	2
cent5_java	

6.2. Top Level Organizers

The Top Level Organizers portlet shows all of the organizers at the top level of the hierarchy so you can see summary events for each organizer.

Figure 4.9. Top Level Organizer Portlet



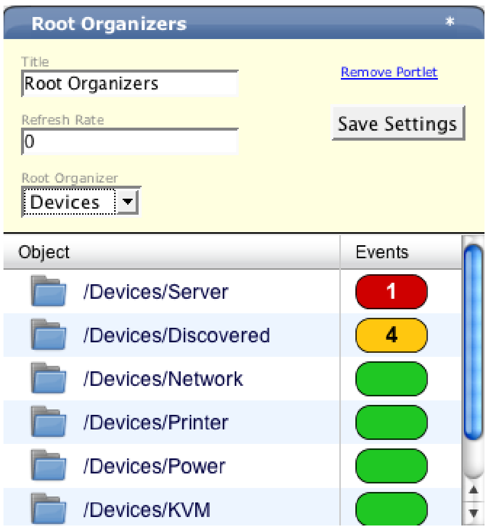
Root Organizers *

Object	Events
/Devices/Server	1
/Devices/Discovered	4
/Devices/Network	🟩
/Devices/Printer	🟩
/Devices/Power	🟩
/Devices/KVM	🟩

6.2.1. Configuring the Top Level Organizer Portlet

To configure the information that appears in the Top Level Organizer Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate, and also choose which Root Organizer you want to display. This is also how you remove the portlet from the Dashboard.

Figure 4.10. Top Level Organizer Configuration



Root Organizers *

Title: Root Organizers

Refresh Rate: 0

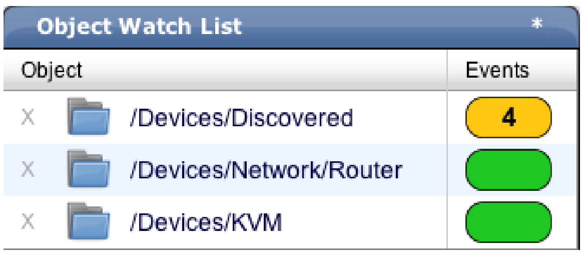
Root Organizer: Devices

Object	Events
/Devices/Server	1
/Devices/Discovered	4
/Devices/Network	🟩
/Devices/Printer	🟩
/Devices/Power	🟩
/Devices/KVM	🟩

6.3. Watch List Portlet

The Watch List Portlet shows a Device Watch List you can configure a list of watch devices an also the refresh rate for the gathering data for devices in the list. The Watch list portlet can also be used to watch individual Event Classes in the case that you have different people in the department handling different kinds of errors - and you want to separate the event-flow. Example; You map disk read errors for the symmpi driver to /VMWare, and you want your VMWare staff to watch that event class - but not necessarily watch everything going on with the guests.

Figure 4.11. Watch List Portlet



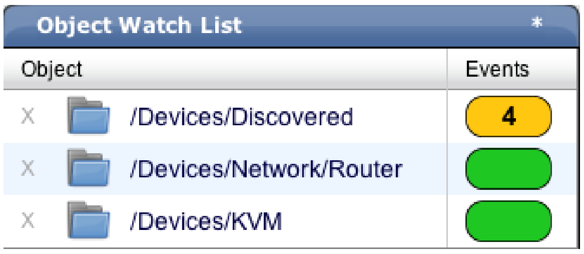
Object Watch List *

	Object	Events
X	/Devices/Discovered	4
X	/Devices/Network/Router	🟩
X	/Devices/KVM	🟩

6.3.1. Configuring the Watch List Portlet

To configure the information that appears in the Device Issues Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate. This is also how you remove the portlet from the Dashboard. You can also add or remove devices from the Watch List here.

Figure 4.12. Watch List Portlet Configuration



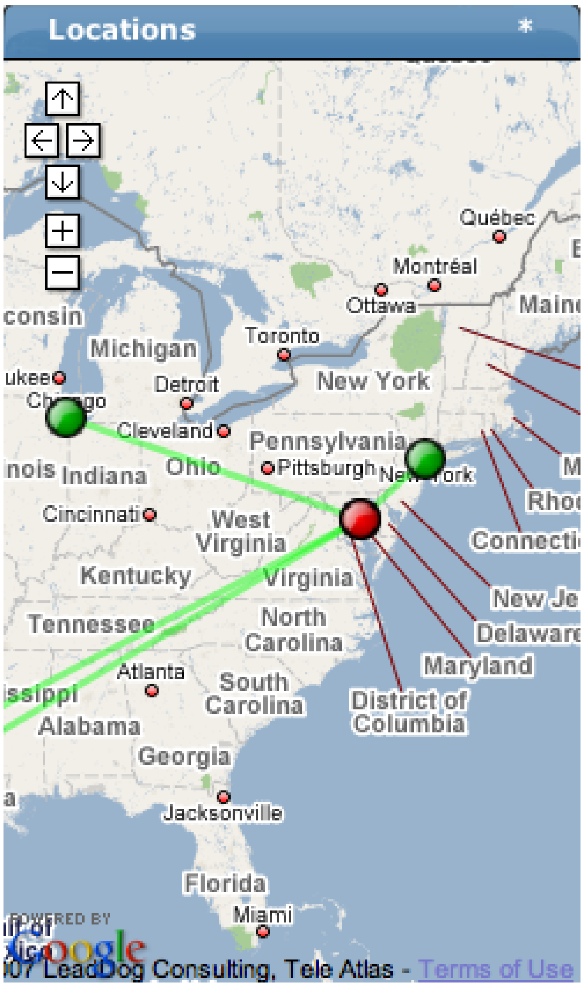
Object Watch List *

	Object	Events
X	/Devices/Discovered	4
X	/Devices/Network/Router	🟩
X	/Devices/KVM	🟩

6.4. Google Maps Portlet

The Google Maps Portlet allows you to see the Location map with Location you have set up and network connection statuses. Here is a look at the Google Maps portlet.

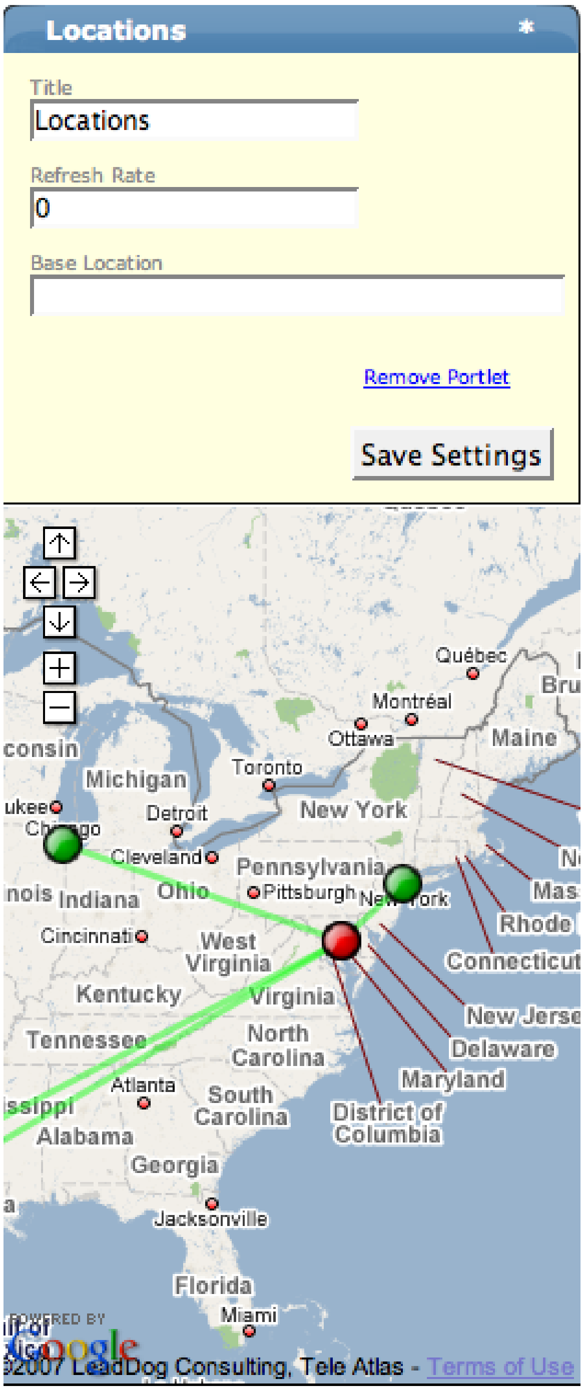
Figure 4.13. Google Maps Portlet



6.4.1. Configuring the Google Maps Portlet

To configure the information that appears in the Device Issues Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate. This is also how you remove the portlet from the Dashboard. You can also set the Base Location here.

Figure 4.14. Configuring the Google Maps Portlet


Locations

Title: Locations

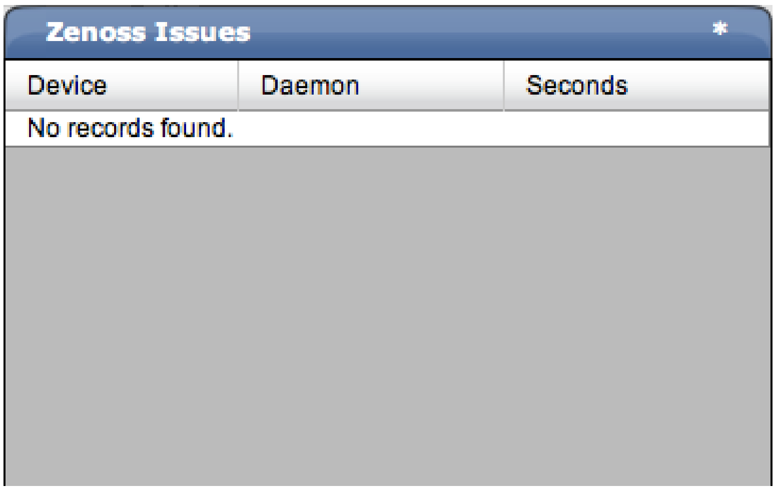
Refresh Rate: 0

Base Location:

6.5. Zenoss Issues Portlet

Zenoss Infrastructure Issues will show any problems with a Zenoss daemon. Heartbeats statuses of the Zenoss daemons appear here when there is a problem. If an issue appears in this list, it could be that the daemon is not running.

Figure 4.15. Zenoss Issues Portlet

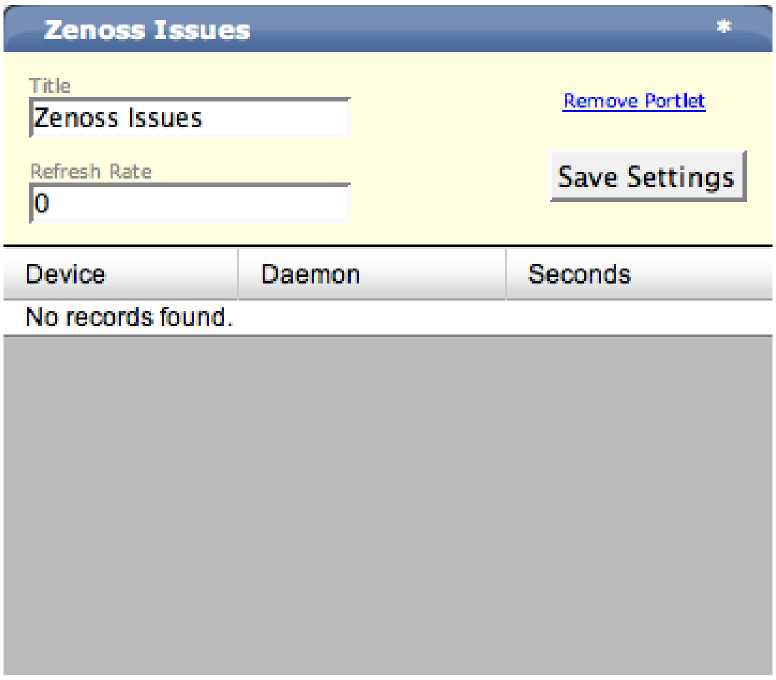


Device	Daemon	Seconds
No records found.		

6.5.1. Configuring the Zenoss Issues Portlet

To configure the information that appears in the Device Issues Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate. This is also how you remove the portlet from the Dashboard.

Figure 4.16. Configuring the Zenoss Issue Portlet



Title: Zenoss Issues

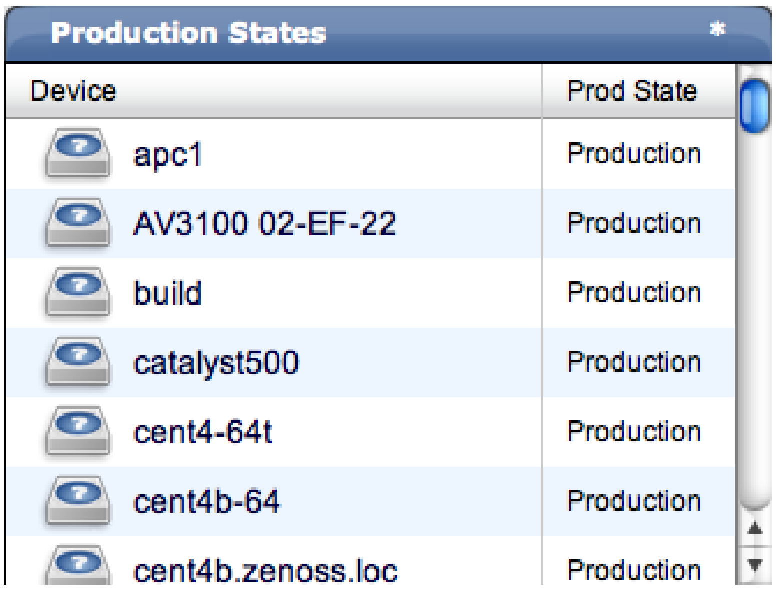
Refresh Rate: 0

Device	Daemon	Seconds
No records found.		

6.6. Production States Portlet

The Production States Portal allows you to see all of the devices in a given Production State.

Figure 4.17. Production State Portlet



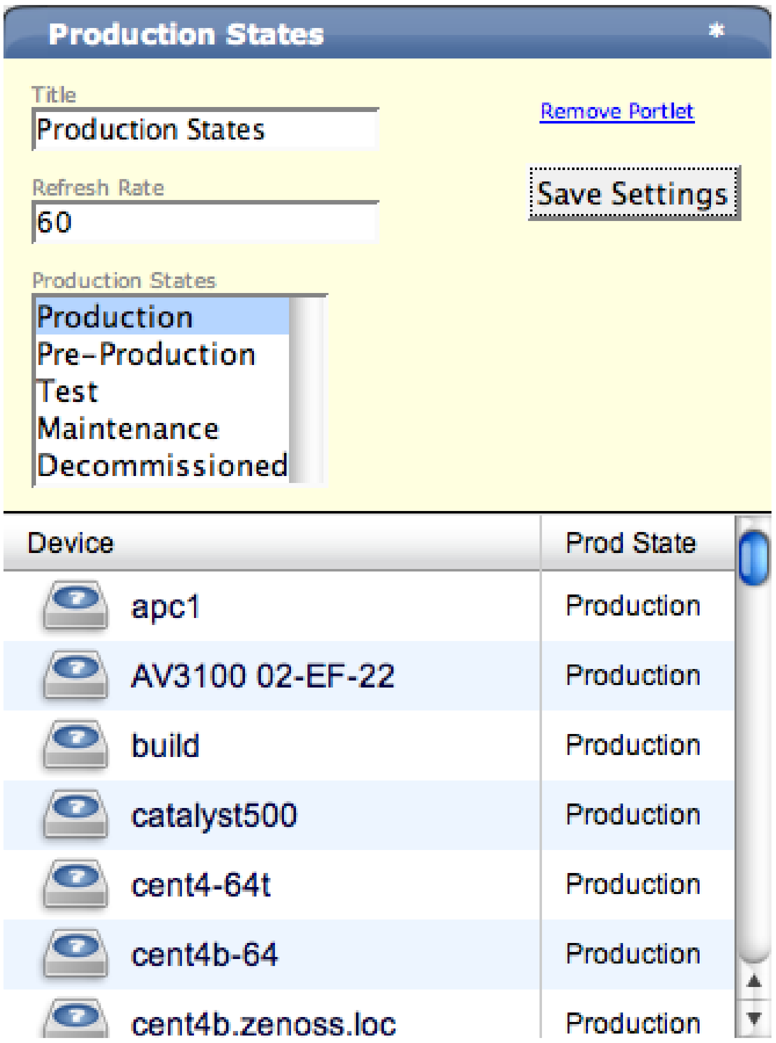
Production States *

Device	Prod State
apc1	Production
AV3100 02-EF-22	Production
build	Production
catalyst500	Production
cent4-64t	Production
cent4b-64	Production
cent4b.zenoss.loc	Production

6.6.1. Configuring the Production States Portlet

To configure the information that appears in the Device Issues Portlet click the asterisk in the top right of the portlet window. A drop down configuration dialog appears where you can change the information for this portlet. You can change the Title of the portlet or the Refresh Rate. This is also how you remove the portlet from the Dashboard. You can also choose what Production States you want to appear on the Dashboard here.

Figure 4.18. Configuring the Production State Portlet


Production States

Title: Production States

Refresh Rate: 60

Production States: Production, Pre-Production, Test, Maintenance, Decommissioned

Device	Prod State
apc1	Production
AV3100 02-EF-22	Production
build	Production
catalyst500	Production
cent4-64t	Production
cent4b-64	Production
cent4b.zenoss.loc	Production

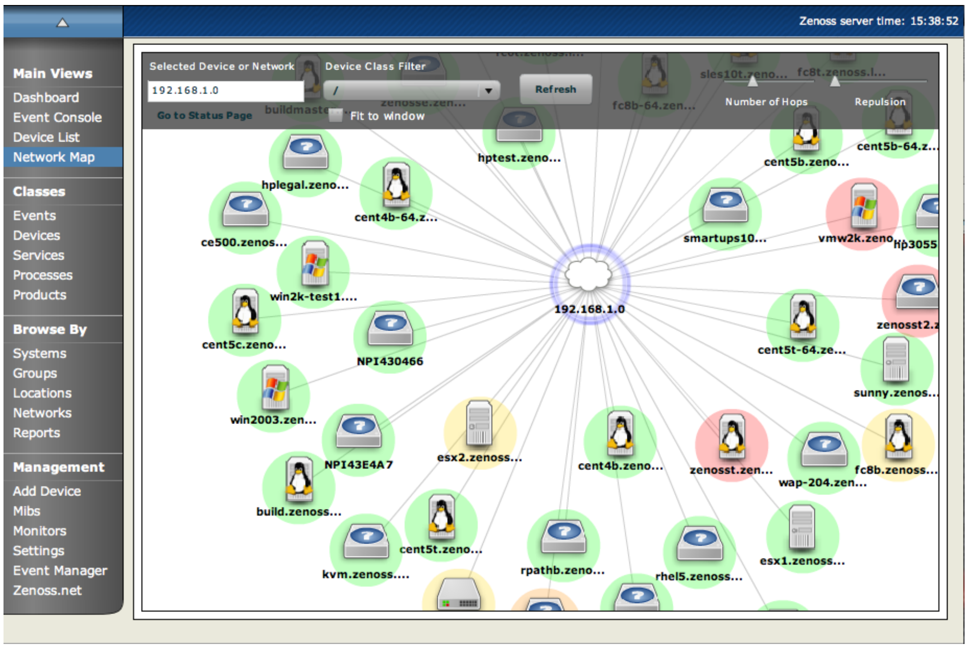
7. Zenoss Network Map

The Network Map is a Flash representation of your network's layer 3 topology. You can see the status of the devices on your network by noting the background color of the devices. You can select a particular device or network by double clicking on its icon in the map. This will center the map on the device or network and show the links from this node based on the number of hops selected. The same result is achieved by typing the name of the device or network and clicking the "Refresh" button. Note that when you select a node what is actually displayed are the links that are currently loaded into the map it does not download new link data at that time.

To load/reload the link data for a particular node, double click on that node, select the number of hops and click the refresh button. You can also filter the devices shown by Device Class. For example, to show only Linux devices, select /Server/Linux in the Device Class Filter and click refresh.

You can also adjust the number of viewable hops from the selected device by using the Number of Hops slider in the top right corner. You can also spread out the icons on the map by using the Repulsion slider also in the top right corner. If you'd like to get more detail on the selected device or network, just click the Go to Status Page link in the top left corner.

Figure 4.19. Zenoss Network Map



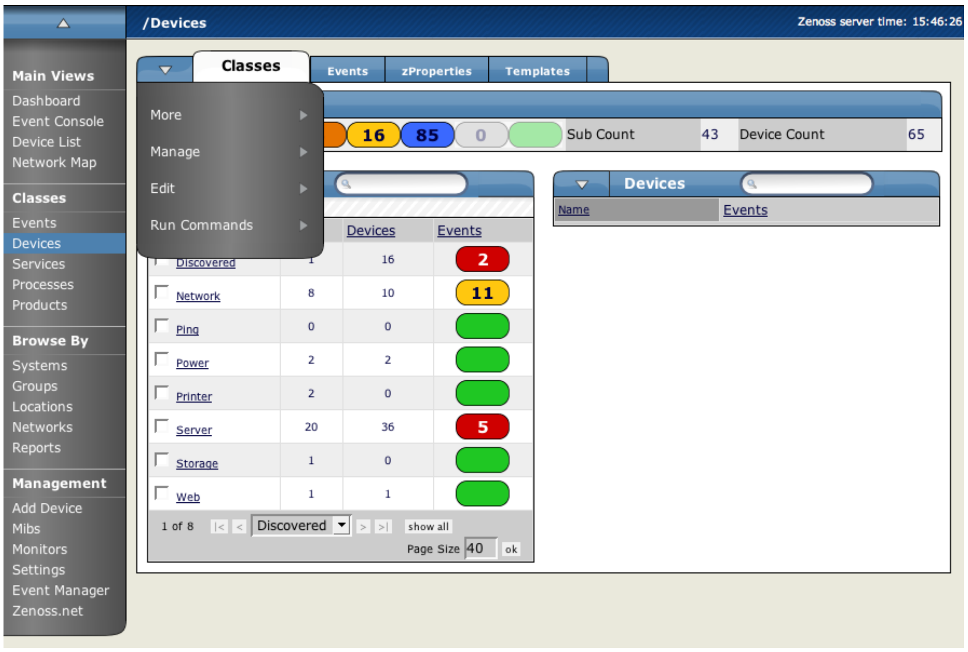
8. "Menu-ized" Elements

Zenoss now has two different kinds of menus where some elements that previously resided on pages are now located.

8.1. Page Menus

Page menus extend the tabs near the top of the page and usually affect the entire object that the page represents. This could be a device, event or any grouping of these elements. In the following figure you can see the Page menu expanded next to the Classes tab.

Figure 4.20. Page Menu

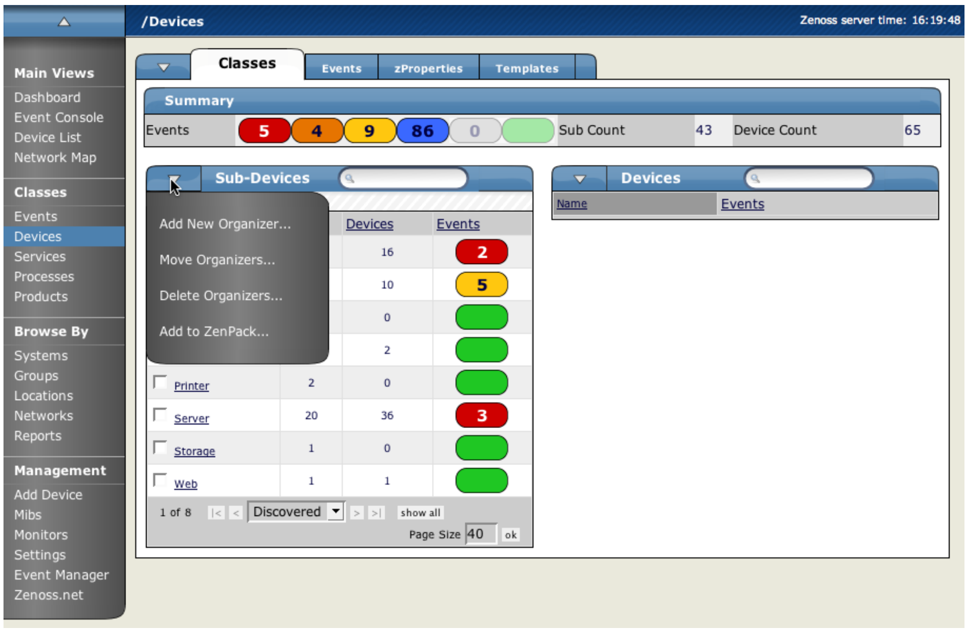


		Devices	Events
Discovered	1	16	2
Network	8	10	11
Ping	0	0	🟩
Power	2	2	🟩
Printer	2	0	🟩
Server	20	36	5
Storage	1	0	🟩
Web	1	1	🟩

8.2. Table Menus

Table menus are accessed by clicking the triangle next to a table title inside a particular page. The following figure, which is the same page as above, shows the Table menu expanded. In this case it is the Sub-Devices table menu.

Figure 4.21. Sub-Devices Table Menu



Events	5	4	9	86	0		Sub Count	43	Device Count	65

		Devices	Events
		16	2
		10	5
		0	
		2	
Printer	2	0	
Server	20	36	3
Storage	1	0	
Web	1	1	

Chapter 5. User Management

1. About Zenoss User Accounts

Each user should have a unique User ID. These User IDs group permissions and alerting rules together as well as providing security to the Zenoss system. To create and manage user accounts in Zenoss, you must be logged into the Zenoss Admin account, or logged in to a created user account with manage privileges. Custom Event Views and Alerting Rules are also defined per user. Those items are discussed in other areas of this documentation.

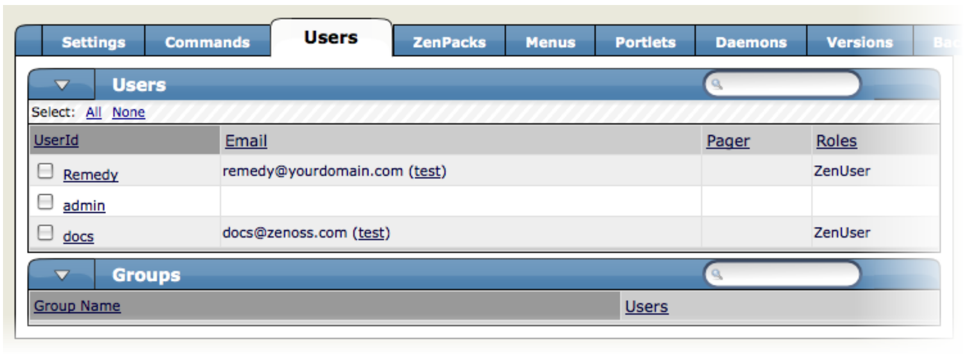
2. Creating New User Accounts

To create a new user account:

1. From the menu on the left side of the Zenoss Dashboard, select Zenoss Settings.

2. Click the Users Tab. The Users Administration page appears.

   Figure 5.1. User Administration page

UserId	Email	Pager	Roles
Remedy	remedy@yourdomain.com (test)		ZenUser
admin			
docs	docs@zenoss.com (test)		ZenUser

Group Name	Users

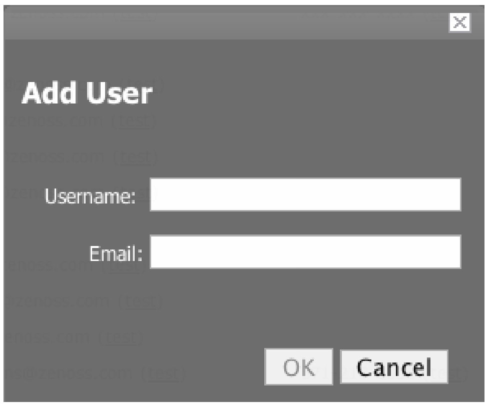
3. Click the User Folder table menu to show the User Options.

   Available options will be Add User, Delete User and Add to ZenPack

4. Select Add User.

   The Add User dialog appears.

   Figure 5.2. Add User Dialog

5. Enter a name for the account in the Username field.

6. Enter an Email address for the user account.

   This is the address where any alerts you set up for this user will be sent.

7. Click the OK button.

   The user appears in the User List.

   You have now created a new user account. You must still edit the User account to provide a password and additional User details. See the Editing Users section in the Users Guide section for information on setting User preferences.

3. Editing User Accounts

1. From the menu on the left side of the Zenoss Dashboard, select Settings.

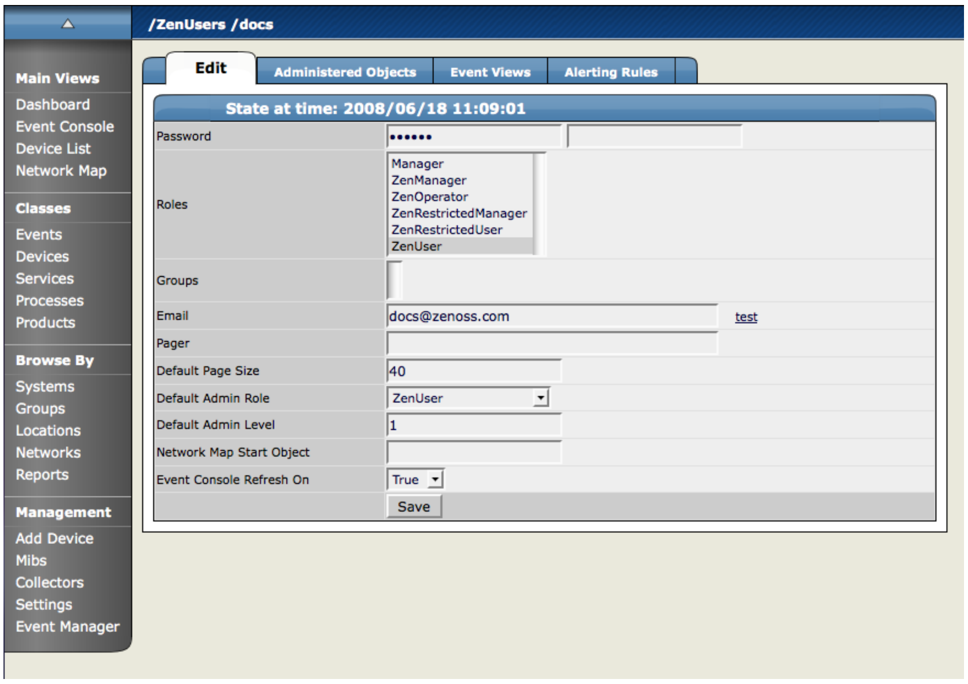
2. Click the Users Tab.

   The Users Administration page appears.

3. Click the name of the user account you want to edit.

   The individual User Administration page for the newly created user appears.

   Figure 5.3. Individual User Administration Page

   State at time: 2008/06/18 11:09:01

Password	••••••
Roles	Manager, ZenManager, ZenOperator, ZenRestrictedManager, ZenRestrictedUser, ZenUser
Groups	
Email	docs@zenoss.com test
Pager	
Default Page Size	40
Default Admin Role	ZenUser
Default Admin Level	1
Network Map Start Object	
Event Console Refresh On	True

4. Make the changes as detailed in the sub sections below and click the Save button to keep the changes.

3.1. Setting User Passwords

From the Individual User Administration page, assign a password for the user. Enter it again to confirm it in the next text box.

3.2. Editing User Contact Information

From the Individual User Administration page, you can enter and edit the email address and/or pager number.

3.3. Assigning Roles and Permissions to Users

From the Individual User Administration page, you can assign specific roles for each user privileges for this User. These roles are roles available to be set by the one Zenoss Admin user. If you have manger role set, then you can change settings for your own account or for any other accounts as well (including other manager accounts).

Table 5.1.

Role	Privileges
Manager	The Manager role has full privileges. This role has viewing and editing privileges on objects in the system. Managers also see the Management area of the Navigation menu on the left side of the web console. These options include: Adding Devices, User management for all users (including changing roles), Monitors, Live Event viewing, History Events and also the About information.
ZenUser	ZenUser has viewing information of all the information in the system., Users with the role ZenUser can also edit their own user information (except for changing their Role). ZenUsers cannot add devices and do not see the Management section of the Navigation menu on the left side of the web console.

3.4. Setting the Default User Level

The Default User Admin Level sets the default role for administered objects. This default is easily changed, but you should choose the role here that is most commonly added. Choose between the following roles to set as the default.

- Administrator
- Engineer
- Analyst
- Tester

These User admin levels are currently merely ways to categorize and group users. In the future there may be more attributes that would be assignable per group/role. For now it is just a way to categorize and classify users.

3.5. Setting the Dashboard Timeout and Refresh Time

You can change the Dashboard Settings such as timeout and Refresh times on a per user basis. In the Dashboard Refresh and Dashboard Timeout fields, enter the number of seconds before a timeout message appears for the Zenoss Dashboard.

3.6. Specifying a Default Dashboard Organizer

The default Dashboard organizer is what appears in the Devices Summary on the Dashboard for this particular user. You can change your default dashboard view to display the information as either Devices, Systems, Groups, or Locations.

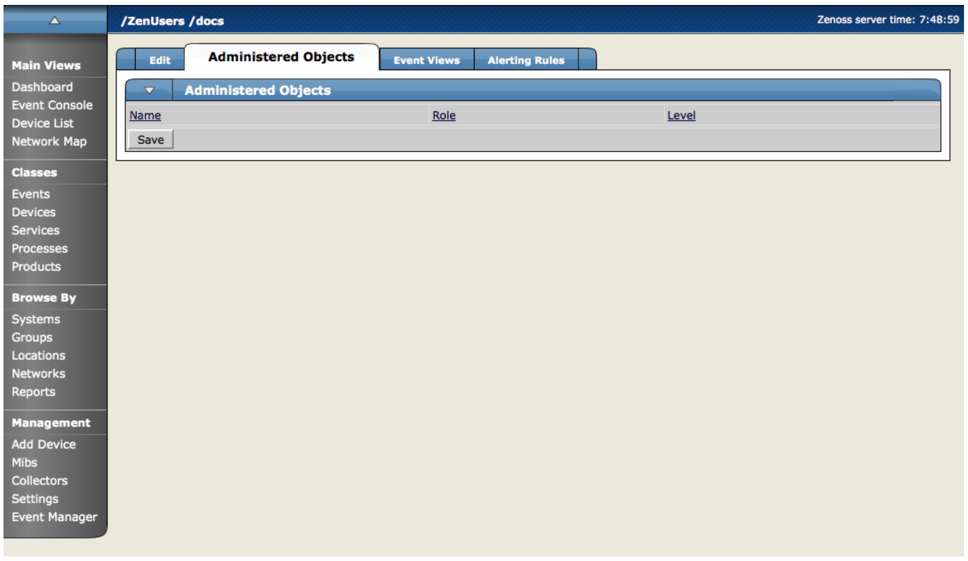
3.7. Associating Objects in Zenoss with Specific Users

You can associate any object in the Zenoss system with a particular user (for monitoring or reporting purposes). To create these associations:

1. From the Individual User Administration page, click the Administered Objects tab.

   The Administered Object Tab appears.

   Figure 5.4. Administered Objects Initial View

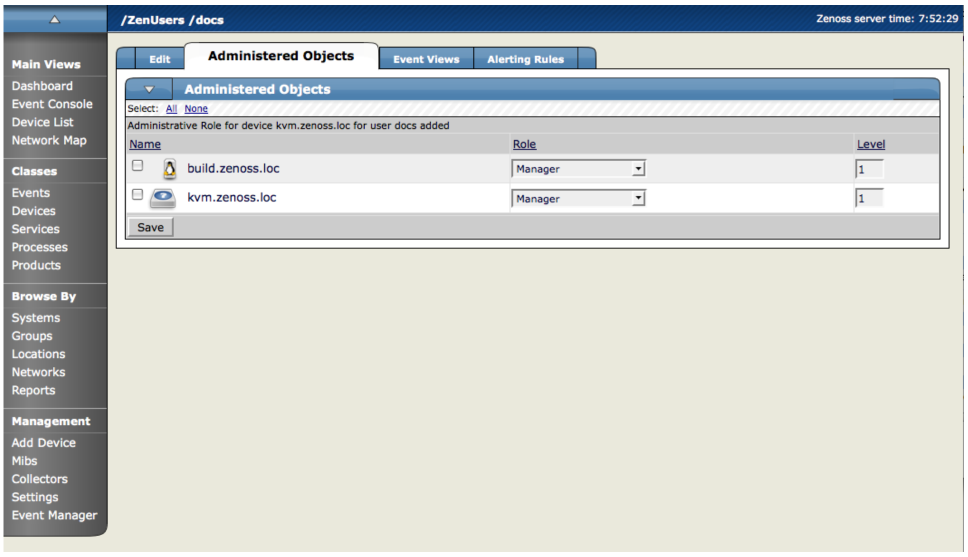
2. Select an object type from the Administered Devices table menu. You can choose between adding a Device, System, Group, or Location.

   A dialog will appears where you can specify the component you want to add as an administered object.

3. Click the OK Button.

   The Device appears in the Administered Devices list for this user.

Figure 5.5. Administered Object - Object Added



Administered Objects

Select: All None

Administrative Role for device kvm.zenoss.loc for user docs added

Name	Role	Level
build.zenoss.loc	Manager	1
kvm.zenoss.loc	Manager	1

4. You can now change the role that is associated for this user on this object.

   The default role is set in the Individual User Administration Page.

   You can also set administered objects from the device you want to add as an administered object. For example, navigate to the device you want to add to this user's administered objects page. Open the page menu for this device, select More and then the Administration option. Open the Administrator's table menu choose Add Administrator and select the user you want to add as an administrator. The object is added to the user's administered object list, and the name appears in this object's Administrator's List.

4. User Groups

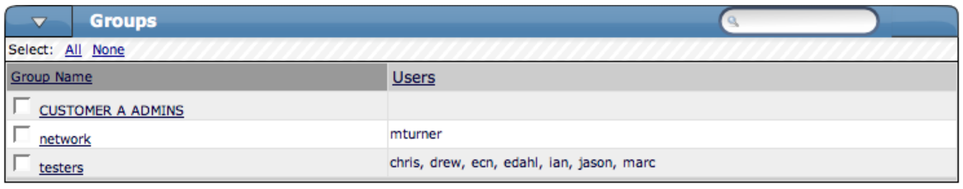
You can create Zenoss User groups to aggregate rules and apply them across multiple user accounts. To create a new User Group:

1. From the Left Navigation menu, select Settings.

2. Click the Users tab.

   The Users tab appears.

Figure 5.6. Users Tab showing User Groups



Groups

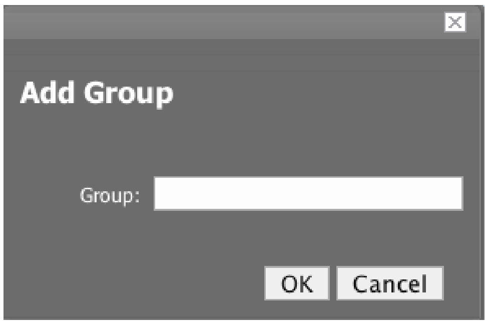
Select: All None

Group Name	Users
CUSTOMER A ADMINS	
network	mturner
testers	chris, drew, ecn, edahl, ian, jason, marc

3. From the Groups table menu, select Add New Group.

   The Add New Group dialog appears.

Figure 5.7. Add New User Group Dialog



4. In the Group field, enter a name for this user group.

5. Click the OK button.

   The name of the group you entered appears in the Groups list.

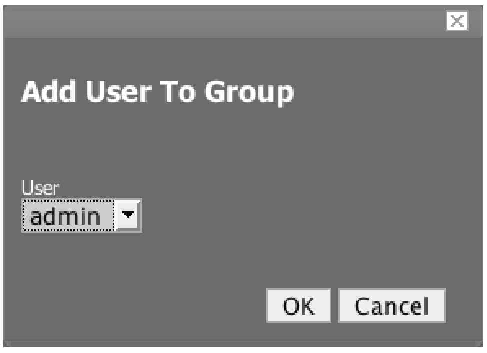
6. Click the name of the Group you just created.

   The Edit tab for this group appears.

7. From the Users In Group Table menu, select Add User.

   The Add User to Group dialog appears.

Figure 5.8. Add User to Group Dialog



8. From the User drop-down, select the users you want to add to the group.

9. Click the OK button.

   The user(s) you have selected appear in the list of users for this group.

You can also choose administered objects and Alerting Rules for this User group. These Alerting Rules will apply to all users in the group. The User's original Alerting Rules and objects will also apply.

Chapter 6. Device Access Control Lists (ACLs) (Commercial)

1. About Device ACLs in Zenoss

The Device ACL Enterprise ZenPack (ZenDeviceACL) adds fine-grained security controls to Zenoss. For example, this control can be used to give limited access to certain departments within a large organization or limit a customer of a service provider to only see their own data. A user with limited access to objects also has a more limited view to screens within the system. As an example, most global views, such as the network map, event console, and all types of class management are not available. The Device List is available as are the device organizers; Systems, Groups, and Locations. A limited set of reports can also be accessed.

2. Key Elements

2.1. Permissions and Roles

Actions within Zenoss are assigned permissions. For instance to access the device edit screen you must have the “Change Device” permission. Permissions not assigned directly to a user since this would be difficult to managed. Instead permissions are granted to Roles, which are then assigned to a user. A common example is the ZenUser role within Zenoss Core. Its primary permission is “View” which grants read-only access to all objects. ZenManagers have additional permissions such as “Change Device” which grants them access to the device edit screen. The Device ACL ZenPack has two the new role, ZenRestrictedManager, which allows a more limited set of device edit functions. In Zenoss Core when you assign a role to a user using the Roles field on the Edit tab it is “global”. When creating a restricted user you may not want to give that user any global role.

2.2. Administered Objects

Device ACLs provide limited control to various objects within the system. Administered objects are the same as the device organizers: Groups, Systems, and Locations and Devices. If access is granted to any device organizer, it flows down to all devices within that organizer. To assign access to objects for a restricted user, you must have the Manager or ZenManager roles. Zenoss grants access to objects is granted using the “Administered Objects” tab of a user or user group. To limit access, you must NOT assign a “global” role to the user or group.

2.3. Users and Groups

Users and user groups work exactly as they would normally. See the section in the User Management section of this guide dealing with users and groups.

2.4. Assigning Administered Object Access

On each user or group there is a tab called “Administered Objects”. The menu has an add item for each type of administered object. Adding an object will pull up a dialog box with live search on the given type of object. After an object has been added you can assign it a role. Roles can be different for each object so a user or group might have ZenUser on a particular device but ZenManger on a location organizer. If multiple roles are granted to a device though direct assignment and organizer assignment the resulting permissions will be additive. In our example above if the device was within the organizer the user would end up with the ZenManager role on the device.

2.5. Portlet Access Control

Within Zenoss Core portlet access can be controlled this is important for Device ACLs see the Zenoss Admin Guide for more details.

3. Setup and Configuration Examples

3.1. Restricted User with ZenUser Role

1. As admin or any user account with Manager or ZenManager role create a user named acltest. Don't forget to set its password.

2. From the user's Edit tab, make sure that no role is assigned.

3. Now go to the user's "Administered Objects" tab.

4. On the menu select the "Add Device…" item and add an existing device to that user.

   The device's role will default to ZenUser, leave it this way for now.

5. Log out of your browser, or open a second browser and login as acltest.

6. Click on the "Device List".

   You should see only the device you assigned to acltest.

7. Navigate to the device and notice that you don't have the "Edit" tab available. This is because you are in read-only mode for this device.

3.2. Restricted User with ZenManager Role

Following the example above:

1. Change the acltest user's role to "ZenManager" on the device.

   You will need to do this as a user with ZenManager global rights. Log out and back in if necessary.

2. Go back to the acltest user "Administered Objects" tab and set the role on your device to ZenManager.

3. As acltest, navigate back to your device and notice that you now have access to the Edit tab. The device menu will have many more items as well

3.3. Adding Device Organizers

1. Go to the Groups root and create a group called "RestrictGroup".

2. Go back to the acltest user's Administered Objects tab and add the group to the user.

3. Now logged in as acltest notice that the left nav has the Groups item. Group can be added to a user. If more are added then the first will be used.

4. Now place a device within this group and as acltest you should not only see the device within the group but also in the device list

3.4. Restricted User Organizer Management

1. Now give the acltest user ZenManager on your RestrictedGroup.

2. As acltest you can now add sub-organizers under the RestrictedGroup.

3.5. Viewing Events

A user in restricted mode does not have access to the global event console. The available events for the user can be seen under his organizers.

4. Detailed Restricted Screen Functionality

4.1. Dashboard

By default the dashboard is configured with only three portlets: Object Watch List, Device Issues, and Production State. These have content that will be restricted to objects for a given user.

4.2. Device List

The device list will automatically be filtered to devices of a restricted user scoped to accessible devices. There are no menu items available.

4.3. Device Organizers

Device organizers are a good way to control groups of devices for a restricted user. Remember that every device added to the group will be accessible to the user. Permissions will be inherited down multiple tiers of a device organizer.

4.4. Reporting

Reports are limited to device reports and performance reports.

Chapter 7. Email and Pager Settings

1. About Email and Pager Settings

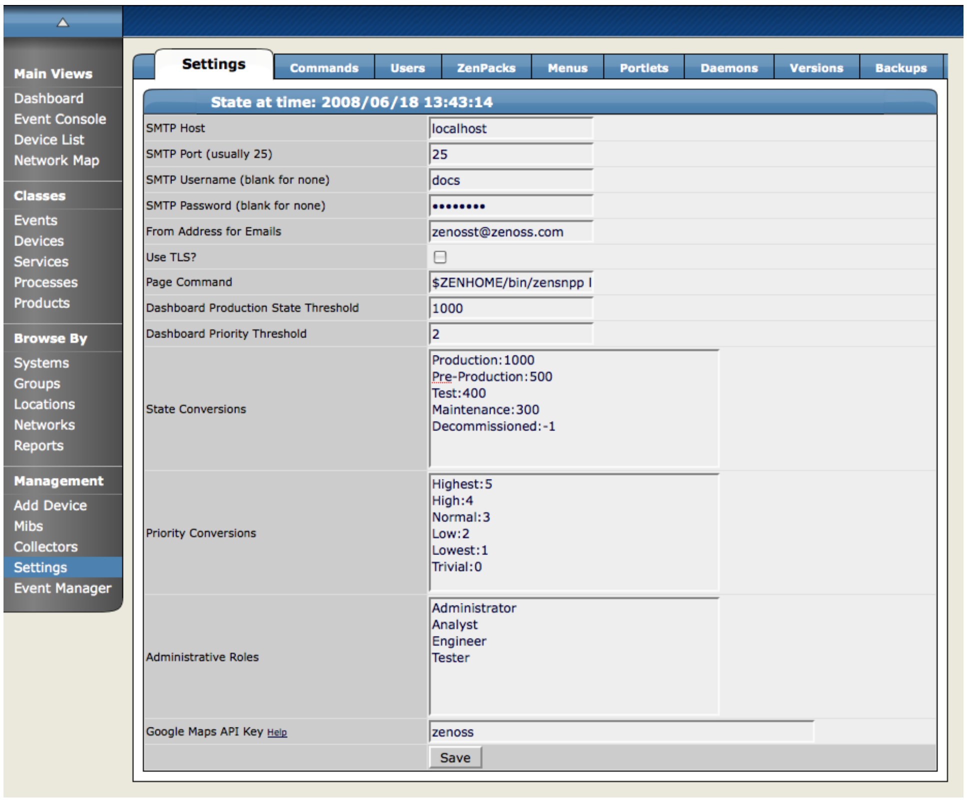
Zenoss alerts can be sent to users via email (SMTP) or pager (SNPP.) Many operating systems include an SMTP server (such as Sendmail or Postfix) with their distributions. If your OS does not include a mail server you will need to install one or specify a separate SMTP server in the settings. Many pagers can accept messages via email, but Zenoss also provides the option of sending pages via SNPP if you specify an SNPP server in the settings.

2. Setting SMTP and SNPP Information

To edit Zenoss's SMTP and SNPP settings make sure you are logged into a Zenoss manager account and click on Settings in the left navigation menu.

1. While logged in to a user account with management privileges, from the left navigation menu, click Settings. The Settings Tab appears.

Figure 7.1. Zenoss Settings- Settings Tab



State at time: 2008/06/18 13:43:14

SMTP Host	localhost
SMTP Port (usually 25)	25
SMTP Username (blank for none)	docs
SMTP Password (blank for none)	••••••••
From Address for Emails	zenosst@zenoss.com
Use TLS?	☐
Page Command	$ZENHOME/bin/zensnpp l
Dashboard Production State Threshold	1000
Dashboard Priority Threshold	2
State Conversions	Production:1000 Pre-Production:500 Test:400 Maintenance:300 Decommissioned:-1
Priority Conversions	Highest:5 High:4 Normal:3 Low:2 Lowest:1 Trivial:0
Administrative Roles	Administrator Analyst Engineer Tester
Google Maps API Key Help	zenoss

2. Change the following SMTP Settings as necessary:

Table 7.1.

Field	Description
SMTP Host	Set the SMTP Host to your corporate email server. If you were a Zenoss employee for example your corporate email server would be mail.zenoss.com.
SMTP Port	Usually Port 25.
SMTP Username	leave this field blank for none
SMTP Password	leave this field blank for none
From Address for Emails	Use if you want the emails to come from a specific email address.
Use TLS?	Check this box if you use transport layer security for your email alerts.

3. Enter a Page Command as necessary if you are using Zenoss to send pages. The pageCommand variable enables Zenoss to executes the pageCommand when a page is sent, and writes the message to the stdin of the subshell. The command prints any error messages to stdout. This enables a wider ranging of paging customization.

   The page Command that is standard in zenoss is:

   ```
   $ZENHOME/bin/zensnpp localhost 444 $RECIPIENT
   ```

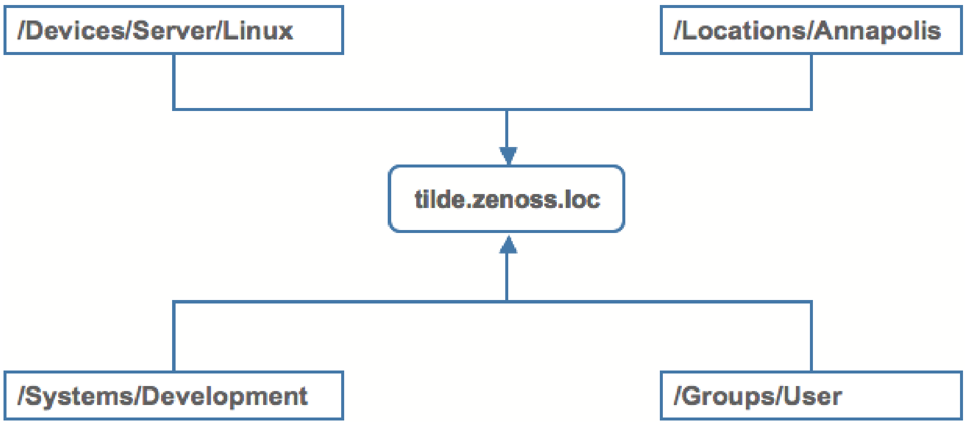
   This uses ZENHOME to send the page from the localhost, but you can use any page command customizations you want. In this case $RECIPIENT is actually the paging address for the user as set in the setting for each user.

Chapter 8. Organizers and Path Navigation in Zenoss

1. About Organizers and Path Navigation

Zenoss allows you to group any objects in the system (devices, subsystems, zProperties, templates, etc.) The groupings can be created in any way that you desire or define. There are also device classes that are tied to events coming into the system and inheritance works the same for any of these groupings. So each device can belong to several different groupings including groups, systems, locations and device classes. In the example below, you can see the device tilde.zenoss.loc belongs to 5 different classifications. Any zProperties and monitoring settings for each of these groups are now applied to to tilde.zenoss.loc.

Figure 8.1. Device Groupings



Zenoss uses several organizers to classify and organize devices in the system.

- Class - the most important organizer - templates and zProperties are inherited through this hierarchy
- Systems
- Groups
- Locations

2. Classes

The most important organizer within Zenoss is Class. There are Device Classes, Event Classes, Service Classes, and Product Classes. Templates and zProperties are can both be inherited based on Class. These attributes can be overwritten further down the class hierarchy all the way down to the individual component level. The Class hierarchy includes all defined and standard Classes and Sub-classes.

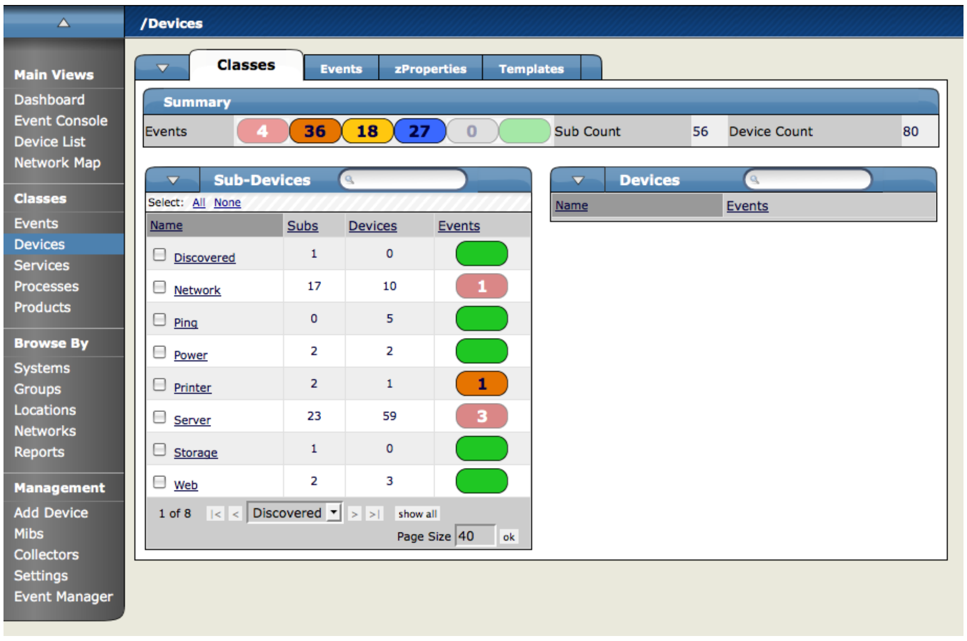
The following examples use Device Classes and Sub-classes, but the same concepts apply to Event Classes, Service Classes, and Product Classes. When you add a new device to Zenoss, you should (after providing the network name or IP address), at a minimum, specify its device class. Templates and zProperties can be set at any level in the Device Class hierarchy.

To see all of the device in Device Class:

1. From the left navigation menu, select Devices.

The Device Classes tab appears. Notice the navigation path above the tab has changed to '/Devices'. This is how you know your position within the class hierarchy.

Figure 8.2. Device Class Tab



Events	4	36	18	27	0	🟩	Sub Count	56	Device Count	80

Name	Subs	Devices	Events
Discovered	1	0	🟩
Network	17	10	1
Ping	0	5	🟩
Power	2	2	🟩
Printer	2	1	1
Server	23	59	3
Storage	1	0	🟩
Web	2	3	🟩

The Device Class tab shows an event rainbow for that class level and a Summary for the next level of class hierarchy with an indicator of whether or not there any devices in any of the classes that have events associated with them.

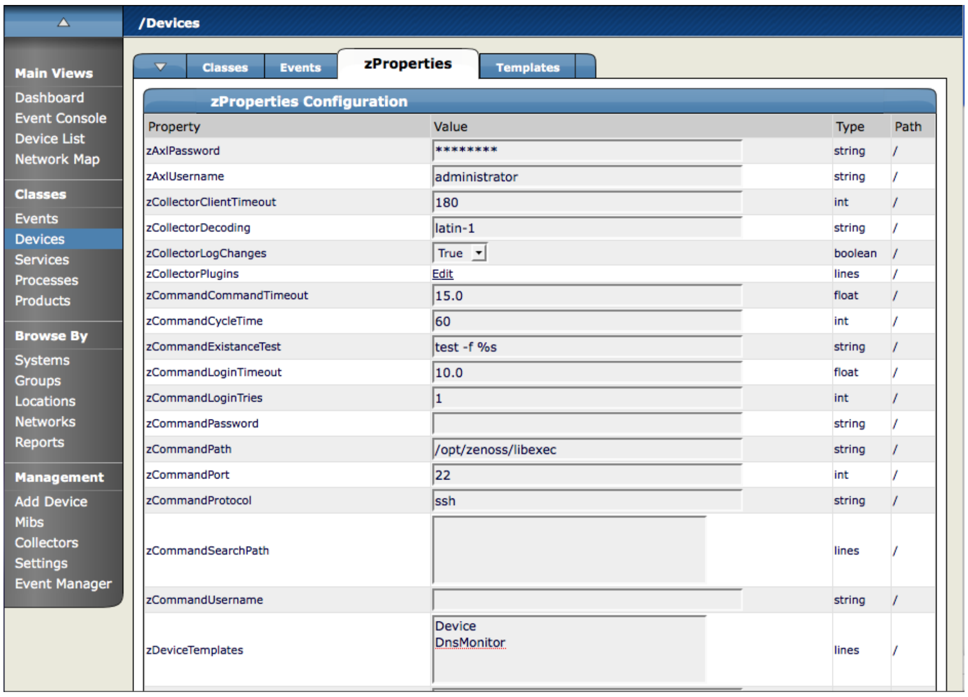
2.1. Setting zProperties at the Class Level

To set zProperties at the Device Class Level:

1. Navigate to the Device Class tab for the class you where you want to set zProperties, and click the zProperties Tab.

   The zProperties tab for this Device Class appears.

Figure 8.3. Device Class zProperties tab



zProperties Configuration

Property	Value	Type	Path
zAxlPassword	********	string	/
zAxlUsername	administrator	string	/
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		string	/
zCommandPath	/opt/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device DnsMonitor	lines	/

2. Now you can define any of the normal Device zProperties and these will apply to all devices in this class or added to this class unless over-ridden at a lower level in the hierarchy.

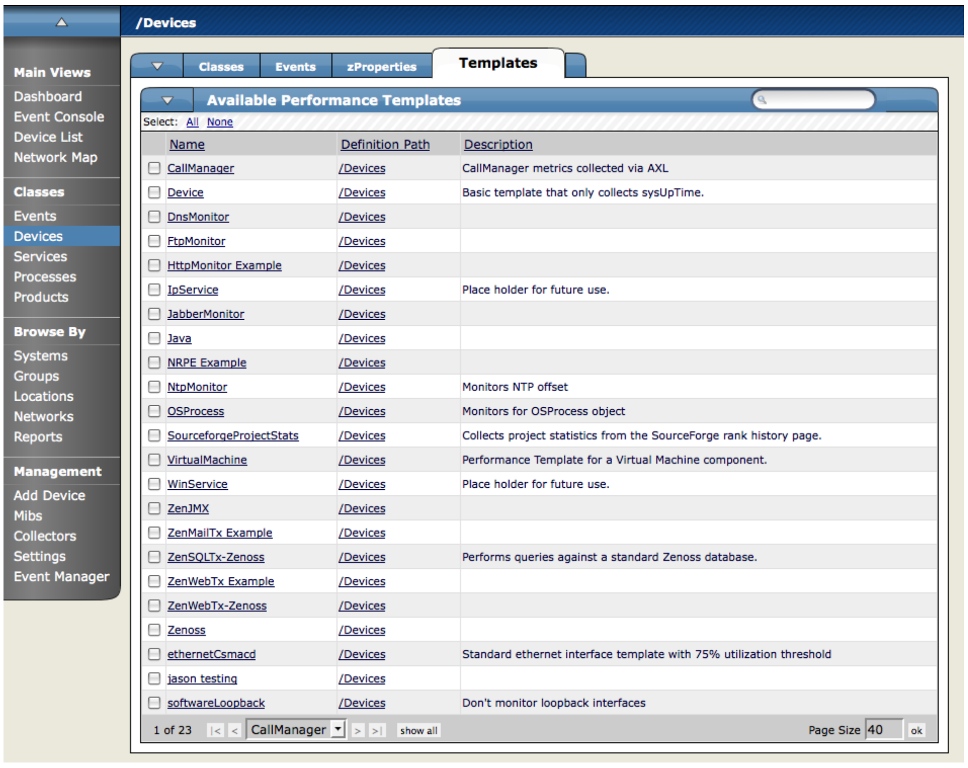
2.2. Defining and Applying Templates at the Class Level

To define Templates at the Device Class Level:

1. Navigate to the Device Class tab for the class you where you want to set Templates, and click the Templates Tab.

   The Templates tab for this Device Class appears.

Figure 8.4. Device Class Template tab



Name	Definition Path	Description
CallManager	/Devices	CallManager metrics collected via AXL
Device	/Devices	Basic template that only collects sysUpTime.
DnsMonitor	/Devices	
FtpMonitor	/Devices	
HttpMonitor Example	/Devices	
IpService	/Devices	Place holder for future use.
JabberMonitor	/Devices	
Java	/Devices	
NRPE Example	/Devices	
NtpMonitor	/Devices	Monitors NTP offset
OSProcess	/Devices	Monitors for OSProcess object
SourceforgeProjectStats	/Devices	Collects project statistics from the SourceForge rank history page.
VirtualMachine	/Devices	Performance Template for a Virtual Machine component.
WinService	/Devices	Place holder for future use.
ZenJMX	/Devices	
ZenMailTx Example	/Devices	
ZenSQLTx-Zenoss	/Devices	Performs queries against a standard Zenoss database.
ZenWebTx Example	/Devices	
ZenWebTx-Zenoss	/Devices	
Zenoss	/Devices	
ethernetCsmacd	/Devices	Standard ethernet interface template with 75% utilization threshold
jason testing	/Devices	
softwareLoopback	/Devices	Don't monitor loopback interfaces

2. Now you can define or bind any Device Template and they will apply to all devices in the class or added to this class unless over-ridden at a lower level in the hierarchy.

2.3. Creating New Classes

To create a new Device Class:

From the Class level where you want to add the organizer, navigate to the Classes tab.

From the SubClasses Table menu, select Add New Organizer and then enter a name for the organizer.

To add devices to this device class you can navigate to the Device List, select the Devices you want to add to the class and from the table menu, select Move to Class and choose which class you want to move the devices to.

3. Systems

Systems are intended to follow virtual setups like you would have in a network setup or systems grouped by functionality.

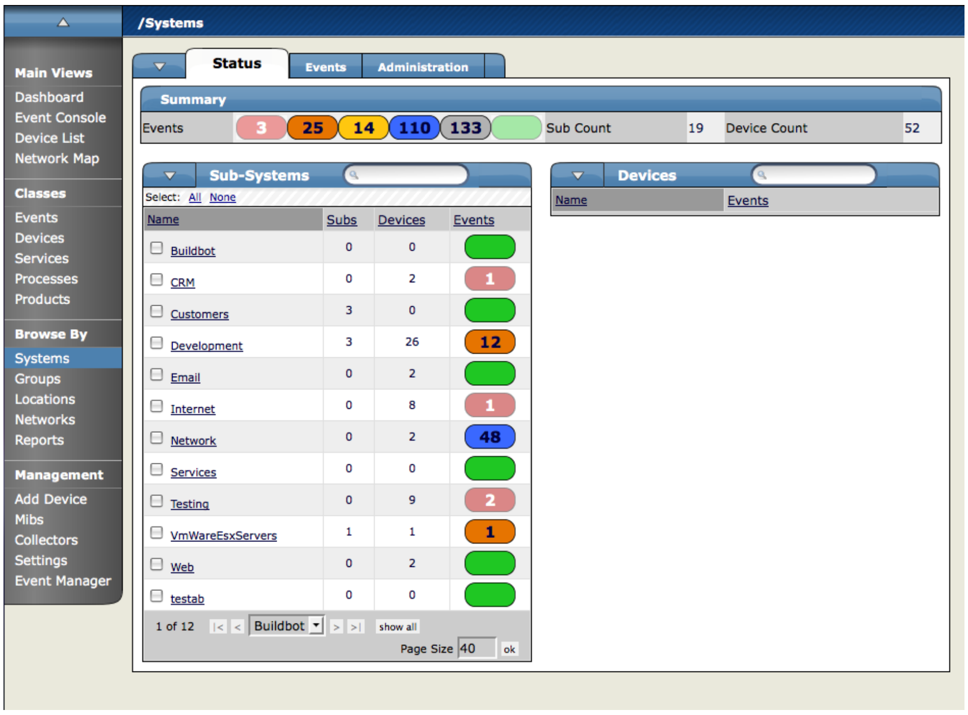
3.1. Adding, Moving and Nesting Systems

To create a new System or Sub-System:

1. From the Navigation menu on the left, under Browse by, select Systems.

   The Sub-systems Status tab appears.

Figure 8.5. Sub-systems Status Menu



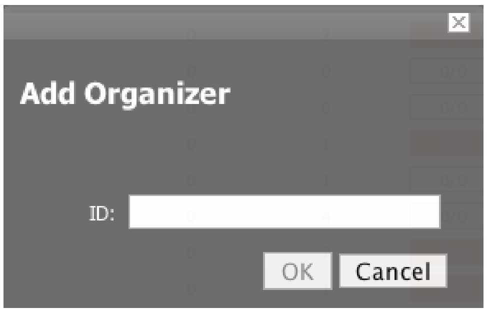
Events	3	25	14	110	133		Sub Count 19	Device Count 52	

Name	Subs	Devices	Events
Buildbot	0	0	🟩
CRM	0	2	1
Customers	3	0	🟩
Development	3	26	12
Email	0	2	🟩
Internet	0	8	1
Network	0	2	48
Services	0	0	🟩
Testing	0	9	2
VmWareEsxServers	1	1	1
Web	0	2	🟩
testab	0	0	🟩

Name	Events

2. Open the Sub-Systems table menu and select the Add New Organizer option. The Add Organizer dialog appears.

Figure 8.6. Add Organizer Dialog



3. In the ID field, Enter the name for the new Sub-system.

4. Click OK.

   The new sub-system is added and appears in the list.

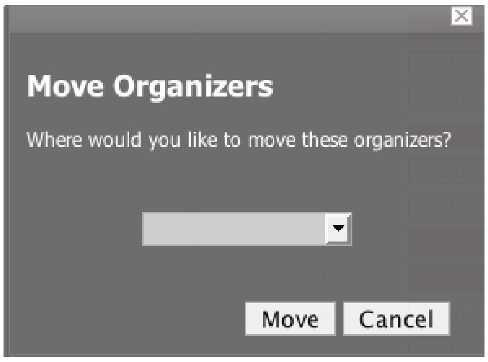
3.1.1. Moving the Sub-System

To move the sub-system into another group or sub-group:

1. Select the system from the system list by clicking the check box next to the systems that you want to move and then open the Sub-Systems table menu to show the Subsystem options.

2. Select the Move Organizer option. The Move Organizer dialog appears:

Figure 8.7. Move Organizer Dialog



3. From the pop-up menu, select where you want to move this system.

4. Click Move.

   The system is moved to the selected system and the attributes page for the system where you moved the system appears.

4. Groups

Using Groups is much like using systems only the intent is to use groups as functional divisions or even monitoring organizers to assign attributes to multiple objects with similar function or even among departmental lines. Note that groups do not appear on the Dashboard. Even though they do not appear on the dashboard, they still help to create additional organizers for monitoring or alerting.

4.1. Adding Groups

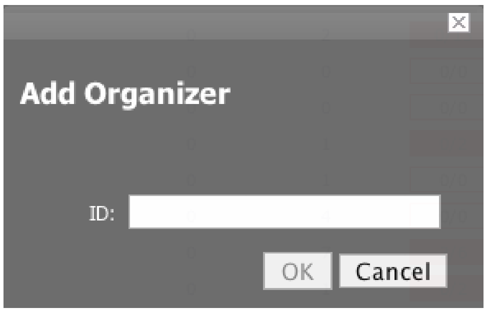
1. From the Navigation menu on the left, under Browse by, select Groups.

2. From the Settings tab, open the Sub-Groups table menu.

3. Select the Add New Organizer option.

   The Add Organizer dialog appears.

Figure 8.8. Add Organizer Dialog



4. In the ID field, Enter the name for the new Sub-Group.

5. Click OK. The new sub-group is added and appears in the list.

4.1.1. Moving Groups

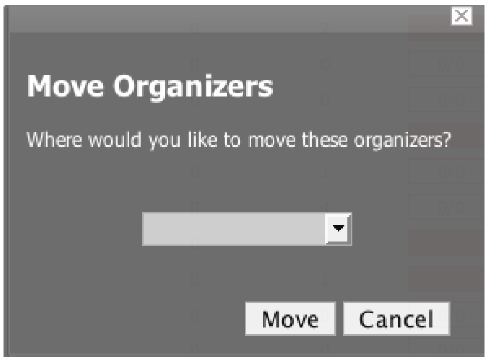
To move the sub-group into another group or sub-group:

1. Select the group from the group list by clicking the check box next to the groups that you want to move and then open the Sub-Groups table menu to show the group options.

2. 1.Select the Move Organizer option.

   The Move Organizer dialog appears.

   Figure 8.9. Move Organizer

3. From the pop-up menu, select where you want to move this group.

4. Click Move.

   The group is moved to the selected group and the attributes page for the group where you moved the group appear.

5. Locations

Using Locations is much like using systems and groups only the intent is to use locations as logical groupings for physical locations of systems. They can be as general as city and state or as specific as rack or closet. It is completely customize-able. Note that locations also do not appear on the Dashboard. Even though they do not appear on the dashboard, they still help to create additional organizers for monitoring or alerting.

5.1. Google Maps Geographic View of Network Health

5.1.1. Overview

Zenoss can map Locations using a Google Maps mashup feature by setting the Location's "Address" property to an address that Google Maps considers valid. The selected Location will appear on the map as a dot. The color of the dot represents the highest severity of any event on any device under that Location. In addition, network connections that span Locations will be represented on the map by lines also color-coordinated to the appropriate to the status of that connection.

To access the Google Map for a network Location, from the left navigation menu, click 'Locations', select the Location you wish to see the network map, and click the 'Map' tab. The Network map is also one of the portlets you can add to the dashboard.

5.1.2. API Key

Before you can use the Google Maps feature, you must register for a Google Maps API key. Free Google Maps API keys are linked to a base URL by which an application must be accessed for Google Maps to function. Enterprise customers do not have this limitation.

To obtain a free Google Maps API key:

1. Point your browser to: http://www.google.com/apis/maps/signup.html

2. Fill in the URL by which you access your Zenoss web interface. Include the port. For example, "http://localhost:8080".

   Users accessing the Zenoss web interface via a different URL than the one you specify here (for example, by IP instead of hostname) will not be able to use the Google Maps feature.

3. Agree to the terms and click OK to receive your API key. Copy it to the clipboard.

5.1.3. Setting an Address for a Location

1. From the left navigation menu, select Locations.

2. From the Summary table, next to Address, click Edit.

   An Edit dialog slides down.

3. In the New Address field, enter a complete address that can, be resolved by Google Maps (if you're unsure, head to http://maps.google.com and see if it maps).

4. Click the Save button.

   You have now created the address for the location that will appear on the map for this location. You must add devices (at least one) to the Location for the dot to appear on the map.

5.1.4. Clearing the Google Maps Cache

Sometimes there can be issues with drawing the maps and seeing the network status of locations or connections. Clearing out the Geocache will solves the problems. Zenoss provides a menu item to clear the geocode cache. To clear the geocode cache, navigate to the Locations tab and from the page menu, select the Clear Geocode Cache menu item.

5.1.5. Network Links

If two devices in the same network are in different mappable Locations, a line will be drawn on the map representing a network connection between the two. If there are multiple separate network connections between the same two Locations, only one line will be drawn. The color of the line will represent the highest severity of any events affecting the connection. These are determined by:

- A ping down event on the device at either end of the connection; or
- Any event on the interface at either end of the connection.

5.1.5.1. zDrawMapLinks Property

Calculating network links on the fly is an expensive procedure. If you have a large number of Devices that have been assigned Locations, drawing those links on the map may take a long time. In order to save time, you can tell

Zenoss not to attempt to draw links for specific networks (for example, a local network comprising many devices that you know does not span multiple Locations).

1. Navigate to the Network and click the "zProperties" tab.
2. Set the "zDrawMapLinks" property to "False."
3. Click "Save."

   Like all zProperties, this setting will be inherited by all subnetworks. If you have few networks for which links would be drawn, it might be a good idea to set zDrawMapLinks to False on /Networks, and only set it to True on a network where you know a Location-spanning WAN connection exists.

5.1.6. Google Maps Example

This example will walk you through creating and displaying some Google map links of devices and sending a test event to see how the links are affected by changes in the system.

1. Navigate to /Network, click "zProperties" and set "zDrawMapLinks" to False. Save.
2. Navigate to /Locations and create two sub-Locations, called "New York" and "Los Angeles".
3. Go to the Status tab of each of the two new Locations. Set the "Address" property of each to "New York, NY" and "Los Angeles, CA" respectively.
4. Set the location of a device in Zenoss to /Locations/New York. Find another device on the same network and set its location to /Locations/Los Angeles.
5. Navigate to /Locations and click on the "Map" tab. Notice that both New York and Los Angeles are represented as dots on the map, but that there is no link drawn between the two.
6. Now navigate to the Network to which both devices are connected. Click the "zProperties" tab and set "zDrawMapLinks" to True. Save.
7. Navigate back to the "Map" tab of /Locations. Notice that a green line is now drawn between New York and Los Angeles.
8. Send an event (See Chapter 10, Section 4 of this guide) to the device in /Locations/New York, with a severity of "Critical." Do not specify a component. Now refresh the /Locations "Map" tab. Notice that the New York dot has become red. Also notice that the link between New York and Los Angeles remains green.
9. Now navigate to the New York device's "OS" tab and determine the id of the component that is connected to the network shared with the Los Angeles device. Send another test event, but this time specify that component. Now refresh the /Locations "Map" tab and notice that the line linking the two locations has become red.

5.2. Adding, Moving, and Nesting Locations

To create a new Location or Sub-Location:

1. From the Navigation menu on the left, under Browse by, select Locations.

   The Sub-locations Status tab appears.

2. Open the Sub-Locations table menu to show Sub-Locations options.

3. Select the Add New Organizer option.

   The Add Organizer dialog appears.

4. In the ID field, Enter the name for the new Sub-location.

5. Click OK.

   The new sub-location is added and appears in the list.

5.2.1. Moving Sub-locations

To move the sub-location into another location or sub-location:

1. Select the location from the location list by clicking the check box next to the locations that you want to move and then from the Sub-Locations table, select the Move Organizer option. The Move Organizer dialog appears.

2. From the pop-up menu, select where you want to move this location.

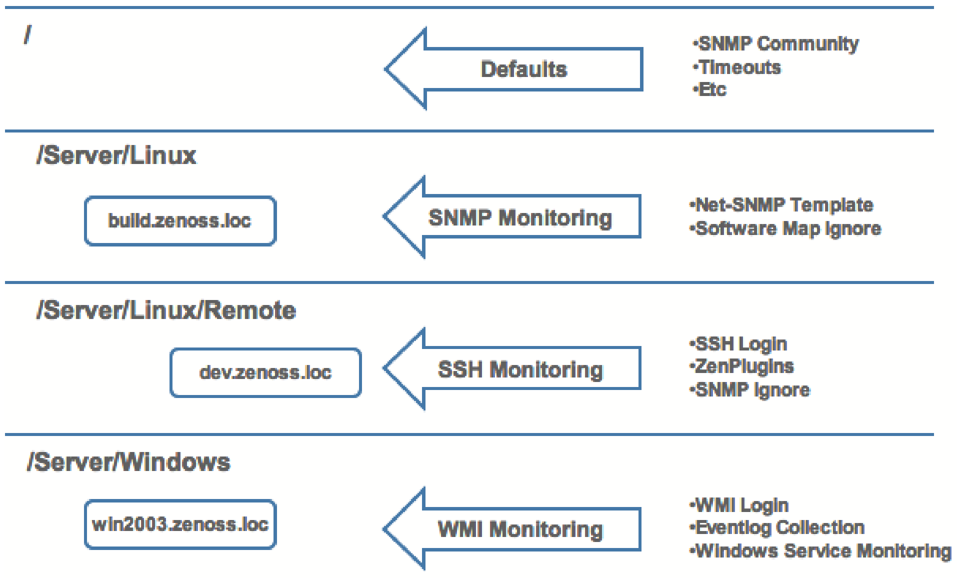
3. Click Move.

   The location is moved to the selected location and the attributes page for the location where you moved the location appears.

6. Inheritance

Inheritance is how many attributes are applied to device at different points in the device hierarchy. The following diagram shows an example of how and where zProperties can be set throughout the device class tree.

Figure 8.10. Zenoss Device Class Tree and Inheritance



In this example, you can see that the default properties can be set at the highest level (/), as you go further down the hierarchy, though, you can see that you can override any of the zProperties set at the root level. The next two lines show how the device tree further defines properties for Linux servers. If you wanted to set up and use SNMP monitoring for the all Linux servers (inclusive of) build.zenoss.loc, you can change these properties at the /Server/Linux level. Now if you wanted to change how you collected information for remote Linux servers (such as dev.zenoss.loc) you can create a sub-group within the /Server/Linux group called /Server/Linux/Remote and set these servers will use SSH monitoring and change the associated properties for that sub-group. Now also within the /Server group you can create another sub-group for Windows servers that change the zProperties specifically for WMI monitoring. All of these zProperties and groupings co-exist with any changes made lower in the hierarchy taking priority. It is very similar to a directory tree only you can place items in multiple organizers.

7. Multiple Mix-In Inheritance and Performance (RRD) Template Binding

Performance configurations are stored in objects called RRDTemplates, frequently just referred to as templates. Templates contain other objects that define where and how to obtain performance data, thresholds for that data and graphs of the data. A template can be defined anywhere in the Device Class hierarchy or on an individual device.

The determination of which templates apply to what objects is called binding. There are two steps to binding: first is determining which template names are appropriate and second is locating the correct templates with those names. The template names used for components such as FileSystems, Interfaces, etc is the same as the components' meta type. For example, file systems use templates name FileSystem. Devices are more complex because users can modify the names of the templates to be used and can specify more than one name. This list of template names is stored in the zDeviceTemplates zProperty. This zProperty can be edited directly on the zProperties page of any Device or Device Class. It can also be edited through the Bind Templates menu item and dialog available from the Templates page of any Device or Device Class. The second step of template binding is finding the correct template with the given name. As with zProperties, templates can be defined directly on a Device or they can be inherited from one of the Device Classes above it. The first templates found with the correct names searching up the hierarchy are used. Any similarly named templates farther up the hierarchy are ignored.

7.1. Template Binding Example 1

You add a new device at /Devices/Server/Linux/Example1Server. You haven't edited its zDeviceTemplates so it is inheriting the value of "Device" from the root Device Class, in this case "/Devices". Zenoss looks to see if there is a template named Device defined on Example1Server itself. There is not, so it checks /Devices/Server/Linux. There is a template named Device defined for that Device Class, so that template is used for Example1Server. There is also a template named Device defined at /Devices, but that one isn't used in this case because the one at /Devices/Server/Linux overrides it.

7.2. Template Binding Example 2

You want to perform specific monitoring of servers running a certain web application, but those servers are spread across several different Device Classes. You create a new template at /Devices called WebApplication with the appropriate Data Sources, Thresholds and Graphs. You then append the name "WebApplication" to the zDeviceTemplates zProperty for the Devices Classes and/or individual Devices running this web application. (You could use the Bind Templates menu item and dialog on the Templates page if you prefer instead of directly editing zDeviceTemplates.)

Chapter 9. zProperties

1. About zProperties

The previous section hinted at zProperties and how they are inherited throughout the system. This section will detail those zProperties and how to access them to set them, and what each one does.

zProperties are properties that Zenoss uses to specify various items that control the information in the system. ZProperties are very powerful and can automate many tasks that are repeatable or things you would normally have to do on an individual basis and apply them in many places or groups.

zProperties are configured either for all items, for an individual device, or for any devices that fall further down in the hierarchy tree. zProperties are also zen-packable meaning the settings you set here will be added to any ZenPack you create for wherever in the class hierarchy you are. zProperties exist for:

- Events
- Devices
- Services
- Networks

2. Event zProperties

To access Event zProperties, from the left navigation menu, select Event and click the zProperties tab. You can also set zProperties for any event class in the events hierarchy by navigating to the level in the event hierarchy where you want to set the zProperty and clicking the zProperties tab.

Table 9.1.

Property Name	Property Type	Description
zEventAction	string	Location to which an event will be stored. Possible values are: status, history and drop. Default is status meaning the event will be an “active” event. History sends the event directly to the history table. Drop tells the system to discard the event.
zEventClearClasses	lines	A list of classes that a clear event should clear in addition to its own class.
zEventSeverity	int	Allows you to override the severity value of an event. If this is -1 it is ignored. Possible values are 0 – 5.

3. Device zProperties

To access Device zProperties, from the left navigational menu, select Devices and then click the zProperties tab. You can also select any of the Device Classes and the zProperties tab to set zProperties anywhere in the Device hierarchy. To set zProperties for an individual Device, navigate to that device, open the page menu, select the More option and then the zProperties tab.

Table 9.2.

Property Name	Property Type	Description
zCollectorClientTimeout	int	Allows you to set the timeout time of the collector client in seconds
zCollectorDecoding	string	Converts incoming characters to Unicode.
zCollectorLogChanges	boolean	Switch to indicate whether to log changes.
zCollectorPlugins	lines	A link to tall collector plugins for this device.
zCommandCommandTimeout	float	Time to wait for a command to complete.
zCommandCycleTime	int	The cycle time you use when executing zCommands for this device or organizer.
zCommandExistanceTest	string	***
zCommandLoginTimeout	float	Time to wait for a login prompt.
zCommandLoginTries	int	Number of times to attempt login.
zCommandPassword	int	Password to use when performing command login.
zCommandPath	string	Default path where ZenCommand plug-ins are installed on the local Zenoss box or a remote box in the case where SSH is used to run the command.
zCommandPort	int	Port to connect to when performing command collection.
zCommandProtocol	string	Protocol to use when performing command collection. Possible values are ssh, telnet.
zCommandSearchPath	lines	Path to search for any commands.
zCommandUsername	string	Username to use when performing command collection.
zDeviceTemplates	lines	Templates associated with this device. Linked by name.
zFileSystemMapIgnoreNames	string	Regular expression of file system names to ignore.
zFileSystemMapIgnoreTypes	lines	DO NOT USE
zIcon	lines	Icon to represent the device wherever device Icon is shown (network map, device status page etc.) Most devices such as windows servers, Linux servers, and routers have images set by default
zIfDescription	boolean	Show the interface description field in the interface list.
zInterfaceMapIgnoreNames	string	A regular expression filters out interfaces that should not be discovered.

Property Name	Property Type	Description
zInterfaceMapIgnoreTypes	string	A regular expression filters out interface maps that should not be discovered.
zIpServiceMapMaxPort	int	Highest port to scan, default to 1024.
zKeyPath	lines	Path to the key to access a device.
zLinks	text	Place to enter any links associated with the device.
zLocalInterfaceNames	string	A regular expression that uses interface name to determine whether or not the IPs on an interface should be incorporated into Zenoss's network map. For instance, a loopback interface "lo" might be excluded.
zLocalIpAddresses	int	IP addresses that should be excluded from the network map. for instance 127.x address most likely be excluded. If you have addresses that you reuse for connections between clustered machines they might be added here as well.
zMaxOIDPerRequest	int	The maximum number of OIDs to be sent by Zenoss' SNMP collection daemons when querying information. Some devices have small buffers for handling this information so the number should be lowered.
zPingInterfaceDescription	string	A catalog query string that will find interfaces to be pinged by description.
zPingInterfaceName	string	A catalog query string that will find interfaces to be pinged by name.
zPingMonitorIgnore	boolean	Whether or not to ping the device.
zProdStateThreshold	int	Production state threshold at which Zenoss will begin to monitor a device. Default of 500 equals Pre-Productions.
zPythonClass	string	DO NOT USE
zRouteMapCollectOnlyIndirect	boolean	Only collect routes that are directly connected to the device.
zRouteMapCollectOnlyLocal	boolean	Only collect local routes (these are usually manually configured vs. those learned through a routing protocol.)
zSnmpAuthPassword	string	The shared private key used for authentication. Must be at least 8 characters long.
zSnmpAuthType	string	use either "MD5" or "SHA" signatures to authenticate SNMP requests
zSnmpCommunities	lines	Array of SNMP community strings that the ZenModeler will try to use when collecting SNMP information.

Property Name	Property Type	Description
zSnmpCommunity	string	Community to be used when collecting SNMP information. If it is different than what is found by ZenModeler, it will be set on the modeled device.
zSnmpMonitorIgnore	boolean	Whether or not to ignore monitoring SNMP on a device.
zSnmpPort	int	Port that the SNMP agent listens on.
zSnmpPrivPassword	string	The shared private key used for encrypting SNMP requests. Must be at least 8 characters long.
zSnmpPrivType	string	Either "DES" or "AES" cryptographic algorithms.
zSnmpSecurityName	string	The Security Name (user) to use when making SNMPv3 requests.
zSnmpTimeout	float	Timeout time in seconds for an SNMP request
zSnmpTries	int	Amount of tries to collect SNMP data
zSnmpVer	string	SNMP version used. Valid values are v1, v2.
zStatusConnectTimeout		The amount of time that zenstatus should wait before marking an IP Service down.
zSysedgeDiskMapIgnoreNames		Currently unused.
zTelnetEnable	boolean	When logging into a Cisco device issue the enable command to enable access during command collection.
zTelnetEnableRegex	string	Regular expression to match the enable prompt.
zTelnetLoginRegex	string	Regular expression to match the login prompt.
zTelnetPasswordRegex	string	Regular expression to match the password prompt.
zTelnetPromptTimeout	float	Time to wait for the telnet prompt to return.
zTelnetSuccessRegexList	lines	List of regular expressions to match the command prompt.
zTelnetTermLength	boolean	On a Cisco device, set term length to Zero.
zWinEventlog	boolean	Whether or not to send the log.
zWinEventlogMinSeverity	int	Sets minimum severity to collect from the win event log. Important to note that the higher the number, the lover the severity. 1 being the most sever and 5 being the least severe.
zWinPassword	string	The password used to remotely login if it is a Windows machine.
zWinServices		

Property Name	Property Type	Description
zWinUser	string	The user name used to remotely login if it is a Windows machine.
zWmiMonitorIgnore	boolean	Use this to turn on or off all WMI monitoring.
zXmlRpcMonitorIgnore	boolean	Use this to turn on or off all XML/RPC monitoring.

4. Service zProperties

To access Service zProperties, from the left navigation menu, select Services and the click the zProperties tab. You can also access the Services zProperties tab anywhere in the services hierarchy by going to the level where you want to set the zProperty and clicking the zProperties tab.

Table 9.3.

Property Name	Property Type	Description
zFailSeverity	int	Determines what severity to send for the specified service.
zHideFieldsFromList	lines	Fields to hide from Services instance list
zMonitor	boolean	Tells whether or not to monitor a ser-vice.

5. Process zProperties

To access Process zProperties, from the left navigation menu, select Processes and the click the zProperties tab. You can also access the Process zProperties tab anywhere in the processes hierarchy by going to the level where you want to set the zProperty and clicking the zProperties tab.

Table 9.4.

Property Name	Property Type	Description
zAlertOnRestart	boolean	Determines whether or not to send an event if the specified process is restar-ted.
zCountProcs	boolean	Determines the number of instances of the process that are running.
zFailSeverity	int	Determines what severity to send for the specified process.
zMonitor	boolean	Tells whether or not to monitor a pro-cess.

6. Network zProperties

To access Network zProperties, from the left navigation menu, select Networks and then click the zProperties tab. You can also access the zProperties tab for any sub-network that exists with in the Networks page.

Table 9.5.

Property Name	Property Type	Description
zAutoDiscover	boolean	Should zendisc perform auto-discovery on this network
zDefaultNetworkTree	lines	List of netmask numbers to use when creating network containers. Default is 24, 32 which will make /24 networks at the top level of the networks tree if a network us smaller than /24.
zPingFailThresh	int	Number of pings to sent without being returned before Zendisc removes the device.

7. Manufacturer zProperties

Table 9.6.

Property Name	Property Type	Description
zDeviceClass	string	FUTURE USE
zDeviceGroup	string	FUTURE USE
zSystem	string	FUTURE USE

Chapter 10. Device Inventory and Configuration

1. What is Inventory and Configuration in Zenoss?

Inventory and configuration refers to the population of the device database and also the collection of information about the devices in the system. This is often referred to as “modeling”, and Zenoss can perform this process using auto-discovery, one-at-a-time using the web UI, or both loaded via an XML file.

2. How Does Zenoss Model Devices?

Zenoss can model devices using SNMP, SSH, or the Telnet transport protocol. Each technique for modeling produces a varying richness of information in the model. SNMP modeling often provides the most complete set of model information, and SSH/Telnet is typically used to augment the model when an SNMP Agent does not report a specific piece of critical information.

3. The ZenModeler Daemon

Modeling is performed using the “ZenModeler” daemon. ZenModeler iterates over the list of devices in the system and attempts to auto-discover the sub-components of each device. This includes network interfaces, file systems, processes, and IP services (as well as others). By default, the system is setup to perform complete remodeling every 6 hours. This may be too often for large deployments. Often this process is run only once per day from a cron job.

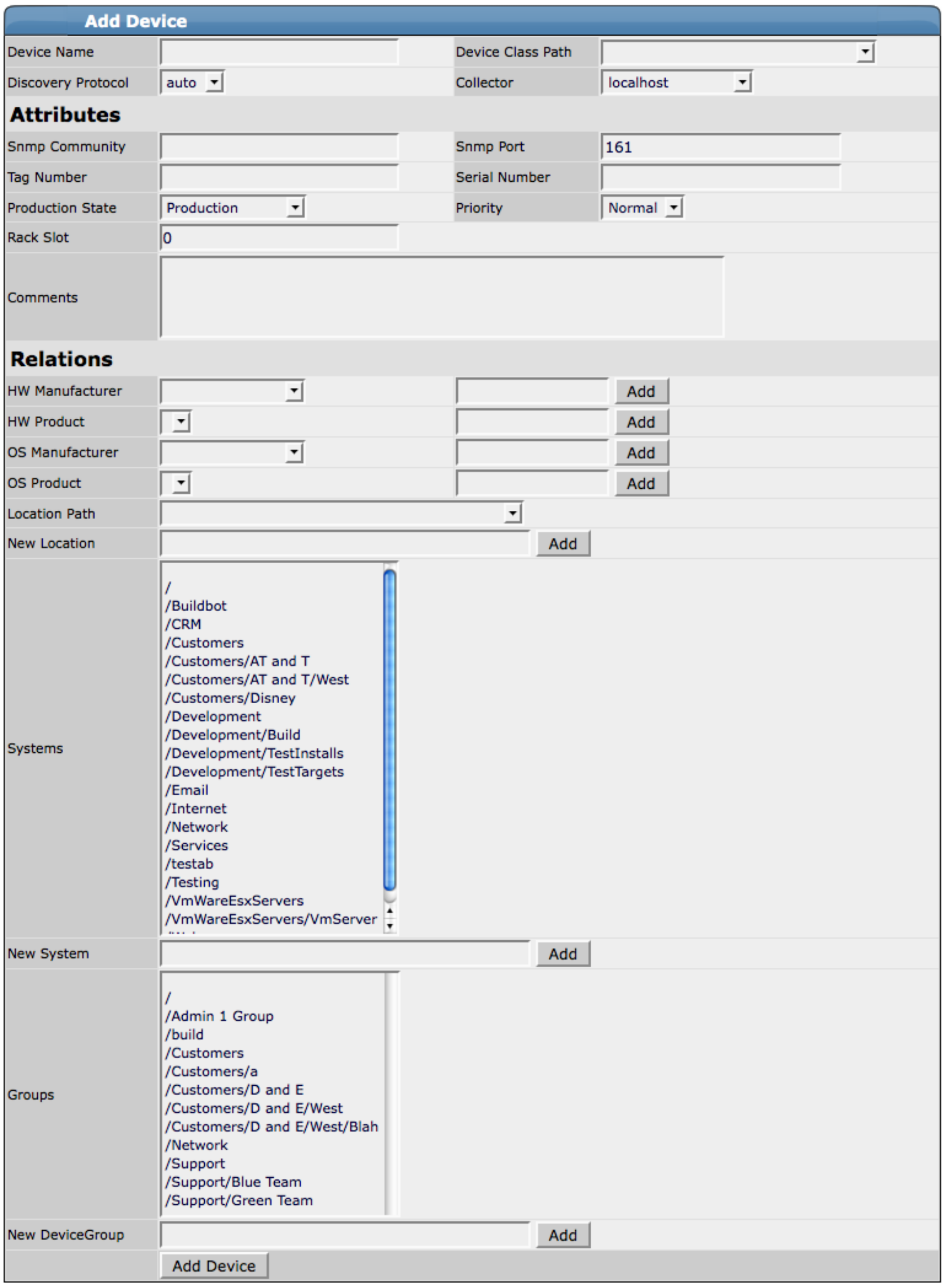
4. Adding an Individual Device

Zenoss will add, model, and monitor those devices you add to the system. This section will walk you through adding an individual device.

1. From the menu on the left side of the page, select Add Device.

   The Add Device page appears.

Figure 10.1. Add Device - Full Page



Add Device

Device Name		Device Class Path	
Discovery Protocol	auto	Collector	localhost

Attributes

Snmp Community		Snmp Port	161
Tag Number		Serial Number	
Production State	Production	Priority	Normal
Rack Slot	0		
Comments			

Relations

HW Manufacturer			Add
HW Product			Add
OS Manufacturer			Add
OS Product			Add
Location Path			
New Location		Add	

Systems:

/
/Buildbot
/CRM
/Customers
/Customers/AT and T
/Customers/AT and T/West
/Customers/Disney
/Development
/Development/Build
/Development/TestInstalls
/Development/TestTargets
/Email
/Internet
/Network
/Services
/testab
/Testing
/VmWareEsxServers
/VmWareEsxServers/VmServer

New System — Add

Groups:

/
/Admin 1 Group
/build
/Customers
/Customers/a
/Customers/D and E
/Customers/D and E/West
/Customers/D and E/West/Blah
/Network
/Support
/Support/Blue Team
/Support/Green Team

New DeviceGroup — Add

Add Device

2. In the Device Name field, enter the network (DNS) name or IP address of the device. Additionally, you can see all of the fields and areas where you can add additional information about a device.

3. Choose a device class from the Device Class drop-down list. We will classify this sever as a windows server so we choose /Server/Windows as the device class path.

4. Select a discovery protocol. You can choose from snmp, or none.

   These methods are described in the above section about how Zenoss models devices.

5. Device Name, Device Class Path and Discovery Protocol are the only required fields for adding a device. Zenoss will attempt to fill in the following fields or they can be filled in later. Information you add here may conflict with information Zenoss discovers about the device.

   NOTE: - When adding Cisco routers in classes other than /Network, you should set the zProperty for zIfDescription to True. This will give you additional information about Cisco routers. The option is already set to True by default for the /Network class.

6. Scroll down to the bottom of the page and click the Add Device button.

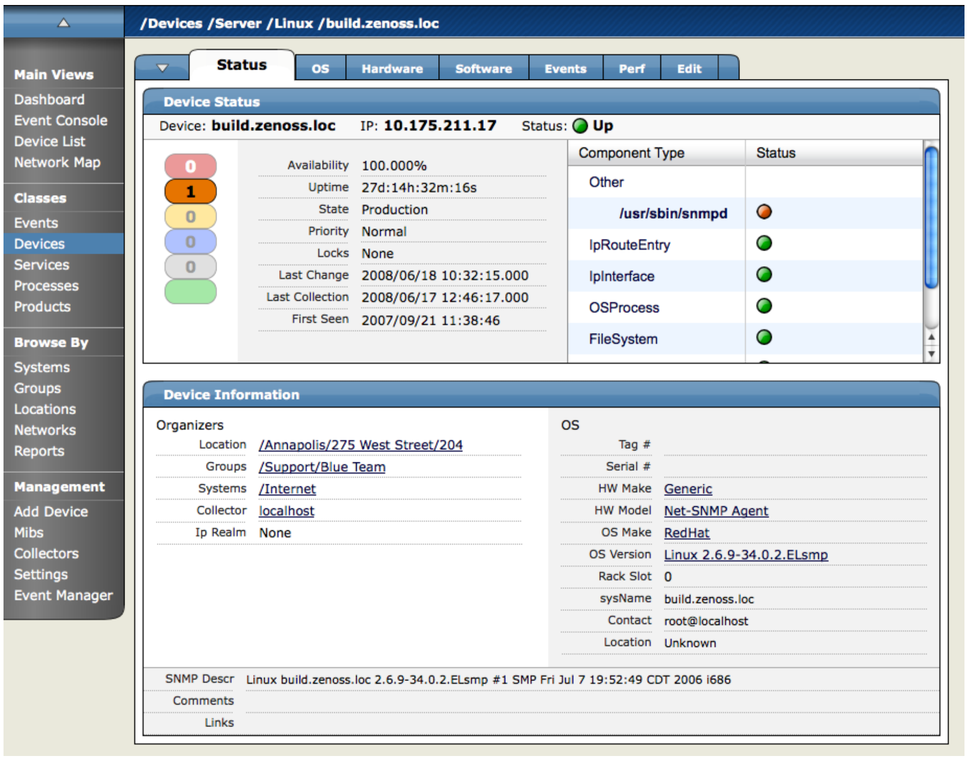
   A Status page appears showing a log of the operations Zenoss is using to gather information about the device.

7. Scroll to the bottom of the page that results when you add a device. (You may have to scroll for a bit). There will be a link you can click to navigate to a device. Click the link that says:

   Navigate to device <device name>

   The Main Device page appears showing the Status Tab.

Figure 10.2. Main Device Page



/Devices /Server /Linux /build.zenoss.loc

Device Status

Device: **build.zenoss.loc** IP: **10.175.211.17** Status: ● **Up**

0, 1, 0, 0, 0

Availability	100.000%
Uptime	27d:14h:32m:16s
State	Production
Priority	Normal
Locks	None
Last Change	2008/06/18 10:32:15.000
Last Collection	2008/06/17 12:46:17.000
First Seen	2007/09/21 11:38:46

Component Type	Status
Other	
/usr/sbin/snmpd	●
IpRouteEntry	●
IpInterface	●
OSProcess	●
FileSystem	●

Device Information

Organizers

Location	/Annapolis/275 West Street/204
Groups	/Support/Blue Team
Systems	/Internet
Collector	localhost
Ip Realm	None

OS

Tag #	
Serial #	
HW Make	Generic
HW Model	Net-SNMP Agent
OS Make	RedHat
OS Version	Linux 2.6.9-34.0.2.ELsmp
Rack Slot	0
sysName	build.zenoss.loc
Contact	root@localhost
Location	Unknown

SNMP Descr	Linux build.zenoss.loc 2.6.9-34.0.2.ELsmp #1 SMP Fri Jul 7 19:52:49 CDT 2006 i686
Comments	
Links	

5. Add an Individual Device with Context

You can also add an individual device with context. This means that wherever in the hierarchy you choose to add the device, that is where the device will appear in the hierarchy rather than the default of adding a device with no context where the device goes into the /Discovered hierarchy by default.

To add a device with context:

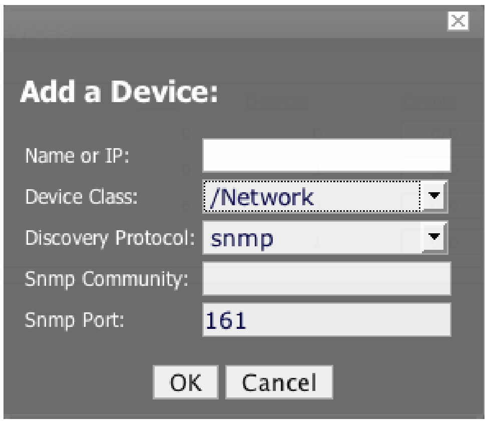
1. Navigate to the place in the Device tree where you want to add the device.

   This is the context part.

2. Open the page menu, select the Manage option and then the Add Device option.

   The Add Device dialog appears.

   Figure 10.3. Add Device with Context Dialog

   Add a Device:

   Name or IP:

   Device Class: /Network

   Discovery Protocol: snmp

   Snmp Community:

   Snmp Port: 161

3. Fill out the fields the same as when you add a device without context (The only difference here is that the Device Class is pre-selected.)

4. Click OK.

   The Device is added into the selected Device Class and the main Device page appears showing the Status Page.

6. Auto-Discovery of Devices

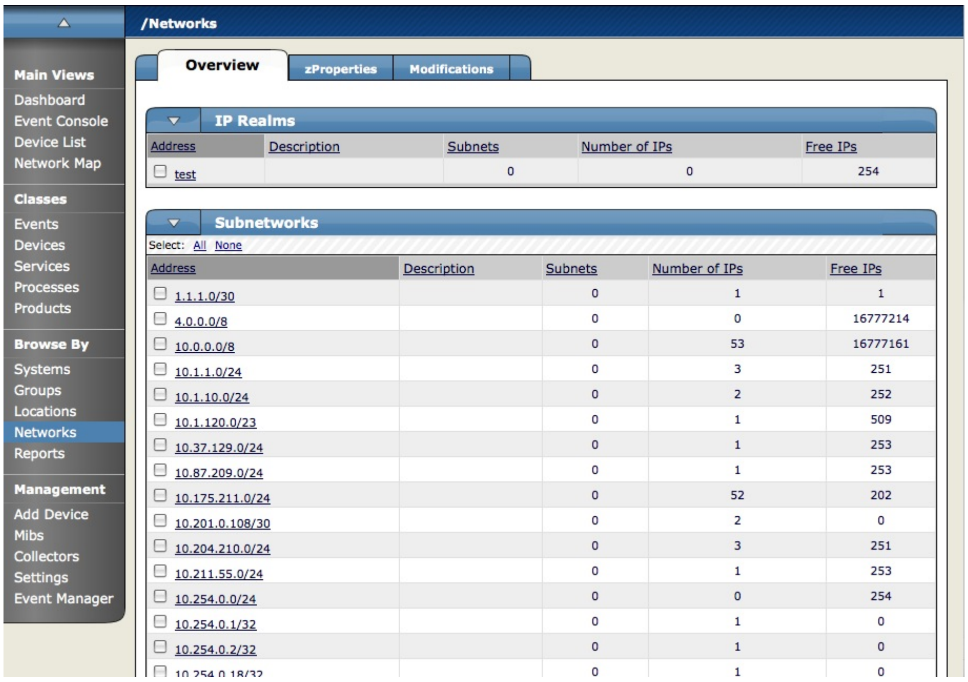
Zenoss can SNMP-walk the entire network and the routing tables and then model each individual device to add them all at once to the database. The daemon that does this is called ZenDisc. To perform auto-discovery, the machine on which Zenoss is installed must have an SNMP agent running.

To add all of the devices on a given network or subnetwork into the Zenoss system:

1. From the left navigation menu select networks.

   The Networks Overview page appears.

Figure 10.4. Networks Overview Tab



IP Realms

Address	Description	Subnets	Number of IPs	Free IPs
test		0	0	254

Subnetworks

Select: All None

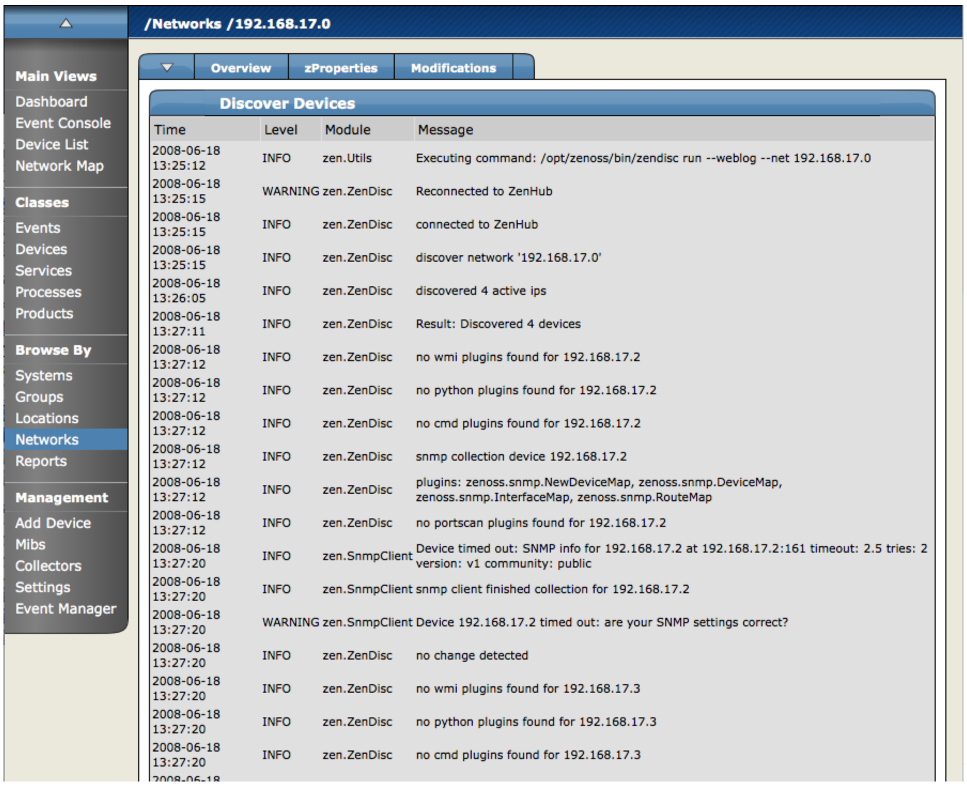
Address	Description	Subnets	Number of IPs	Free IPs
1.1.1.0/30		0	1	1
4.0.0.0/8		0	0	16777214
10.0.0.0/8		0	53	16777161
10.1.1.0/24		0	3	251
10.1.10.0/24		0	2	252
10.1.120.0/23		0	1	509
10.37.129.0/24		0	1	253
10.87.209.0/24		0	1	253
10.175.211.0/24		0	52	202
10.201.0.108/30		0	2	0
10.204.210.0/24		0	3	251
10.211.55.0/24		0	1	253
10.254.0.0/24		0	0	254
10.254.0.1/32		0	1	0
10.254.0.2/32		0	1	0
10.254.0.18/32		0	1	0

2. Click the checkbox next to the network you want to add all of the devices from. You can also use Subnetworks table menu to add a new network to the list.

3. Once you have selected the network, open the Subnetworks table menu and select Discover Devices.

4. The Discover Device page will appear showing you the status of all the device collections going on.

Figure 10.5. Auto-Discovery of Devices



Time	Level	Module	Message
2008-06-18 13:25:12	INFO	zen.Utils	Executing command: /opt/zenoss/bin/zendisc run --weblog --net 192.168.17.0
2008-06-18 13:25:15	WARNING	zen.ZenDisc	Reconnected to ZenHub
2008-06-18 13:25:15	INFO	zen.ZenDisc	connected to ZenHub
2008-06-18 13:25:15	INFO	zen.ZenDisc	discover network '192.168.17.0'
2008-06-18 13:26:05	INFO	zen.ZenDisc	discovered 4 active ips
2008-06-18 13:27:11	INFO	zen.ZenDisc	Result: Discovered 4 devices
2008-06-18 13:27:12	INFO	zen.ZenDisc	no wmi plugins found for 192.168.17.2
2008-06-18 13:27:12	INFO	zen.ZenDisc	no python plugins found for 192.168.17.2
2008-06-18 13:27:12	INFO	zen.ZenDisc	no cmd plugins found for 192.168.17.2
2008-06-18 13:27:12	INFO	zen.ZenDisc	snmp collection device 192.168.17.2
2008-06-18 13:27:12	INFO	zen.ZenDisc	plugins: zenoss.snmp.NewDeviceMap, zenoss.snmp.DeviceMap, zenoss.snmp.InterfaceMap, zenoss.snmp.RouteMap
2008-06-18 13:27:12	INFO	zen.ZenDisc	no portscan plugins found for 192.168.17.2
2008-06-18 13:27:20	INFO	zen.SnmpClient	Device timed out: SNMP info for 192.168.17.2 at 192.168.17.2:161 timeout: 2.5 tries: 2 version: v1 community: public
2008-06-18 13:27:20	INFO	zen.SnmpClient	snmp client finished collection for 192.168.17.2
2008-06-18 13:27:20	WARNING	zen.SnmpClient	Device 192.168.17.2 timed out: are your SNMP settings correct?
2008-06-18 13:27:20	INFO	zen.ZenDisc	no change detected
2008-06-18 13:27:20	INFO	zen.ZenDisc	no wmi plugins found for 192.168.17.3
2008-06-18 13:27:20	INFO	zen.ZenDisc	no python plugins found for 192.168.17.3
2008-06-18 13:27:20	INFO	zen.ZenDisc	no cmd plugins found for 192.168.17.3

This command will first model the monitoring machine and then walk through the routing tables of all routers it can find. Auto-discovery will go as far as valid SNMP access is found or until a network is discovered in the DMD that has its zAutoDiscover property set to False.

Routers discovered through this process will be placed in the device path /Network/Router.

When devices are discovered they are placed into the /Discovered device path. They should then be moved to a more specific part of the tree. Servers are normally organized by OS, so windows machines might go to /Server/Windows. Other information can be added to a device, like its Business System or its Location, using the Edit tab on a device.

When devices are discovered they are placed into the /Discovered device path. They should then be moved to a more specific part of the tree. Servers are normally organized by OS, so windows machines might go to /Server/Windows. Other information can be added to a device, like its Business System or its Location, using the Edit tab on a device.

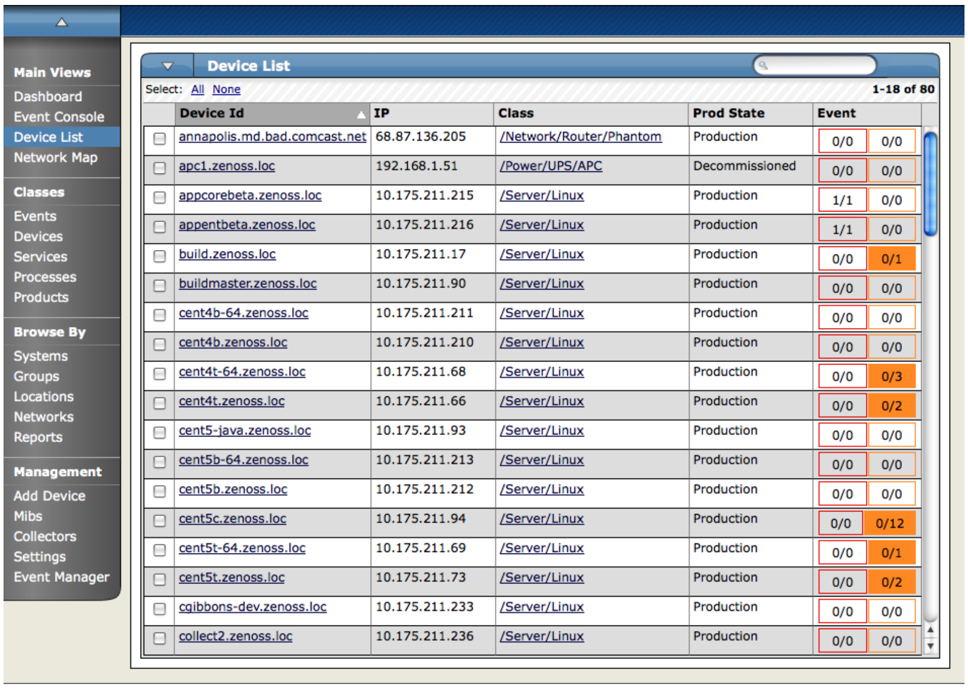
7. Device List

The Device List shows a list of all the Devices in the system. You can search for all devices, see an event summary/ You can also perform some management tasks to apply to devices or groups of devices in the list.

To access the device list, from the left navigation menu, select Device List.

The Device List appears.

Figure 10.6. Device List



1-18 of 80

	Device Id	IP	Class	Prod State	Event	
☐	annapolis.md.bad.comcast.net	68.87.136.205	/Network/Router/Phantom	Production	0/0	0/0
☐	apc1.zenoss.loc	192.168.1.51	/Power/UPS/APC	Decommissioned	0/0	0/0
☐	appcorebeta.zenoss.loc	10.175.211.215	/Server/Linux	Production	1/1	0/0
☐	appentbeta.zenoss.loc	10.175.211.216	/Server/Linux	Production	1/1	0/0
☐	build.zenoss.loc	10.175.211.17	/Server/Linux	Production	0/0	0/1
☐	buildmaster.zenoss.loc	10.175.211.90	/Server/Linux	Production	0/0	0/0
☐	cent4b-64.zenoss.loc	10.175.211.211	/Server/Linux	Production	0/0	0/0
☐	cent4b.zenoss.loc	10.175.211.210	/Server/Linux	Production	0/0	0/0
☐	cent4t-64.zenoss.loc	10.175.211.68	/Server/Linux	Production	0/0	0/3
☐	cent4t.zenoss.loc	10.175.211.66	/Server/Linux	Production	0/0	0/2
☐	cent5-java.zenoss.loc	10.175.211.93	/Server/Linux	Production	0/0	0/0
☐	cent5b-64.zenoss.loc	10.175.211.213	/Server/Linux	Production	0/0	0/0
☐	cent5b.zenoss.loc	10.175.211.212	/Server/Linux	Production	0/0	0/0
☐	cent5c.zenoss.loc	10.175.211.94	/Server/Linux	Production	0/0	0/12
☐	cent5t-64.zenoss.loc	10.175.211.69	/Server/Linux	Production	0/0	0/1
☐	cent5t.zenoss.loc	10.175.211.73	/Server/Linux	Production	0/0	0/2
☐	cgibbons-dev.zenoss.loc	10.175.211.233	/Server/Linux	Production	0/0	0/0
☐	collect2.zenoss.loc	10.175.211.236	/Server/Linux	Production	0/0	0/0

7.1. Managing Multiple Devices using the Device List

You can use the Device list page menu to manage multiple devices by selecting the devices (by clicking in the boxes next to the devices you want to move) and then selecting the appropriate menu option from the menu.

Available options for managing multiple devices are:

- Move to class – for moving devices to new classes
- Set Groups – for assigning devices to groups
- Set Systems – for assigning devices to systems
- Set Location – for assigning devices to locations
- Set Monitor – for assigning monitors for collecting from selected devices
- Delete devices – for removing devices from the system
- Lock devices – for providing configuration locks for devices

8. Individual Device Tabs

Once Zenoss discovers and models a device it assigns certain properties and attributes to a device and classifies these discoveries throughout a Device's information tabs. The following tabs are available when an individual device is selected.

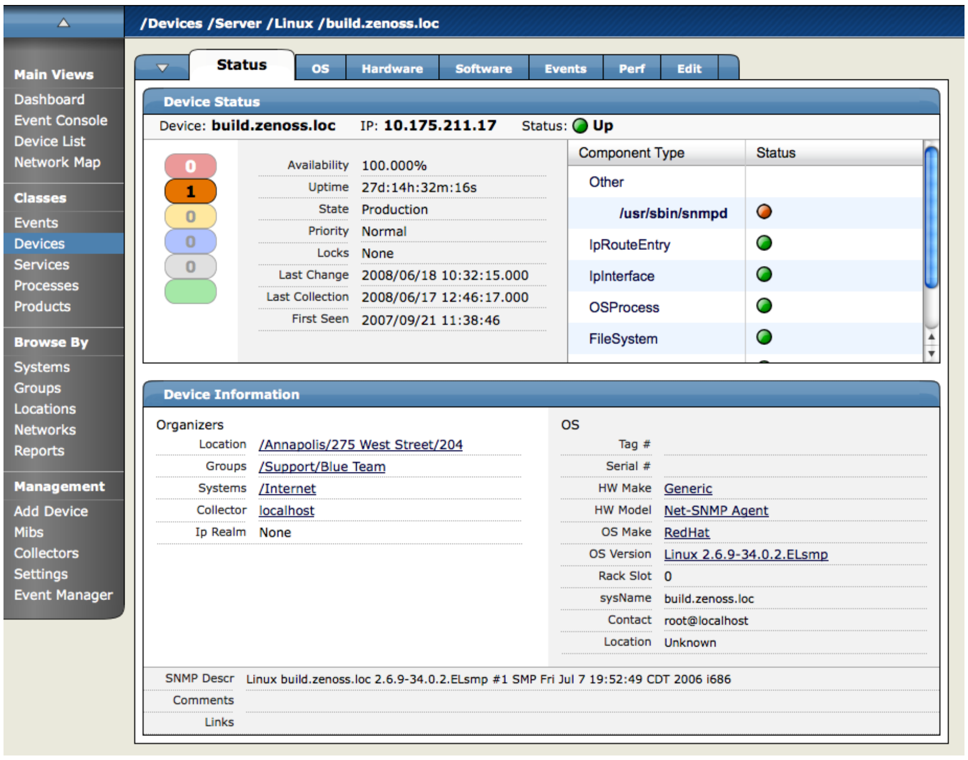
8.1. Device Status Tab

The Status tab is the default tab that is shown when you click on a device.

There is a Device Status table that shows important device status at a glance. The number of events grouped by severity can be found on the left side of this table. You can click on the 'Event Rainbow' to view the list of events for the device.

Also on the left side of the Device Status table is the device availability, uptime, last collection and modified time. On the right side of the Device Status table is the Component Status list. Each item in the list is a type of device component, which are IpService, WinService, IpRouteEntry, IpInterface, CPU, FileSystem, and Other. The status of each device component type is determined by the collective status of the monitored components of the same type. If the IpService status is green, then all monitored IpServices on this device are functioning normally. If there is an event related to a monitored IpService, the component and highest severity event associated with that component will be displayed in the status. If there is an event unrelated to a known component, it will be placed under the component type Other. If you click on the component type, you will be sent to the OS tab of the Device where you can manage the components on this device.

Figure 10.7. Individual Device - Status Tab



/Devices /Server /Linux /build.zenoss.loc

Device Status

Device: **build.zenoss.loc** IP: **10.175.211.17** Status: ● **Up**

0, 1, 0, 0, 0

Availability	100.000%
Uptime	27d:14h:32m:16s
State	Production
Priority	Normal
Locks	None
Last Change	2008/06/18 10:32:15.000
Last Collection	2008/06/17 12:46:17.000
First Seen	2007/09/21 11:38:46

Component Type	Status
Other	
/usr/sbin/snmpd	●
IpRouteEntry	●
IpInterface	●
OSProcess	●
FileSystem	●

Device Information

Organizers

Location	/Annapolis/275 West Street/204
Groups	/Support/Blue Team
Systems	/Internet
Collector	localhost
Ip Realm	None

OS

Tag #	
Serial #	
HW Make	Generic
HW Model	Net-SNMP Agent
OS Make	RedHat
OS Version	Linux 2.6.9-34.0.2.ELsmp
Rack Slot	0
sysName	build.zenoss.loc
Contact	root@localhost
Location	Unknown

SNMP Descr	Linux build.zenoss.loc 2.6.9-34.0.2.ELsmp #1 SMP Fri Jul 7 19:52:49 CDT 2006 i686
Comments	
Links	

8.1.1. Device Status Tab Example

1. Click on the "OS" tab of any device. If you do not have an IpInterface eth0, add the interface by clicking the "Add IpInterface" menu item in the "Add..." submenu. Once you have added the component you'll want it to be monitored.

2. On the IpInterface edit screen, set Monitor to "True" and click "Save".

3. Now return to the Device Status page.

   The IpInterface type should appear.

4. Now update the table by sending an event linked to a component and not linked to any known components. From the left navigation menu, click Events.

5. From the page menu, click on the "Add Event" menu item. Enter the name of the device. In the Component field, enter "eth0". Leave the severity as "Critical" and click the "OK" button.

6. Now send another event, once again use the "Add Event" menu item. Enter the name of the device. In the Component field, enter "Not a known component" (or any other string that isn't a name of a component on this device) Set the severity to "Error" and click the "OK" button.

7. Now refresh the Device Status page.

   You should see that the IpInterface type group has a component "eth0" with a red status indicating that a Critical event has been received.

8. If you click on the status bubble, you will be taken to the Events page for that device. You'll also notice that you have a new type group, Other, with the component name we chose earlier. The status color of this component should be orange which represents the Error event you created.

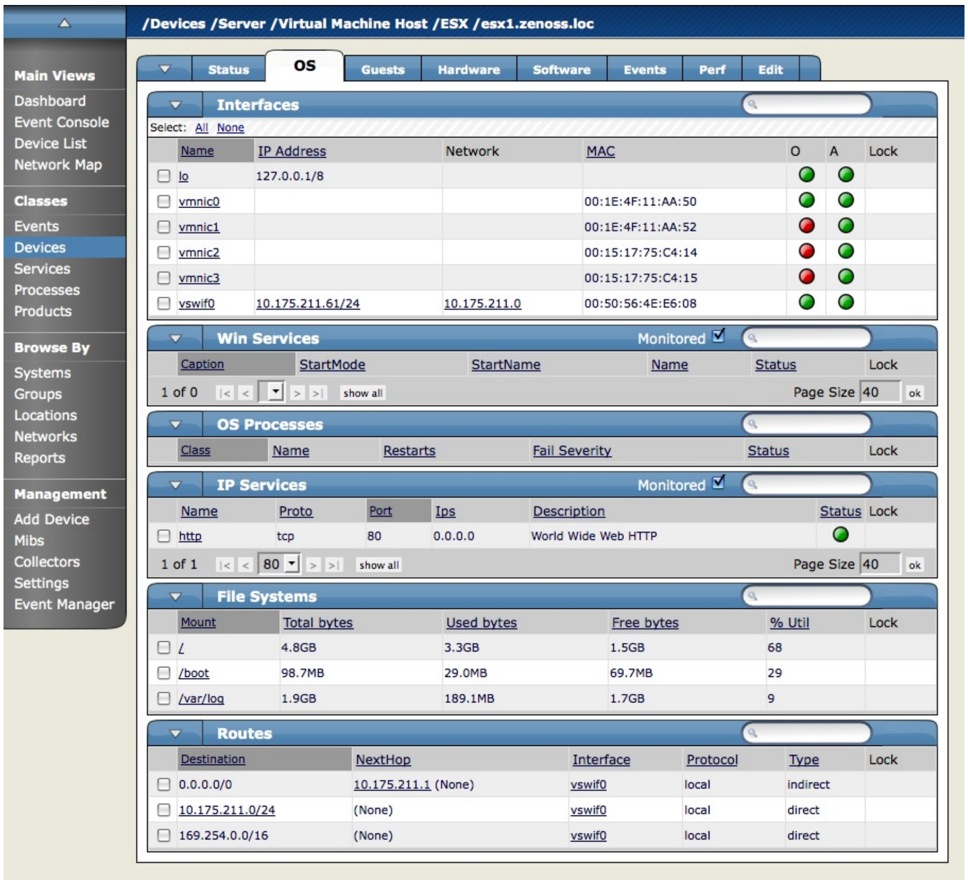
   There is also a Device Information table that shows device details. The left side of the Device Information table has the assigned organizers such as Location, Device Groups, Systems as well as the assigned status monitors and performance monitor. You can click on a organizer or monitor to view the status page for the organizer or monitor. The right side of the Device Information table includes data about the hardware and operating system of the device.

8.2. OS (Operating Systems) Tab

The OS tab contains logical Operating System components such as:

- Interfaces
- IP Services
- Win Services (for Windows devices)
- File Systems
- Routes
- OS Processes

Figure 10.8. Individual Device - OS Tab



/Devices /Server /Virtual Machine Host /ESX /esx1.zenoss.loc

Main Views: Dashboard, Event Console, Device List, Network Map

Classes: Events, Devices, Services, Processes, Products

Browse By: Systems, Groups, Locations, Networks, Reports

Management: Add Device, Mibs, Collectors, Settings, Event Manager

Tabs: Status, OS, Guests, Hardware, Software, Events, Perf, Edit

Interfaces

Select: All None

Name	IP Address	Network	MAC	O	A	Lock
lo	127.0.0.1/8			🟢	🟢	
vmnic0			00:1E:4F:11:AA:50	🟢	🟢	
vmnic1			00:1E:4F:11:AA:52	🔴	🟢	
vmnic2			00:15:17:75:C4:14	🔴	🟢	
vmnic3			00:15:17:75:C4:15	🔴	🟢	
vswif0	10.175.211.61/24	10.175.211.0	00:50:56:4E:E6:08	🟢	🟢	

Win Services Monitored ✓

Caption	StartMode	StartName	Name	Status	Lock

1 of 0 show all Page Size 40 ok

OS Processes

Class	Name	Restarts	Fail Severity	Status	Lock

IP Services Monitored ✓

Name	Proto	Port	Ips	Description	Status	Lock
http	tcp	80	0.0.0.0	World Wide Web HTTP	🟢	

1 of 1 80 show all Page Size 40 ok

File Systems

Mount	Total bytes	Used bytes	Free bytes	% Util	Lock
/	4.8GB	3.3GB	1.5GB	68	
/boot	98.7MB	29.0MB	69.7MB	29	
/var/log	1.9GB	189.1MB	1.7GB	9	

Routes

Destination	NextHop	Interface	Protocol	Type	Lock
0.0.0.0/0	10.175.211.1 (None)	vswif0	local	indirect	
10.175.211.0/24	(None)	vswif0	local	direct	
169.254.0.0/16	(None)	vswif0	local	direct	

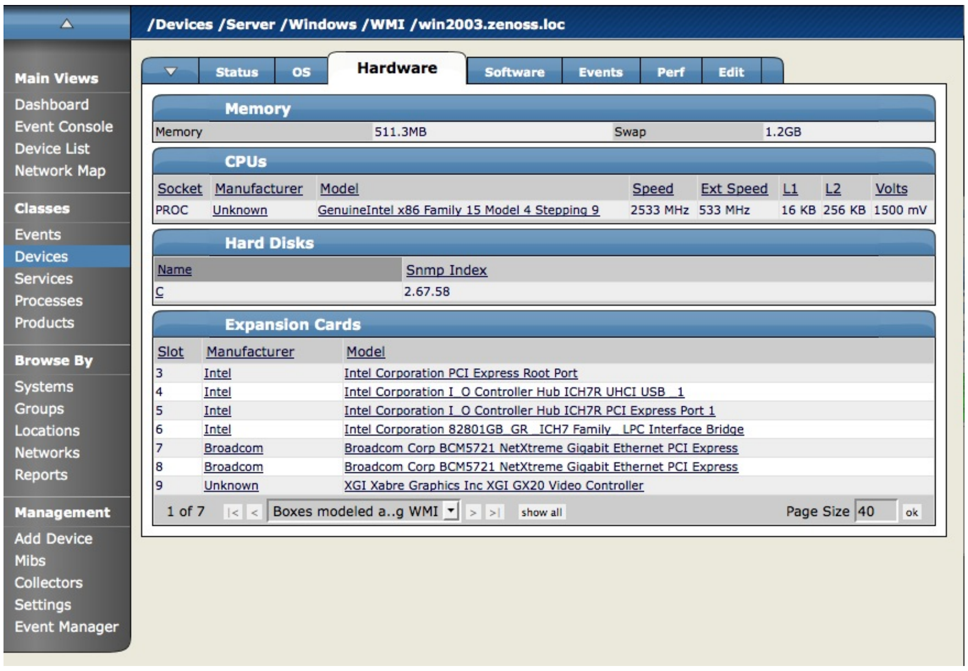
8.2.1. File System Monitoring

File system monitoring is only usable if the system has a good HOST-RESOURCES MIB. File systems monitor the amount of blocks used and show total bytes, available bytes, used bytes and percent used. You can set thresholds to alert when a specific event occurs such as the disk reaches 90% utilization.

8.3. Hardware Tab

The Hardware tab gives you information about the device’s available/used memory, available/used swap and information on the CPU(s).

Figure 10.9. Individual Device Hardware Tab



Memory

Memory	511.3MB	Swap	1.2GB

CPUs

Socket	Manufacturer	Model	Speed	Ext Speed	L1	L2	Volts
PROC	Unknown	GenuineIntel x86 Family 15 Model 4 Stepping 9	2533 MHz	533 MHz	16 KB	256 KB	1500 mV

Hard Disks

Name	Snmp Index
C	2.67.58

Expansion Cards

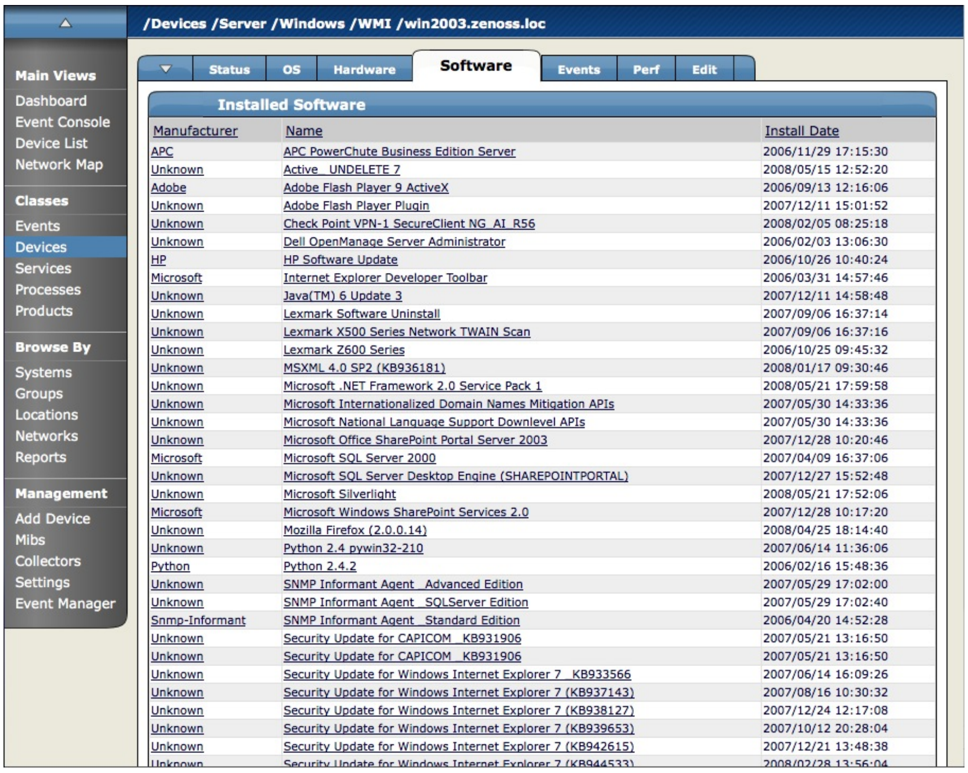
Slot	Manufacturer	Model
3	Intel	Intel Corporation PCI Express Root Port
4	Intel	Intel Corporation I_O Controller Hub ICH7R UHCI USB _1
5	Intel	Intel Corporation I_O Controller Hub ICH7R PCI Express Port 1
6	Intel	Intel Corporation 82801GB_GR _ICH7 Family _LPC Interface Bridge
7	Broadcom	Broadcom Corp BCM5721 NetXtreme Gigabit Ethernet PCI Express
8	Broadcom	Broadcom Corp BCM5721 NetXtreme Gigabit Ethernet PCI Express
9	Unknown	XGI Xabre Graphics Inc XGI GX20 Video Controller

1 of 7 |< < Boxes modeled a..g WMI > >| show all Page Size 40 ok

8.4. Software Tab

The Software tab gives a list of all installed software. This software can be sorted by Manufacturer, Name or Install Date.

Figure 10.10. Individual Device - Software Tab



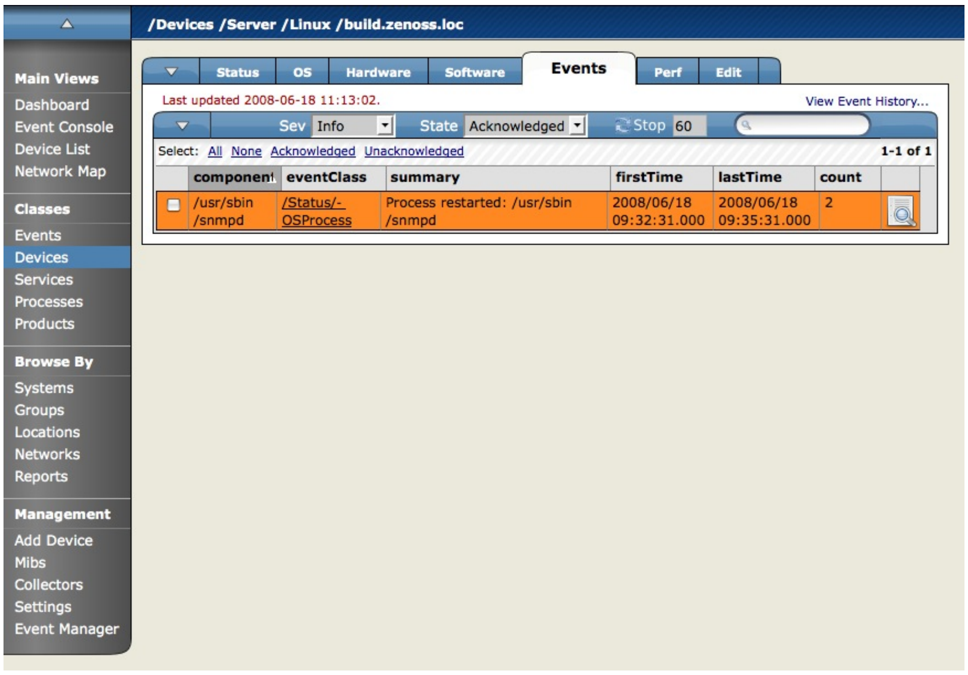
Installed Software

Manufacturer	Name	Install Date
APC	APC PowerChute Business Edition Server	2006/11/29 17:15:30
Unknown	Active_ UNDELETE 7	2008/05/15 12:52:20
Adobe	Adobe Flash Player 9 ActiveX	2006/09/13 12:16:06
Unknown	Adobe Flash Player Plugin	2007/12/11 15:01:52
Unknown	Check Point VPN-1 SecureClient NG_AI_R56	2008/02/05 08:25:18
Unknown	Dell OpenManage Server Administrator	2006/02/03 13:06:30
HP	HP Software Update	2006/10/26 10:40:24
Microsoft	Internet Explorer Developer Toolbar	2006/03/31 14:57:46
Unknown	Java(TM) 6 Update 3	2007/12/11 14:58:48
Unknown	Lexmark Software Uninstall	2007/09/06 16:37:14
Unknown	Lexmark X500 Series Network TWAIN Scan	2007/09/06 16:37:16
Unknown	Lexmark Z600 Series	2006/10/25 09:45:32
Unknown	MSXML 4.0 SP2 (KB936181)	2008/01/17 09:30:46
Unknown	Microsoft .NET Framework 2.0 Service Pack 1	2008/05/21 17:59:58
Unknown	Microsoft Internationalized Domain Names Mitigation APIs	2007/05/30 14:33:36
Unknown	Microsoft National Language Support Downlevel APIs	2007/05/30 14:33:36
Unknown	Microsoft Office SharePoint Portal Server 2003	2007/12/28 10:20:46
Microsoft	Microsoft SQL Server 2000	2007/04/09 16:37:06
Unknown	Microsoft SQL Server Desktop Engine (SHAREPOINTPORTAL)	2007/12/27 15:52:48
Unknown	Microsoft Silverlight	2008/05/21 17:52:06
Microsoft	Microsoft Windows SharePoint Services 2.0	2007/12/28 10:17:20
Unknown	Mozilla Firefox (2.0.0.14)	2008/04/25 18:14:40
Unknown	Python 2.4 pywin32-210	2007/06/14 11:36:06
Python	Python 2.4.2	2006/02/16 15:48:36
Unknown	SNMP Informant Agent _Advanced Edition	2007/05/29 17:02:00
Unknown	SNMP Informant Agent _SQLServer Edition	2007/05/29 17:02:40
Snmp-Informant	SNMP Informant Agent _Standard Edition	2006/04/20 14:52:28
Unknown	Security Update for CAPICOM _KB931906	2007/05/21 13:16:50
Unknown	Security Update for CAPICOM _KB931906	2007/05/21 13:16:50
Unknown	Security Update for Windows Internet Explorer 7 _KB933566	2007/06/14 16:09:26
Unknown	Security Update for Windows Internet Explorer 7 (KB937143)	2007/08/16 10:30:32
Unknown	Security Update for Windows Internet Explorer 7 (KB938127)	2007/12/24 12:17:08
Unknown	Security Update for Windows Internet Explorer 7 (KB939653)	2007/10/12 20:28:04
Unknown	Security Update for Windows Internet Explorer 7 (KB942615)	2007/12/21 13:48:38
Unknown	Security Update for Windows Internet Explorer 7 (KB944533)	2008/02/28 13:56:04

8.5. Events Tab

The Events tab is very similar to other event views. You can sort events using several items including, but not limited to, Component, EventClass and Count. You can also filter events by Severity, State or a regular expression applied over all events. You can also can move, map or acknowledge Events here.

Figure 10.11. Individual Device - Events Tab



/Devices /Server /Linux /build.zenoss.loc

Last updated 2008-06-18 11:13:02.

Sev Info State Acknowledged Stop 60

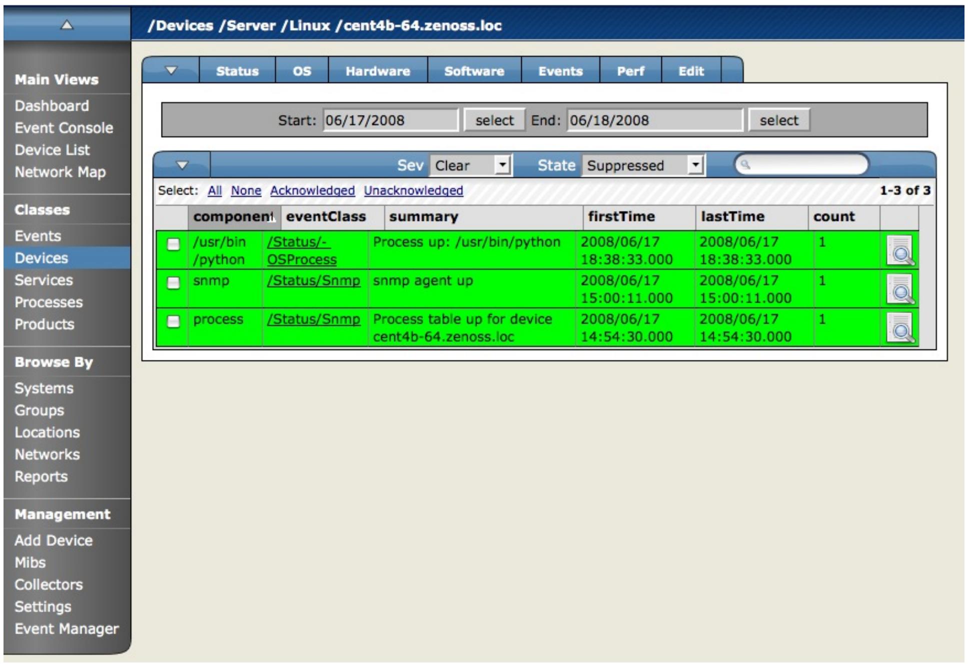
1-1 of 1

	component	eventClass	summary	firstTime	lastTime	count
☐	/usr/sbin /snmpd	/Status/- OSProcess	Process restarted: /usr/sbin /snmpd	2008/06/18 09:32:31.000	2008/06/18 09:35:31.000	2

8.6. History Tab

The History tab shows the events that have finished with the event life cycle by some means and have been archived in the history.

Figure 10.12. Individual Device - Status Tab



/Devices /Server /Linux /cent4b-64.zenoss.loc

Start: 06/17/2008 End: 06/18/2008

Sev Clear State Suppressed

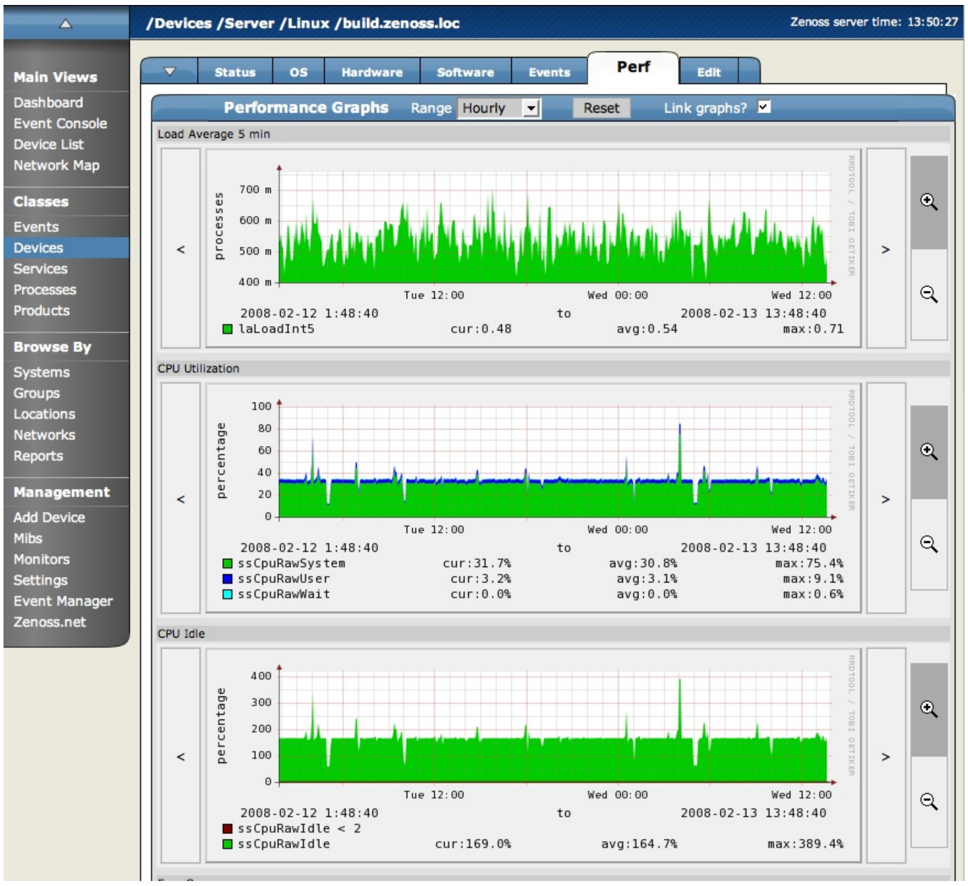
1-3 of 3

	component	eventClass	summary	firstTime	lastTime	count
☐	/usr/bin /python	/Status/- OSProcess	Process up: /usr/bin/python	2008/06/17 18:38:33.000	2008/06/17 18:38:33.000	1
☐	snmp	/Status/Snmp	snmp agent up	2008/06/17 15:00:11.000	2008/06/17 15:00:11.000	1
☐	process	/Status/Snmp	Process table up for device cent4b-64.zenoss.loc	2008/06/17 14:54:30.000	2008/06/17 14:54:30.000	1

8.7. Performance Tab

The Perf tab shows performance graphs for the currently selected device. It displays Load Average 5 Min, CPU Utilization, CPU Idle, Free Memory and Free Swap. This can be changed to update hourly, daily, weekly, monthly or yearly.

Figure 10.13. Individual Device Performance tab



/Devices /Server /Linux /build.zenoss.loc

Zenoss server time: 13:50:27

Range: Hourly

Link graphs? ✓

Load Average 5 min

```
2008-02-12 1:48:40              to          2008-02-13 13:48:40
🟩 laLoadInt5          cur:0.48       avg:0.54          max:0.71
```

CPU Utilization

```
2008-02-12 1:48:40              to          2008-02-13 13:48:40
🟩 ssCpuRawSystem     cur:31.7%      avg:30.8%         max:75.4%
🟦 ssCpuRawUser       cur:3.2%       avg:3.1%          max:9.1%
🟦 ssCpuRawWait       cur:0.0%       avg:0.0%          max:0.6%
```

CPU Idle

```
2008-02-12 1:48:40              to          2008-02-13 13:48:40
🟥 ssCpuRawIdle < 2
🟩 ssCpuRawIdle       cur:169.0%     avg:164.7%        max:389.4%
```

8.7.1. Graph Surfing

You can use the controls on the sides of the graph to “graph surf”. Graph surfing is when you use the controls on the sides of the graph to change the graph and look at it in different ways. The arrows and magnifying glasses allow you to see the data in different ways.

8.7.1.1. Linking the graphs together

The default is that all the graphs move in sync with one another so that if you click the back arrow for any graph, they will all move backward, however you can keep it so that the arrow for each graph only works for that particular graph by un-checking the Link Graphs box.

8.7.1.2. Resetting the View

The Reset button returns to the default (initial view) of the graphs. Scrolling data back and forth horizontally. To scroll data on the horizontally, use the arrows on either side of each graph. Each click will scroll the next unit back and once you have scrolled data back, you can also use the arrows to scroll the graphs forward.

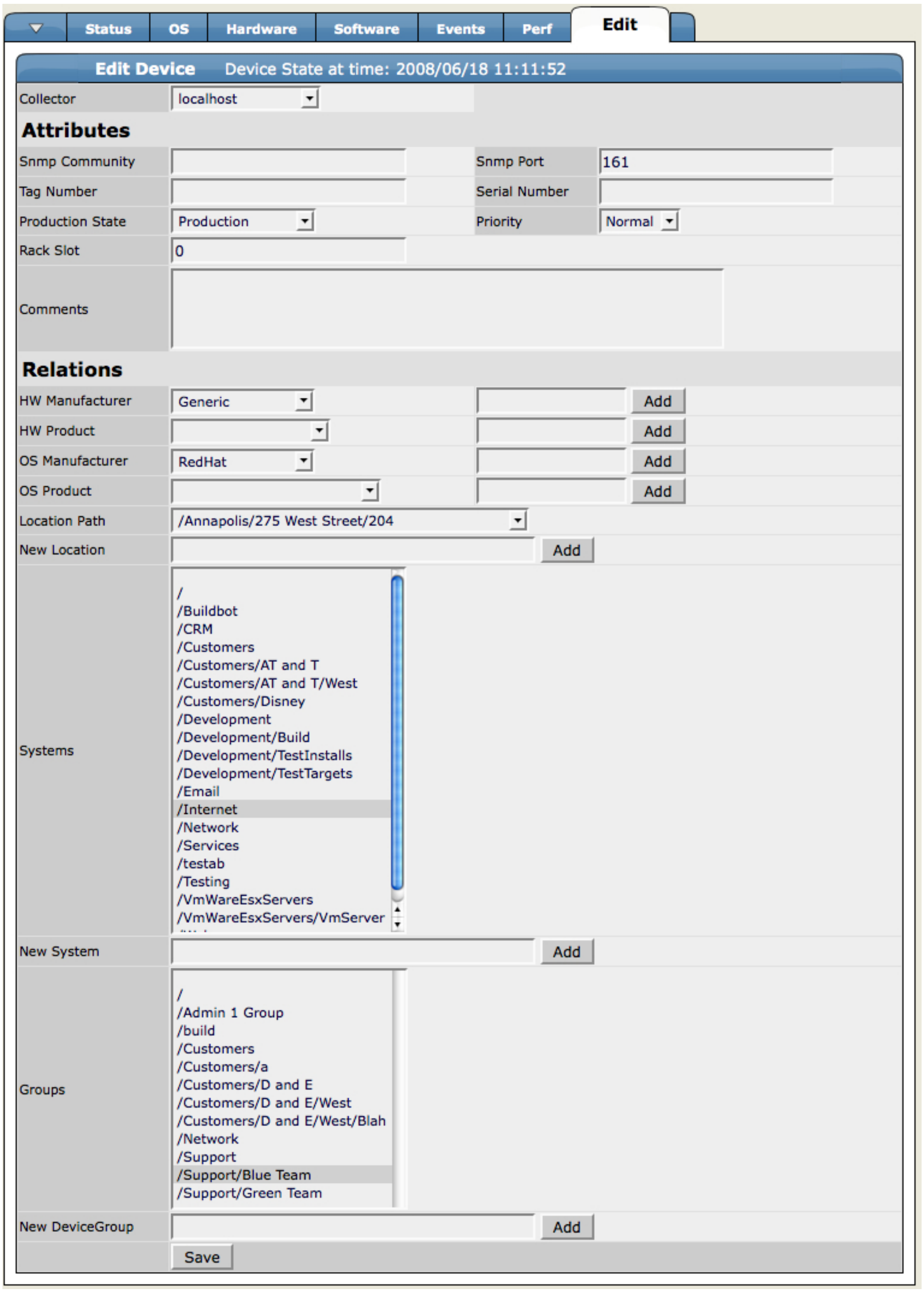
8.7.1.3. Zooming In on Data

You can also zoom in and out on the data that appears on graph by clicking the enlarge (+) magnifying glass or zooming out by clicking on the (-) magnifying glass. You can use any combination of scrolling and zooming to see the data exactly how you want to see it.

8.8. Edit Tab

Use this tab to change any of the properties of the device.

Figure 10.14. Individual Device - Edit Tab



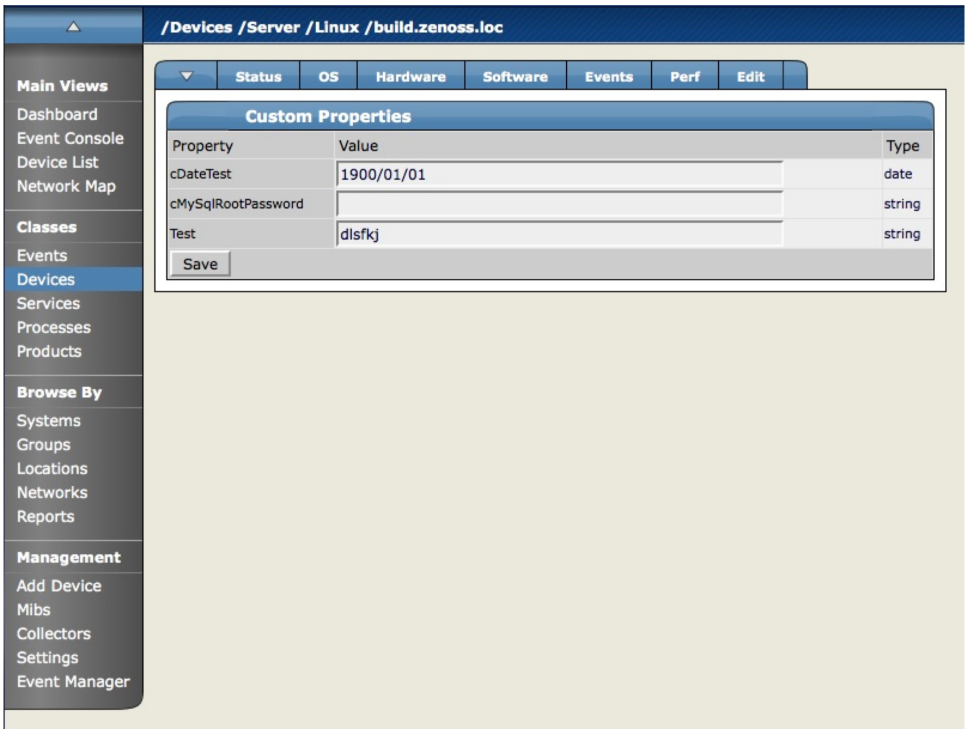
Edit Device Device State at time: 2008/06/18 11:11:52

Collector	localhost		
Attributes			
Snmp Community		Snmp Port	161
Tag Number		Serial Number	
Production State	Production	Priority	Normal
Rack Slot	0		
Comments			
Relations			
HW Manufacturer	Generic		Add
HW Product			Add
OS Manufacturer	RedHat		Add
OS Product			Add
Location Path	/Annapolis/275 West Street/204		
New Location		Add	
Systems	/ /Buildbot /CRM /Customers /Customers/AT and T /Customers/AT and T/West /Customers/Disney /Development /Development/Build /Development/TestInstalls /Development/TestTargets /Email /Internet /Network /Services /testab /Testing /VmWareEsxServers /VmWareEsxServers/VmServer		
New System		Add	
Groups	/ /Admin 1 Group /build /Customers /Customers/a /Customers/D and E /Customers/D and E/West /Customers/D and E/West/Blah /Network /Support /Support/Blue Team /Support/Green Team		
New DeviceGroup		Add	
	Save		

8.9. Custom Tab

The Custom tab allows you to set values in the custom fields you define in Custom Schema. To access the Custom tab, open the Device page menu, select More and then Custom.

Figure 10.15. Individual Device Custom Tab

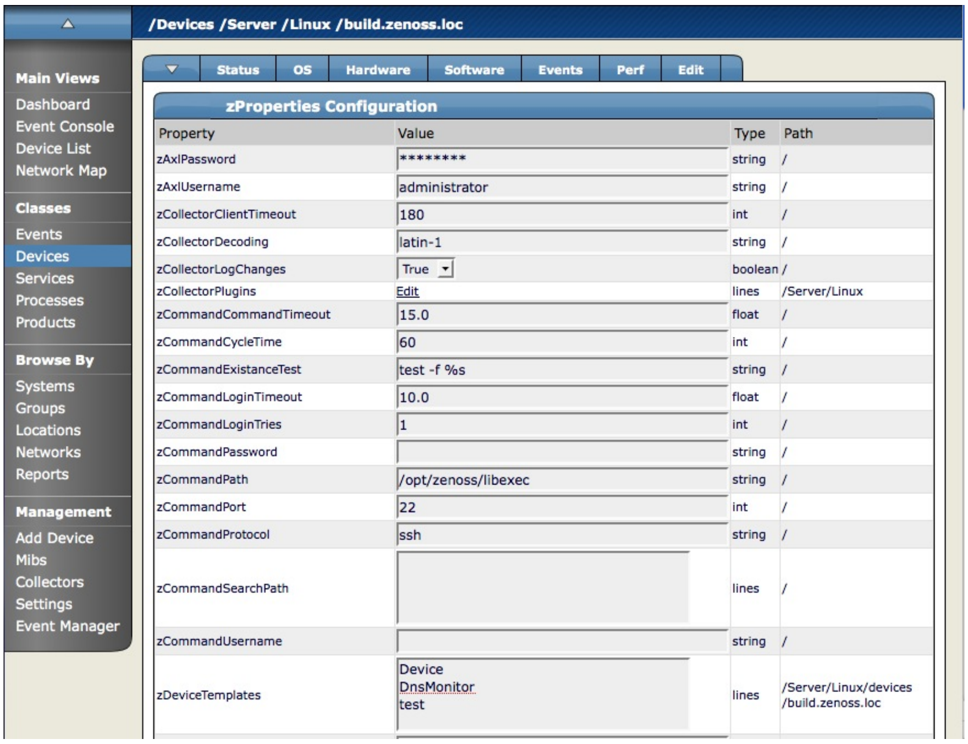


Property	Value	Type
cDateTest	1900/01/01	date
cMySqlRootPassword		string
Test	dlsfkj	string

8.10. zProperties Tab

The Device zProperties tab is where you can configure zProperties either for all devices, for an individual device, or for any devices that fall further down in the device hierarchy tree. To configure them for all devices click the zProperties tab from the Device Overview tab. To configure zProperties for an individual device, click on the device name in the Device Overview and then click the zProperties tab for that device. To access the zProperties tab, open the device table menu, select More and then zProperties.

Figure 10.16. Individual Device - zProperties Tab

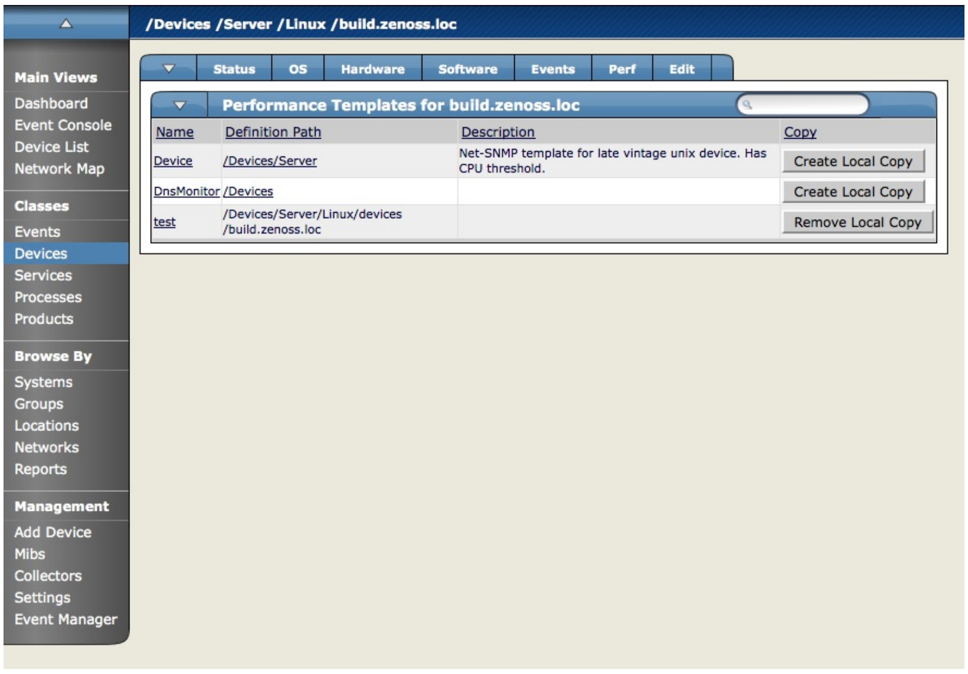


Property	Value	Type	Path
zAxlPassword	********	string	/
zAxlUsername	administrator	string	/
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/Server/Linux
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		string	/
zCommandPath	/opt/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device DnsMonitor test	lines	/Server/Linux/devices /build.zenoss.loc

8.11. Templates Tab

The template tab shows all of the performance templates bound by name to this device or group of devices. To access the Templates tab, open the Device table menu, select More and then Templates.

Figure 10.17. Individual Device - Templates Tab

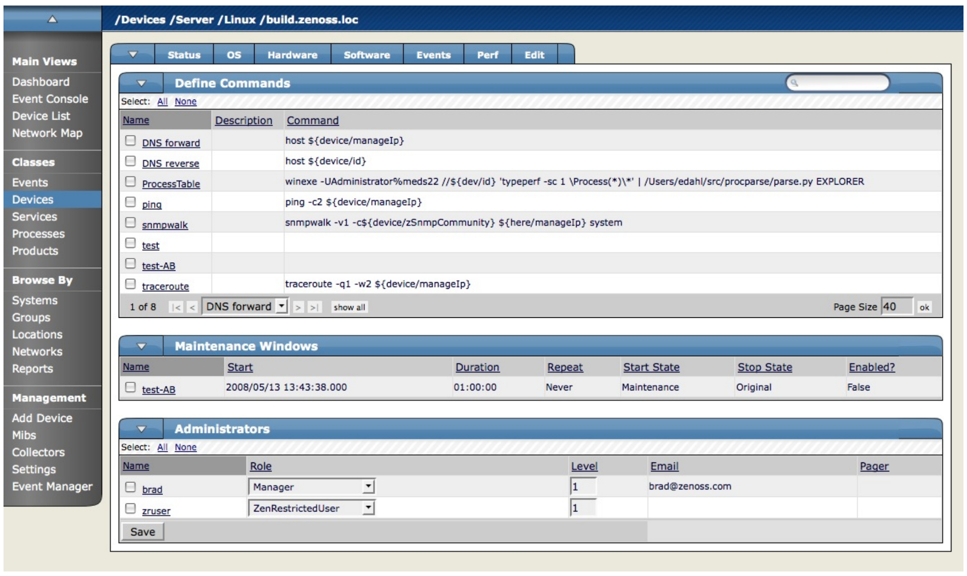


Name	Definition Path	Description	Copy
Device	/Devices/Server	Net-SNMP template for late vintage unix device. Has CPU threshold.	Create Local Copy
DnsMonitor	/Devices		Create Local Copy
test	/Devices/Server/Linux/devices /build.zenoss.loc		Remove Local Copy

8.12. Administration Tab

To access the Administration tab, open the Device page menu, select More and then Administration.

Figure 10.18. Individual Device Administration Tab



/Devices /Server /Linux /build.zenoss.loc

Define Commands

Select: All None

Name	Description	Command
DNS forward		host ${device/manageIp}
DNS reverse		host ${device/id}
ProcessTable		winexe -UAdministrator%meds22 //${dev/id} 'typeperf -sc 1 \Process(*)*' | /Users/edahl/src/procparse/parse.py EXPLORER
ping		ping -c2 ${device/manageIp}
snmpwalk		snmpwalk -v1 -c${device/zSnmpCommunity} ${here/manageIp} system
test		
test-AB		
traceroute		traceroute -q1 -w2 ${device/manageIp}

1 of 8 DNS forward show all Page Size 40 ok

Maintenance Windows

Name	Start	Duration	Repeat	Start State	Stop State	Enabled?
test-AB	2008/05/13 13:43:38.000	01:00:00	Never	Maintenance	Original	False

Administrators

Select: All None

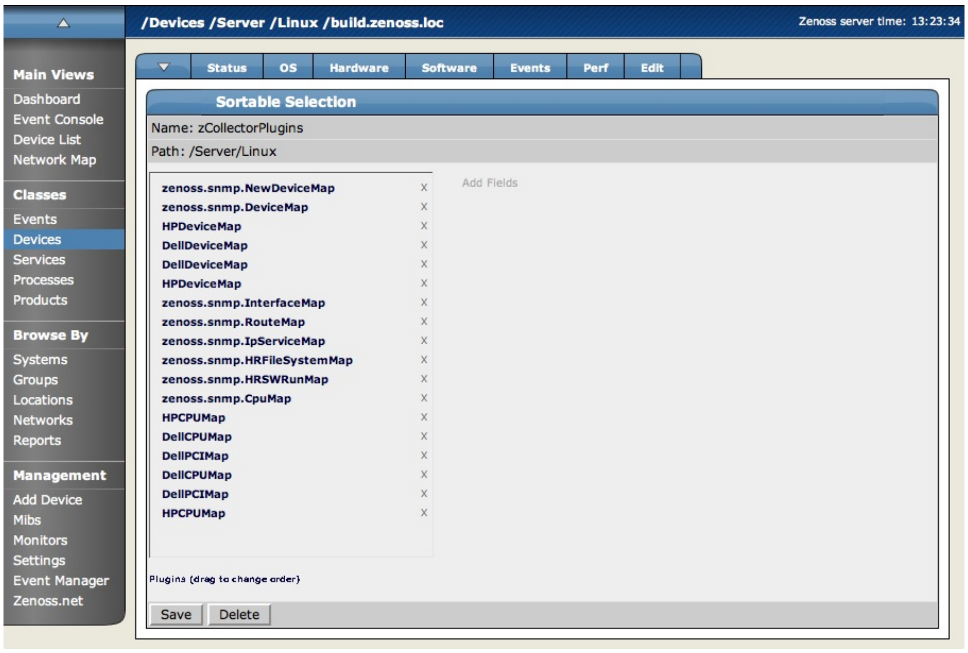
Name	Role	Level	Email	Pager
brad	Manager	1	brad@zenoss.com	
zruser	ZenRestrictedUser	1		

Save

8.13. Collector Plugins tab

The Collector Plugins tab shows you a list of all plugins available for the device. To access the Collector Plugins tab, open the Device page menu, select More and then Collector Plugins.

Figure 10.19. Individual Device - Collector Plugins Tab



/Devices /Server /Linux /build.zenoss.loc

Zenoss server time: 13:23:34

Sortable Selection

Name: zCollectorPlugins

Path: /Server/Linux

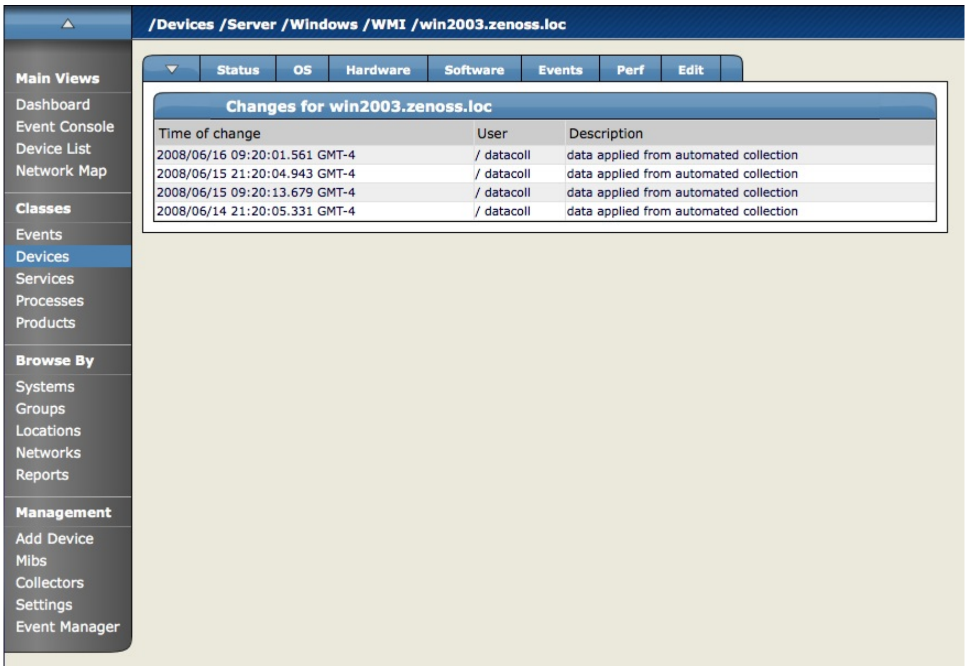
zenoss.snmp.NewDeviceMap x
zenoss.snmp.DeviceMap x
HPDeviceMap x
DellDeviceMap x
DellDeviceMap x
HPDeviceMap x
zenoss.snmp.InterfaceMap x
zenoss.snmp.RouteMap x
zenoss.snmp.IpServiceMap x
zenoss.snmp.HRFileSystemMap x
zenoss.snmp.HRSWRunMap x
zenoss.snmp.CpuMap x
HPCPUMap x
DellCPUMap x
DellPCIMap x
DellCPUMap x
DellPCIMap x
HPCPUMap x

Plugins (drag to change order)

8.14. Modifications Tab

The Modifications tab logs user changes via the Zope interface. To access the Modifications tab, open the Device page menu, select More and then Modifications.

Figure 10.20. Individual Device - Modifications Tab



Changes for win2003.zenoss.loc

Time of change	User	Description
2008/06/16 09:20:01.561 GMT-4	/ datacoll	data applied from automated collection
2008/06/15 21:20:04.943 GMT-4	/ datacoll	data applied from automated collection
2008/06/15 09:20:13.679 GMT-4	/ datacoll	data applied from automated collection
2008/06/14 21:20:05.331 GMT-4	/ datacoll	data applied from automated collection

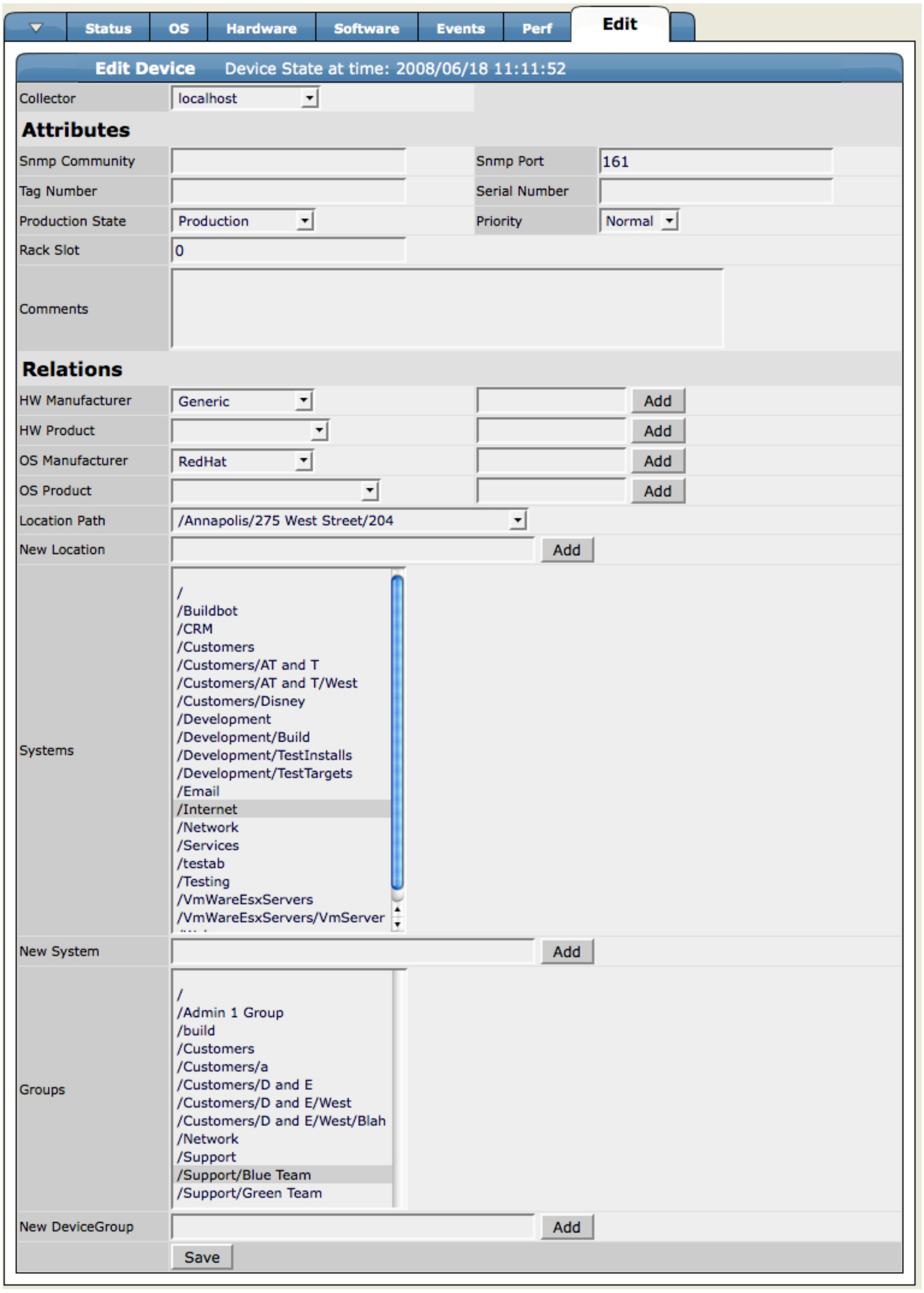
9. Searching For a Device By Name or IP Address

The first and most prominent way to search for a device is to type in the name or IP of the device in the “Device/IP Search” field at the top of the application. However this is not very useful if you cannot remember the name of the device or are not looking for a specific device. In this case you can use the “Browse by” section of the left navigation menu. Browse by allows you to browse by Systems, Groups, Locations and Reports. Browsing by Systems allows you to browse by some common types of Devices such as File Servers, Printers and Infrastructure and are arranged hierarchically. Browsing by Groups allows you to browse by groups that you can set up to organize your devices. These groups can be named whatever you like and are arranged hierarchically. Browsing by Locations is basically the same as the others, is just allows you to arrange your devices into groups based on location and are arranged hierarchically as well.

10. Editing Device Configurations

If you wish to edit the configuration of a device, first locate the device either by using the “Browse By” menu or by searching for the device in the search box at the top of the application. You can then edit the device configuration by selecting the “Edit” tab of the device and changing any of the information. The following figure shows the Edit tab for a Linux server.

Figure 10.21. Device - Edit Tab



Edit Device Device State at time: 2008/06/18 11:11:52

Collector	localhost		
Attributes			
Snmp Community		Snmp Port	161
Tag Number		Serial Number	
Production State	Production	Priority	Normal
Rack Slot	0		
Comments			
Relations			
HW Manufacturer	Generic		Add
HW Product			Add
OS Manufacturer	RedHat		Add
OS Product			Add
Location Path	/Annapolis/275 West Street/204		
New Location		Add	
Systems	/ /Buildbot /CRM /Customers /Customers/AT and T /Customers/AT and T/West /Customers/Disney /Development /Development/Build /Development/TestInstalls /Development/TestTargets /Email /Internet /Network /Services /testab /Testing /VmWareEsxServers /VmWareEsxServers/VmServer		
New System		Add	
Groups	/ /Admin 1 Group /build /Customers /Customers/a /Customers/D and E /Customers/D and E/West /Customers/D and E/West/Blah /Network /Support /Support/Blue Team /Support/Green Team		
New DeviceGroup		Add	
	Save		

11. Managing Devices

To manage various device attributes you can use the Manage menu item in the page menu for each device. The options for this menu are More... Manage... and Run Commands... Use the Manage item to make all of the changes described in the section below. It is also important to note that you can set up many of these management activities according to the inheritance and hierarchies of the various device organizers. Just navigate to the organizer (Class, System, Group or Location) and follow the same steps. Exceptions to this inheritance are noted in each section below.

11.1. Remodeling a Device

To make Zenoss got to a device and recollect all of the configuration information for the device:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then Model Device.

   The Device is remodeled and the remodeling status page appears.

11.2. Changing the Device Class of a Device

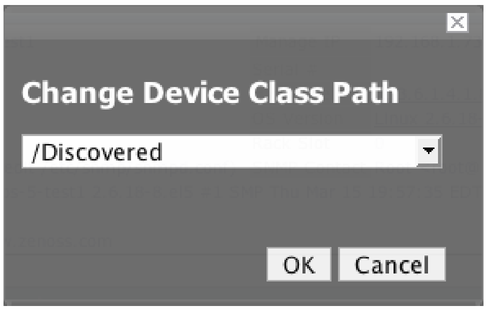
To change the Device Class for a device:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then Change Class.

   The Device Class Change dialog appears.

   Figure 10.22. Change Device Class Path Dialog

3. Select the new Device class for this device from the drop-down menu.

4. Click OK.

   The device class for the current device has been changed.

11.3. Resetting Device Manage IP

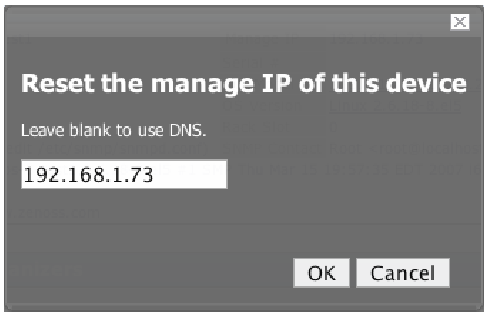
To Reset the manage IP of a Device in the system:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then Reset IP.

   The Rest IP dialog appears.

Figure 10.23. Reset IP Dialog



Reset the manage IP of this device

Leave blank to use DNS.

192.168.1.73

3. Enter the new IP for the device (or leave it blank to allow the IP to be set by DNS).

4. Click OK.

   The IP for the device is reset.

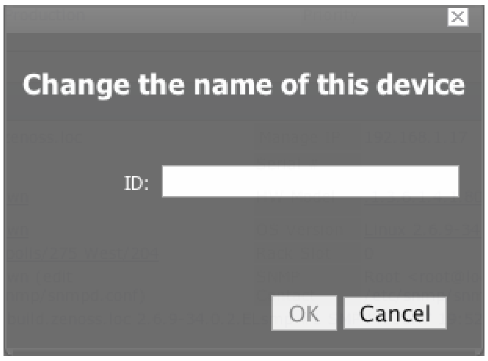
11.4. Renaming a Device

To rename a device that is in the Zenoss system:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then Rename Device.

Figure 10.24. Rename Device Dialog



Change the name of this device

ID:

3. In the ID field, enter the new name for the device.

4. Click OK.

   The Device is renamed in the system.

11.5. Locking Device Configurations

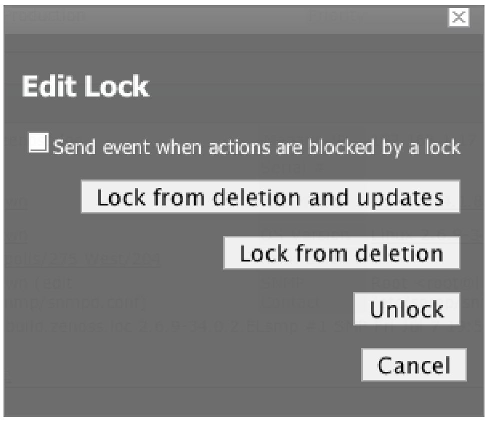
You can lock a device's configuration in a few ways to prevent changes from being overwritten when remodeling the device. There are two levels of locking that are available. You can Lock the configuration from deletion and updates, or lock from deletion. You can also use this dialog to Unlock the device configuration.

1. Navigate to the Device.

2. From the Device page menu, select Manage and then Lock.

   The Lock Configuration Confirmation Dialog appears.

   Figure 10.25. Configuration Lock Editing Dialog

   
   Edit Lock
Send event when actions are blocked by a lock
Lock from deletion and updates
Lock from deletion
Unlock
Cancel

3. If you want to send events when actions are blocked by a lock click the checkbox.

4. Change the locking configuration by clicking the button for the lock configuration you want to set.

   The lock is set for the device.

11.6. Resetting the Device Community

To change the Device community string:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then select Reset Community.

   The community for the device is reset.

11.7. Pushing Configuration Changes Back to the Zenoss System

You can push the any changes you make to the device configuration to the Zenoss system and the associated collectors, without waiting for a remodel.

To push the changes:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then select Push Changes.

   A status message appears in the upper right of the screen confirming that the changes have been pushed up to the collectors.

11.8. Clearing Heartbeats

To clear the heartbeats associated with a particular device:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then select Clear Heartbeats.

   The heartbeats for this device are moved to the event history.

   The Edit tab for the ZenEventManager appears. Make any changes you want to make (if necessary)and click Save.

11.9. Deleting Devices From the System

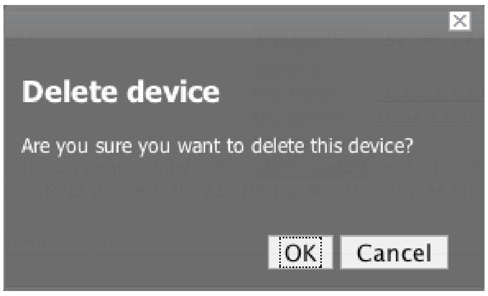
To delete a device from the Zenoss system:

1. Navigate to the Device.

2. From the Device page menu, select Manage and then select Delete Device.

   The Delete Device Confirmation dialog appears.

   Figure 10.26. Delete Device Confirmation Dialog

3. Click OK.

   The Device is removed from the system.

12. Modeling Devices Using SNMP

This section various describes the methods used to model devices using SNMP.

12.1. Testing to See if a Device is Running SNMP

To test whether or not a device is running SNMP run:

```
$ snmpwalk -v1 -c communityString gate system
```

If this command does not time out, SNMP is installed and working correctly.

12.2. Modeling Remote Windows Devices Using SNMP

By default Windows may not have any SNMP installed. To install SNMP, go to Start -> Control Panel ->Add or Remove Programs -> Add/Remove Windows Components. Check the box for Management and Monitoring tools and install them. Next, you need to turn the services on and configure them, so go to Control Panel -> Administrative Tools -> Services and start both SNMP Service and SNMP Trap Service. Set the SNMP Community string in the SNMP Service properties to the community string of your SNMP.

If you want processor and memory monitoring, install SNMP-Informant on the device. Go to www.snmp-informant.com and download SNMP for Windows.

To collect Windows Event logs or log files from a windows box using syslog you can use SyslogAgent from http://syslogserver.com/syslogagent.html.

12.3. Modeling Remote Linux Devices Using SNMP

To configure a Linux machine for monitoring, it must have SNMP installed. A good Linux SNMP application is net-snmp. Download, install and configure net-snmp and you can use SNMP to monitor Linux devices with Zenoss.

12.4. Modeling Cisco Devices Using SNMP

Cisco devices come with SNMP already installed. However, you have to configure SNMP on each Cisco device to be in the same community as the rest of your network.

13. Modeling Using SSH/COMMAND

In some situations a system may not offer an SNMP Agent that Zenoss can securely access. This is often the case when Zenoss is used to monitor production systems that only provide SSH access. Additionally, when SNMP modeling is not available (or it does not provide a complete set of information), command plugins can be used to model a device. Note: These “modeling” command plugins should not be confused with the “performance” command plugins used elsewhere in Zenoss.

Each built-in zenoss modeling command plugin is differentiated by the platform on which it runs. To determine the platform for the device you want to model, run the “uname” command in a shell on the device. The compatibility options for the Zenoss modeling plugins are currently Linux or Darwin. The modeling command plugins can be extended for platforms not listed above. However, the procedure for creating new modeling command plugins is beyond the scope of the Administration Guide.

To model a device using command plugins, first you must add the device to zenoss using the protocol "none" and then choose which plugins you wish to apply.

1. Go to the Add Device page.
2. Set discover to “None”.
3. Once the device is added, navigate to the device and view its zProperties tab.
4. If necessary, set zCommandUsername and zCommandPassword to the username and password of the device (or set up passwordless authentication using RSA/DSA keys.)
5. Click “Edit” next to zCollectorPlugins.
6. Click Add Fields on the right to get a complete list of command plugins.
7. Make sure zenoss.cmd.uname is positioned first on the left hand side.
8. Click the X next to plugins you wish to remove, and drag other plugins over to the left.
9. Remodel the device.

13.1. Using Device Class to Monitor Devices Using SSH

The /Server/Cmd device class is an example configuration for modeling and monitoring devices using SSH. The CollectorPlugins have been modified as described above, and the Device, Filesystem, and Ethernet interface templates used to gather data over SSH have been created. You can use this device class as a reference for your own configuration; or, if you have a device that needs to be modeled or monitored via SSH/Command, you can place it in this device class to use the preconfigured templates and zProperties. You will still need to set the zCommandUsername and zCommandPassword zProperties to the appropriate SSH login information for each device.

14. Modeling Devices Using PortScan

You can also do a modeling of IP services by doing a port scan. To model the device using a port scan:

1. Go to the zProperties tab of a device.

2. Change the zTransportPreference to “portscan”.

3. Remodel the device.

14.1. Using the /Server/Scan Device Class to Monitor with Portscan

The /Server/Scan device class is an example configuration for modeling devices using a port scan. You can use this device class as a reference for your own configuration; or, if you have a device that will use only a port scan, you can place it under this device class and remodel the device.

15. Modeling Plugins

Zenoss uses plug-in maps to map real world information into the standard Zenoss model. Input to the plug-ins can come from SNMP, SSH or Telnet. Selection of which plug-ins to run against a device is done by matching the plug-in name against the zProperty, zCollectorPlugins. Plug-ins selected in zCollectorPlugins are the ones that are collected.

- DeviceMap – collects basic information about a device such as its OS type and hardware model.

- InterfaceMap – collects the list of interfaces on a device.

- RouteMap – collects the routing table from the device.

- IpServicesMap – collects the IP services running on the device.

- FileSystemMap – collects the list of file systems on a device.

15.1. Getting a List of Modeling Plugins for a Device

Plugins are controlled by a regular expression that matches their name. You can get a list of plugins for any device by navigating to the device and opening the page menu, selecting More and then Collector Plugins. The Collector Plugins tab appears showing all of the collector plugins for the device.

16. Debugging the Modeling Process

The modeler can be run from the command line against a single device. This can be handy if you need to debug an issue with a plugin. By passing the command " --collect" to the modeler, you can control which collection maps are used. This command will run only the interface plugin against a device called build.zenoss.loc:

```
$ zenmodeler run -v 10 --collect=IpInterface -d build.zenoss.loc
```

If it returns any stack traces, send them to our support team along with the command you ran and the version information of your Zenoss instance.

17. Using zLinks to Add Custom Links to a Device Status Page

You can add custom links within Zenoss to associate with a device, device class, or hierarchy of devices. You can use the zLinks zProperty to add links that will appear in the Status tab. These links are typically used to simplify access to other control screens and drill-down interfaces available on the device.

To add links:

1. Select a device from the Device list where you want to add a custom link.

   You can also create zLinks at the Device Class level to apply to all devices in the Class and also any devices added to the class in the future.

2. From the device page menu, select More and then zProperties.

3. Scroll down to the zLinks zProperty.

4. Enter the link in the zLinks text field in the Value column.

   You can use any URL expressions including http, telnet, file, mailto, etc. You can put other types of http markup, including image references, such as the performance graphs from other pages.

5. Click Save.

   The zLink is saved and will now appear on the Status tab for the selected device.

18. Dumping and Loading Devices Using XML Lists

Zenoss allows you to export a list of your devices to an XML file for easy importing into another Zenoss instance. From the command line, use the command:

```
zendevicedump -o mydevicelist.xml
```

This will dump the names of your devices, including their device classes, groups, systems etc. to a file called mydevicelist.xml.

To load these devices into another Zenoss instance (or reload them into the same instance), while in the Zenoss instance where you want the devices to be discovered, run the command:

```
zendeviceload -i mydevicelist.xml
```

Zenoss will then attempt to discover each of the devices in the XML file.

Chapter 11. Monitoring VMware Devices

1. Monitoring VMware Servers with Zenoss

Note

This feature is available only for Zenoss Enterprise and Professional versions.

With Zenoss, you can collect information to monitor your VMware infrastructure. By entering a single set of connection parameters, you allow Zenoss to:

- Obtain the names and properties of various entities in your VMware infrastructure
- Monitor metrics collected by VMware
- Retrieve VMware events

Zenoss extracts VMware information through the VMware Infrastructure (VI) SDK, VMware's SOAP interface to its line of server virtualization products. The SDK can be accessed from an individual ESX server or vCenter Server (previously, VirtualCenter Server) instance, which can return information about many ESX servers.

Note

For more information about VMware infrastructure, see VMware's *Introduction to VMware Infrastructure* at:

http://www.vmware.com/support/pubs/vi_pages/vi_pubs_35u2.html.

1.1. ZenVMware ZenPack

Zenoss' VMware features are implemented through the ZenVMware ZenPack. The ZenVMware ZenPack uses the 2.5 version of the VI API. It is compatible with ESX Server 3.5, VirtualCenter Server 2.5, and ESX Server 3i. It is not explicitly compatible with ESX Server 3.0.x or VirtualCenter 2.0.x, or any previous versions.

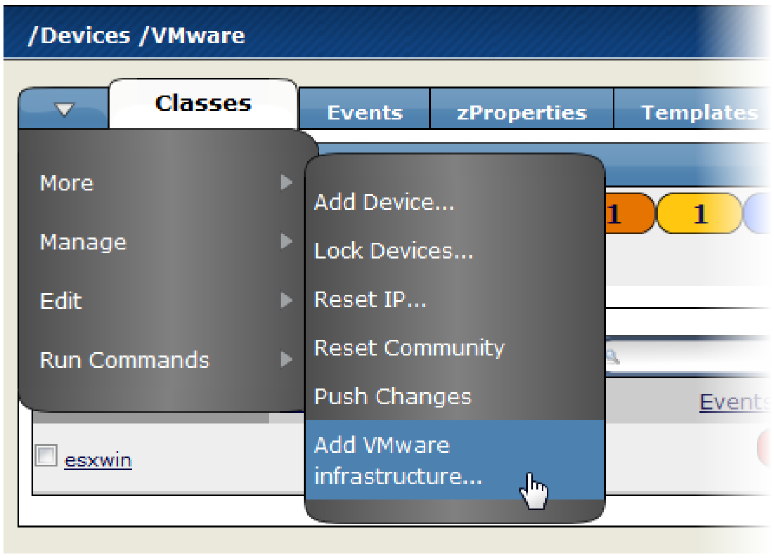
1.2. Setting Up VMware Monitoring

Follow these steps to begin monitoring your VMware servers.

1. From the Zenoss Navigation Menu, click Devices.

   The list of Devices appears.

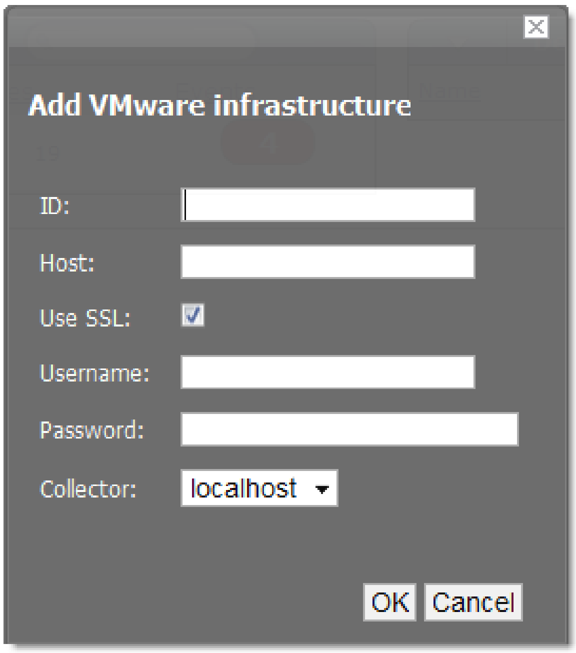
Figure 11.1. Add VMware Infrastructure



2. In the Sub-Devices list, click VMware.
3. From the Page menu, select Manage > Add VMware infrastructure.

   The Add VMware infrastructure dialog appears.

Figure 11.2. Add VMware Infrastructure Dialog



4. Enter parameters to connect to the ESX server or vCenter Server that will provide monitoring capabilities.
   - ID - Enter a name for the infrastructure to be monitored.
   - Host - Enter the hostname of the server providing the VI SDK connections. This can be an individual ESX server or the location of a vCenter Server instance.

- Use SSL - Select this option if the connection should be made by using SSL encryption.
- Username - Enter the user name used to authenticate.
- Password - Enter the password used to authenticate.
- Collector - Select the collector to use to retrieve information from the VI SDK endpoint.

5. Click OK.

   Zenoss begins modeling the VMware infrastructure. It places the information in the device hierarchy under `/Devices/VMware/ID`, where `ID` is the value of the ID field you entered during setup.

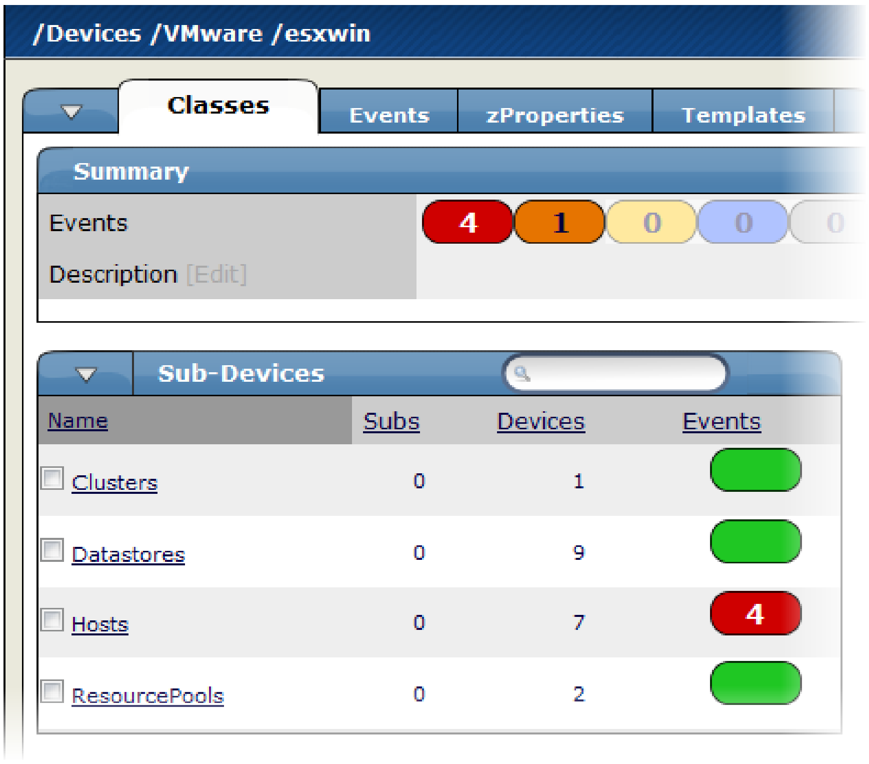
1.3. Viewing VMware Devices

Zenoss represents these VMware entities as devices:

- Hosts (ESX servers)
- Resource Pools
- Data stores
- Clusters

Each of these categories is represented as a device class under the newly created organizer. For example, if the ID of an infrastructure is esxwin, then four device classes appear below `/Devices/VMware/esxwin`: Clusters, Datastores, Hosts, and ResourcePools.

Figure 11.3. VMware Device Classes



Events	4 1 0 0 0
Description [Edit]	

Sub-Devices

Name	Subs	Devices	Events
Clusters	0	1	🟩
Datastores	0	9	🟩
Hosts	0	7	4
ResourcePools	0	2	🟩

If the SDK endpoint is an individual ESX server, then the Clusters organizer will be empty. (A VMware cluster is a concept external to an individual host.)

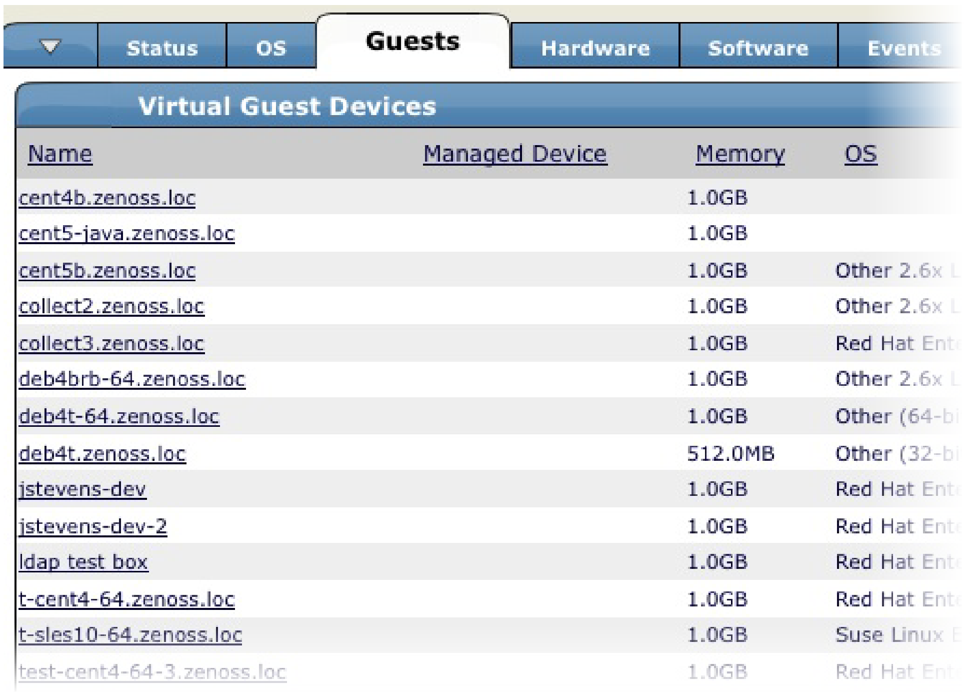
1.4. Viewing Guest Virtual Machines

To view guest VMs on an ESX server:

1. Navigate to a device in the Hosts class.
2. Click the Guests tab.

   The Virtual Guests Devices list appears.

Figure 11.4. Virtual Guest Devices



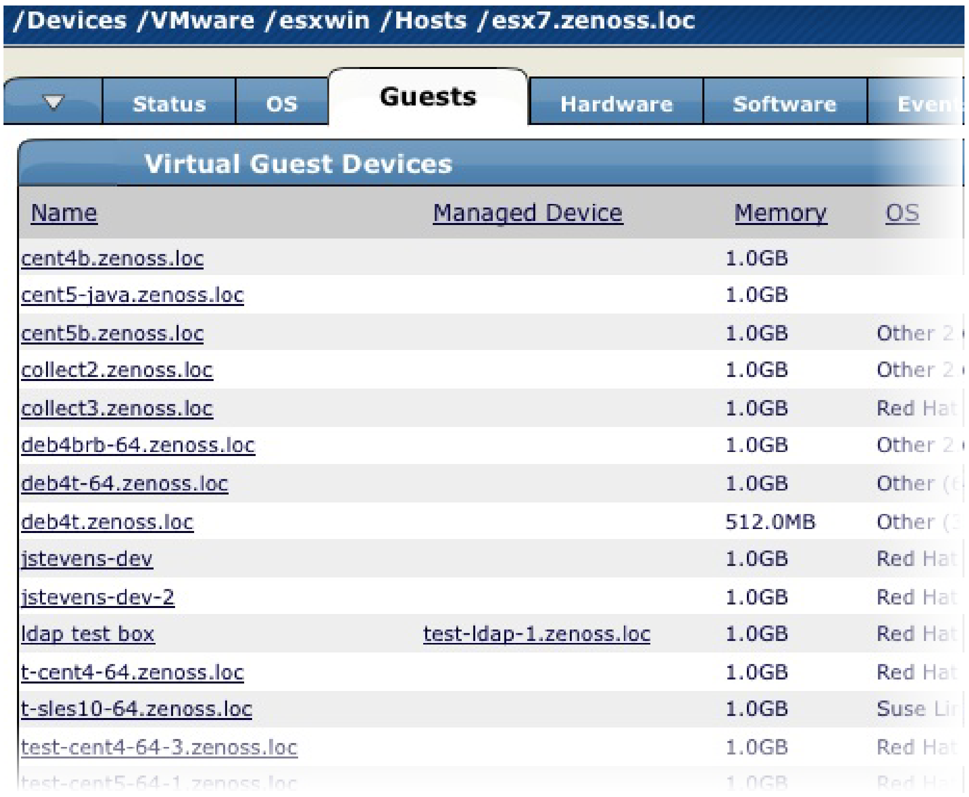
Virtual Guest Devices

Name	Managed Device	Memory	OS
cent4b.zenoss.loc		1.0GB	
cent5-java.zenoss.loc		1.0GB	
cent5b.zenoss.loc		1.0GB	Other 2.6x L
collect2.zenoss.loc		1.0GB	Other 2.6x L
collect3.zenoss.loc		1.0GB	Red Hat Ent
deb4brb-64.zenoss.loc		1.0GB	Other 2.6x L
deb4t-64.zenoss.loc		1.0GB	Other (64-bi
deb4t.zenoss.loc		512.0MB	Other (32-bi
jstevens-dev		1.0GB	Red Hat Ent
jstevens-dev-2		1.0GB	Red Hat Ent
ldap test box		1.0GB	Red Hat Ent
t-cent4-64.zenoss.loc		1.0GB	Red Hat Ent
t-sles10-64.zenoss.loc		1.0GB	Suse Linux E
test-cent4-64-3.zenoss.loc		1.0GB	Red Hat Ent

In the list, the first column contains a link to the guest component, named the same name as the VM. (This is not necessarily the same as the VM hostname.) If the VM has been modeled elsewhere in Zenoss, then a link to that device appears in the Managed Device column.

s shown in the previous figure, none of the VMs are being monitored in their "native" device classes. For example, the guest named "ldap test box" is a Linux VM with the hostname "test-ldap-1.zenoss.loc." If you add that device to /Devices/Server/Linux (by using the Add Device feature in the Left Navigation area), a link will appear.

Figure 11.5. Virtual Guest Devices - Managed Device



/Devices /VMware /esxwin /Hosts /esx7.zenoss.loc

Virtual Guest Devices

Name	Managed Device	Memory	OS
cent4b.zenoss.loc		1.0GB	
cent5-java.zenoss.loc		1.0GB	
cent5b.zenoss.loc		1.0GB	Other 2
collect2.zenoss.loc		1.0GB	Other 2
collect3.zenoss.loc		1.0GB	Red Hat
deb4brb-64.zenoss.loc		1.0GB	Other 2
deb4t-64.zenoss.loc		1.0GB	Other (
deb4t.zenoss.loc		512.0MB	Other (
jstevens-dev		1.0GB	Red Hat
jstevens-dev-2		1.0GB	Red Hat
ldap test box	test-ldap-1.zenoss.loc	1.0GB	Red Hat
t-cent4-64.zenoss.loc		1.0GB	Red Hat
t-sles10-64.zenoss.loc		1.0GB	Suse Li
test-cent4-64-3.zenoss.loc		1.0GB	Red Hat
test-cent5-64-1.zenoss.loc		1.0GB	Red Ha

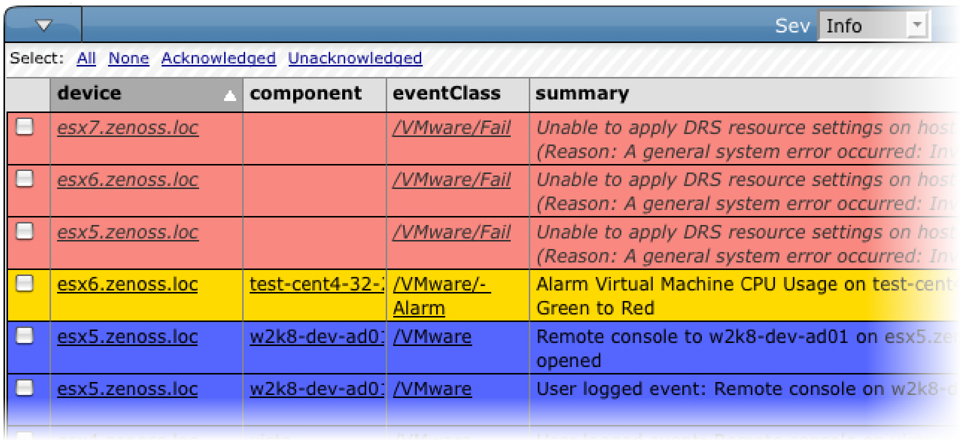
Click the Name link to go to the Guest component status page, which shows the VM's relationships to other VMware entities, and provides access to VMware-specific metrics and events.

Click the managed device link to go to the Device status page, which contains information about the device as a separate Linux or Windows server. These two status pages link to each other.

1.5. VMware Events

VMware records a wide range of events that are available through the VI SDK. Zenoss extracts these events and makes them available in the event console.

Figure 11.6. VMware Events (Event Console)



Sev Info

Select: All None Acknowledged Unacknowledged

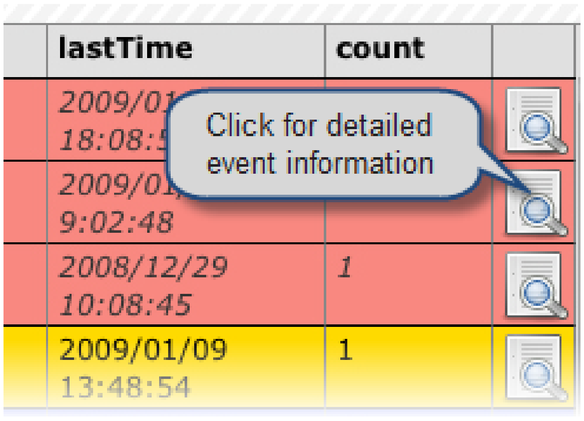
	device	component	eventClass	summary
☐	esx7.zenoss.loc		/VMware/Fail	Unable to apply DRS resource settings on host (Reason: A general system error occurred: In
☐	esx6.zenoss.loc		/VMware/Fail	Unable to apply DRS resource settings on host (Reason: A general system error occurred: In
☐	esx5.zenoss.loc		/VMware/Fail	Unable to apply DRS resource settings on host (Reason: A general system error occurred: In
☐	esx6.zenoss.loc	test-cent4-32-	/VMware/-Alarm	Alarm Virtual Machine CPU Usage on test-cent Green to Red
☐	esx5.zenoss.loc	w2k8-dev-ad0	/VMware	Remote console to w2k8-dev-ad01 on esx5.ze opened
☐	esx5.zenoss.loc	w2k8-dev-ad0	/VMware	User logged event: Remote console on w2k8-

The device column shows the ID of the VMware entity with which the event is associated, unless the event is specific to a guest VM. In that case, the device column shows the ID of the host, and the component column displays the ID of the guest.

Zenoss maps the VMware event to the event class and assigns the event a severity level. The event class appears in the eventClass column.

To see detailed event information and the original VMware event type, click the magnifying glass that appears at the end of the event row.

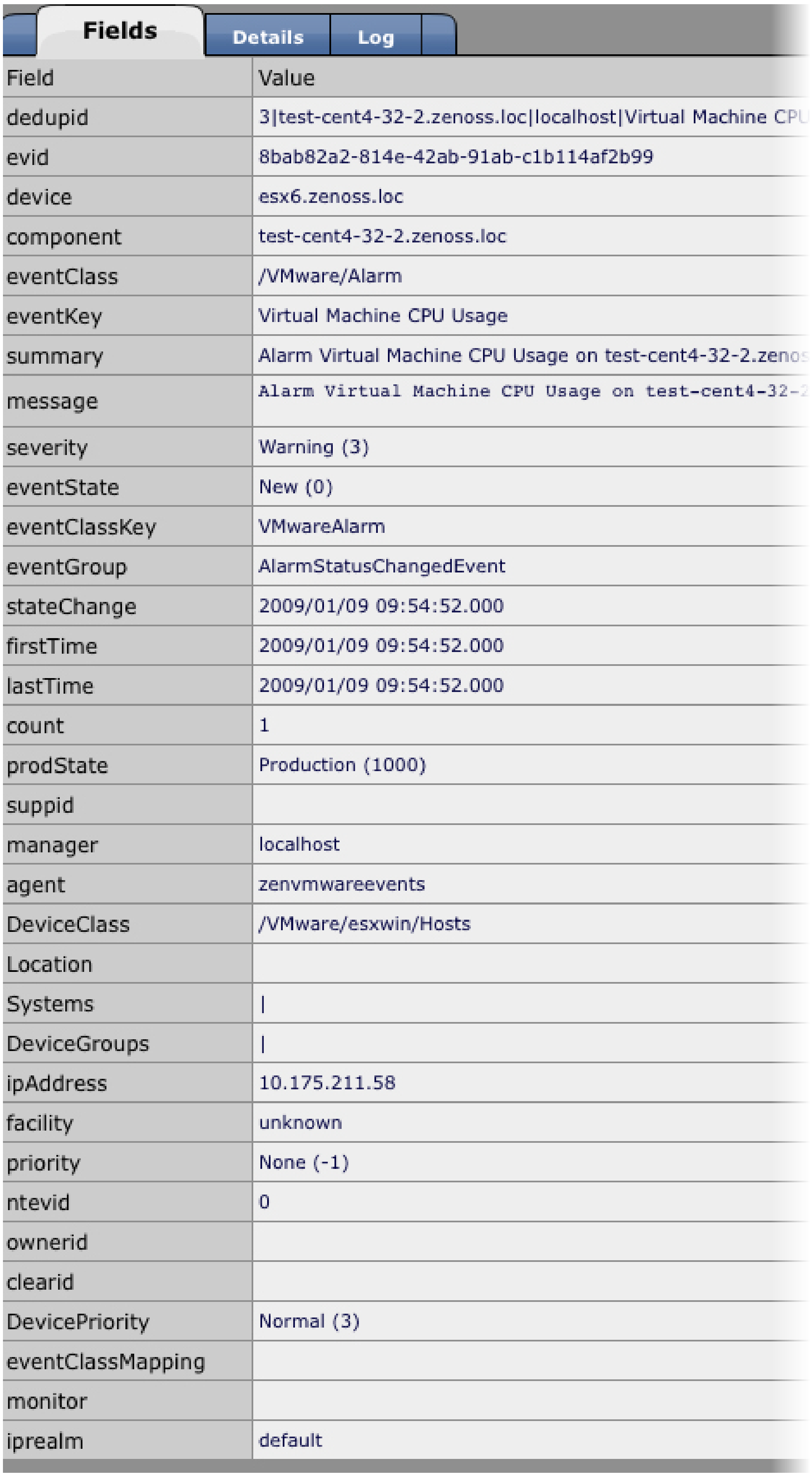
Figure 11.7. VMware Events (Event Console) - View Details



	lastTime	count	
	2009/01 18:08:5		
	2009/01 9:02:48		
	2008/12/29 10:08:45	1	
	2009/01/09 13:48:54	1	

The VMware event type is the value shown for eventGroup.

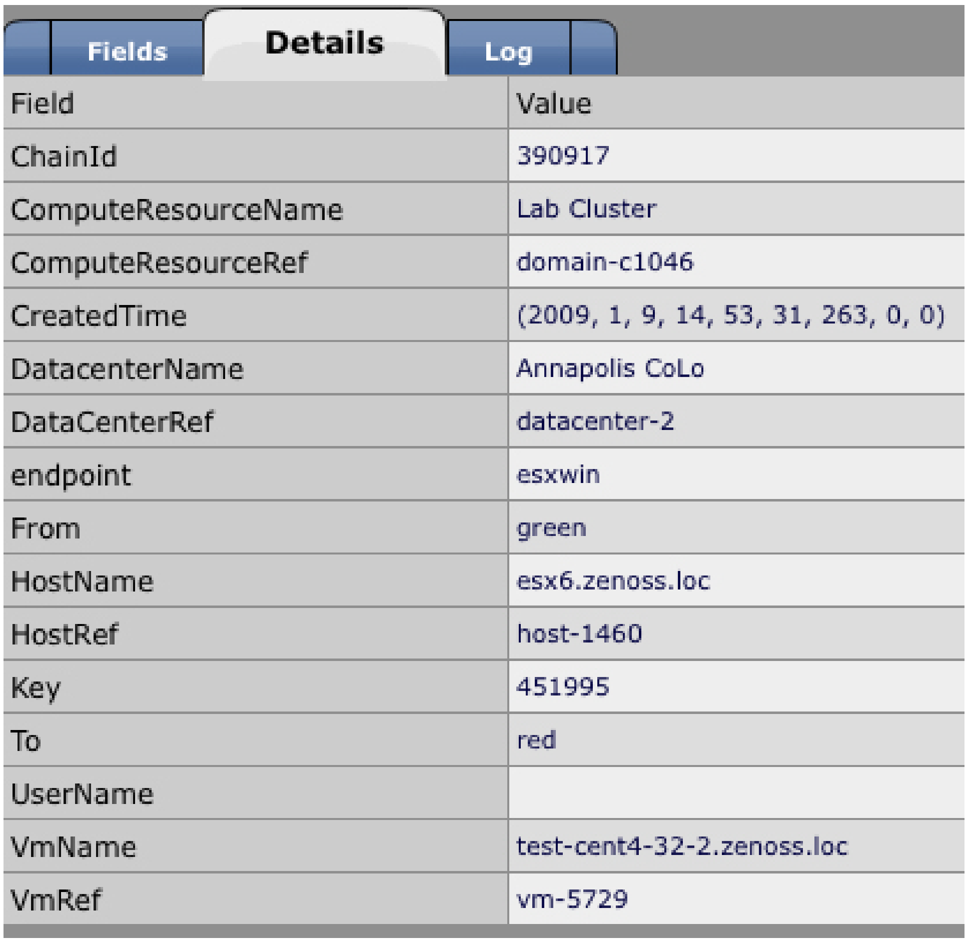
Figure 11.8. Event Details (Fields Tab)



Field	Value
dedupid	3|test-cent4-32-2.zenoss.loc|localhost|Virtual Machine CPU
evid	8bab82a2-814e-42ab-91ab-c1b114af2b99
device	esx6.zenoss.loc
component	test-cent4-32-2.zenoss.loc
eventClass	/VMware/Alarm
eventKey	Virtual Machine CPU Usage
summary	Alarm Virtual Machine CPU Usage on test-cent4-32-2.zenos
message	Alarm Virtual Machine CPU Usage on test-cent4-32-2
severity	Warning (3)
eventState	New (0)
eventClassKey	VMwareAlarm
eventGroup	AlarmStatusChangedEvent
stateChange	2009/01/09 09:54:52.000
firstTime	2009/01/09 09:54:52.000
lastTime	2009/01/09 09:54:52.000
count	1
prodState	Production (1000)
suppid	
manager	localhost
agent	zenvmwareevents
DeviceClass	/VMware/esxwin/Hosts
Location	
Systems	|
DeviceGroups	|
ipAddress	10.175.211.58
facility	unknown
priority	None (-1)
ntevid	0
ownerid	
clearid	
DevicePriority	Normal (3)
eventClassMapping	
monitor	
iprealm	default

The Details tab shows additional "raw" content from the VMware event.

Figure 11.9. Event Details (Details Tab)



Field	Value
ChainId	390917
ComputeResourceName	Lab Cluster
ComputeResourceRef	domain-c1046
CreatedTime	(2009, 1, 9, 14, 53, 31, 263, 0, 0)
DatacenterName	Annapolis CoLo
DataCenterRef	datacenter-2
endpoint	esxwin
From	green
HostName	esx6.zenoss.loc
HostRef	host-1460
Key	451995
To	red
UserName	
VmName	test-cent4-32-2.zenoss.loc
VmRef	vm-5729

1.6. Migration Events

When a VMotion guest migrates from one host to another, VMware records events to signal its progress. When a VmMigrated event occurs, it is duplicated to become two events, which are mapped to the `/VMware/Migration` event class in Zenoss. One event contains the originating host as the device; the other lists the destination host as the device.

An Event Command (see the Commands tab on the Event Manager page) reacts to these events by remodeling the two hosts and generating an updated view of the guests. The time required to produce updated guest lists (from the time migration completes) is between 30 seconds and four minutes.

1.7. Custom Data Sources

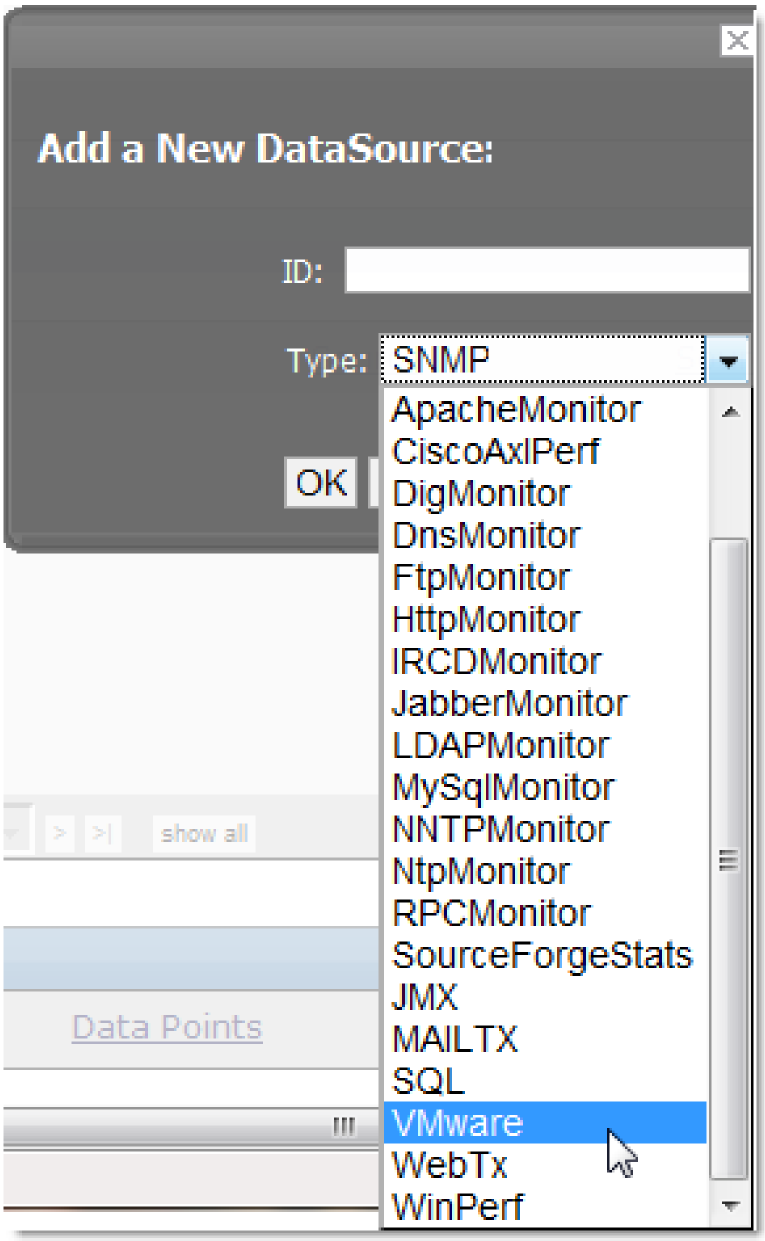
In Zenoss, metric-bearing VMware entities (such as Hosts, Guests, and Clusters) have associated templates. These templates define which metrics are gathered. By default, only a subset is collected; however, you can add more by adding data sources to the templates. Once created, you can then create custom graphs from these data sources.

To create a custom data source:

1. Select the template to which you want to add the data source.
2. From the DataSources list of options, select Add Datasource.

   The Add DataSource dialog appears.

Figure 11.10. Add Data Source



3. Select the VMware data source from the list of options.

4. Enter or select values to create the data source:

   - Event Key - Not used.
   - Severity - Not used.
   - Group, Counter, and Roll-up Type - VMware-specific data points are determined by this trio of strings. For information about each of these metrics, see the chapter titled "Performance Counters Reference" in the *VI SDK Programming Guide*, available from this location:

     http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/visdk25programmingguide.pdf

   - Instance - Certain metrics are further specified by an instance name. For example, the metric whose Group/Counter/Rollup Type triplet is Network/Network Data Receive Rate/average requires the name of the actual interface for full specification. In Zenoss, this metric is represented by the data source nicRx on the template VMwareNic. The VMwareNic template is bound to the individual host interfaces, each of whose ID is the interface name. In this case, the instance name is ${here/instanceId}.

5. Click Save to save the new data source.

Chapter 12. Event Monitoring

1. About Event Monitoring In Zenoss

The Zenoss Event Management system can collect events from syslog, Windows event log, SNMP traps, and XML-RPC. Processing is performed on raw events to integrate them tightly into the Zenoss model. Specifically, an event is run through a set of rules to determine the class of the event, which can then provide additional information such as event severity or up-down correlation.

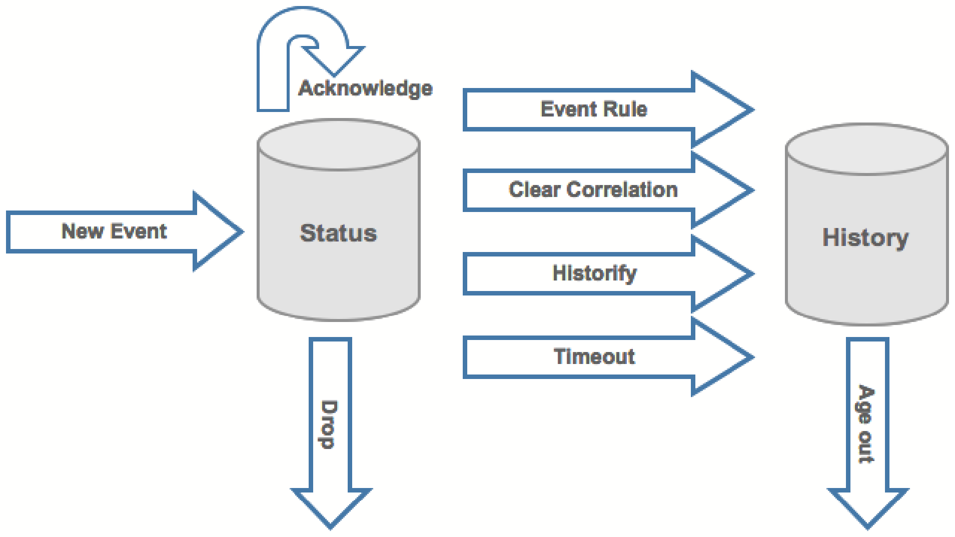
2. Event Concepts

These are the primary concepts associated with Zenoss and its event monitoring system.

2.1. Event Life Cycle

The Zenoss Event Life Cycle is a very straightforward process. The first step of the life cycle is the creation of the event. The default state of the event is set to “New”. The event can then be Acknowledged, Suppressed or “dropped” with an Event class rule. From there, an event will be archived into the Event History database in one of four ways. The event can manually be put into the history database; it can be put into the database due to auto clear correlation (bad event happens, good event for the same thing happens, move bad event to history), an event class rule, and an inactive timeout.

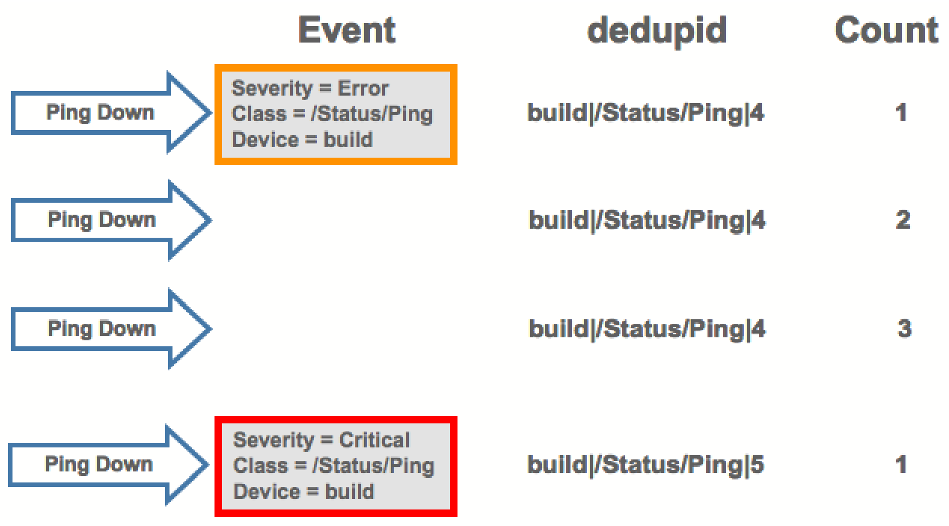
Figure 12.1. Event Life Cycle



2.2. De-duplication

If a single event is submitted multiple times for some reason, instead of the event clogging up the event log with hundreds or perhaps even thousands of events, an event counter is incremented. The matching is done through the event class. Additional matching event submissions do not generate more instances of that event in the event list, but that they only increased the event counter. This is important so that the occurrences of the same event occurrences do not create alert “chatter” in the time between when the event occurs and the time it takes you to acknowledge or correct the event.

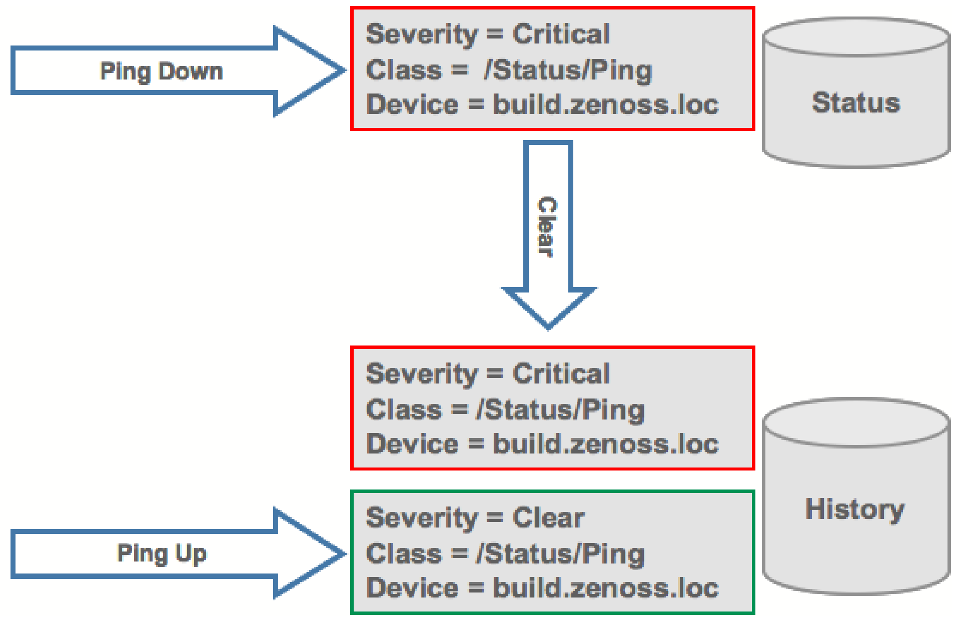
Figure 12.2. De-duplication



2.3. Begin-End Correlation

An event enters the Event Life Cycle at the start of the event, and at the end of the event, it leaves the life cycle. If a positive event that corresponds to a negative event that has already been received, the associated negative event is cleared. This is done by matching several key data points in the events themselves.

Figure 12.3. Begin-End Correlation

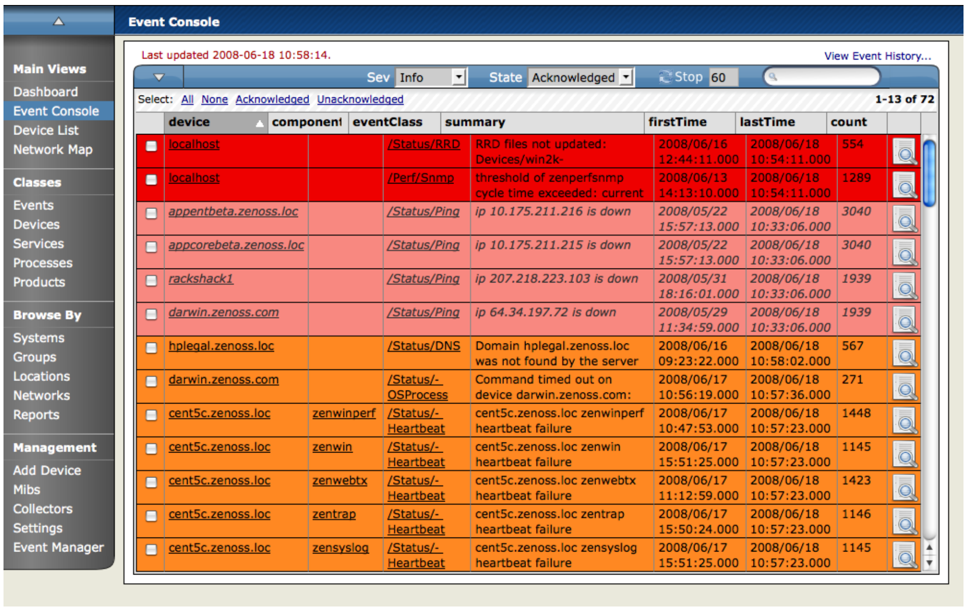


3. Zenoss Event Console

The Event Console is the central nervous system for event viewing. It is a repository for all events that come into the system. To access the event console, from the left navigation menu, click Event Console.

The Event console appears.

Figure 12.4. Zenoss Event Console



Last updated 2008-06-18 10:58:14.

Sev Info State Acknowledged Stop 60

Select: All None Acknowledged Unacknowledged 1-13 of 72

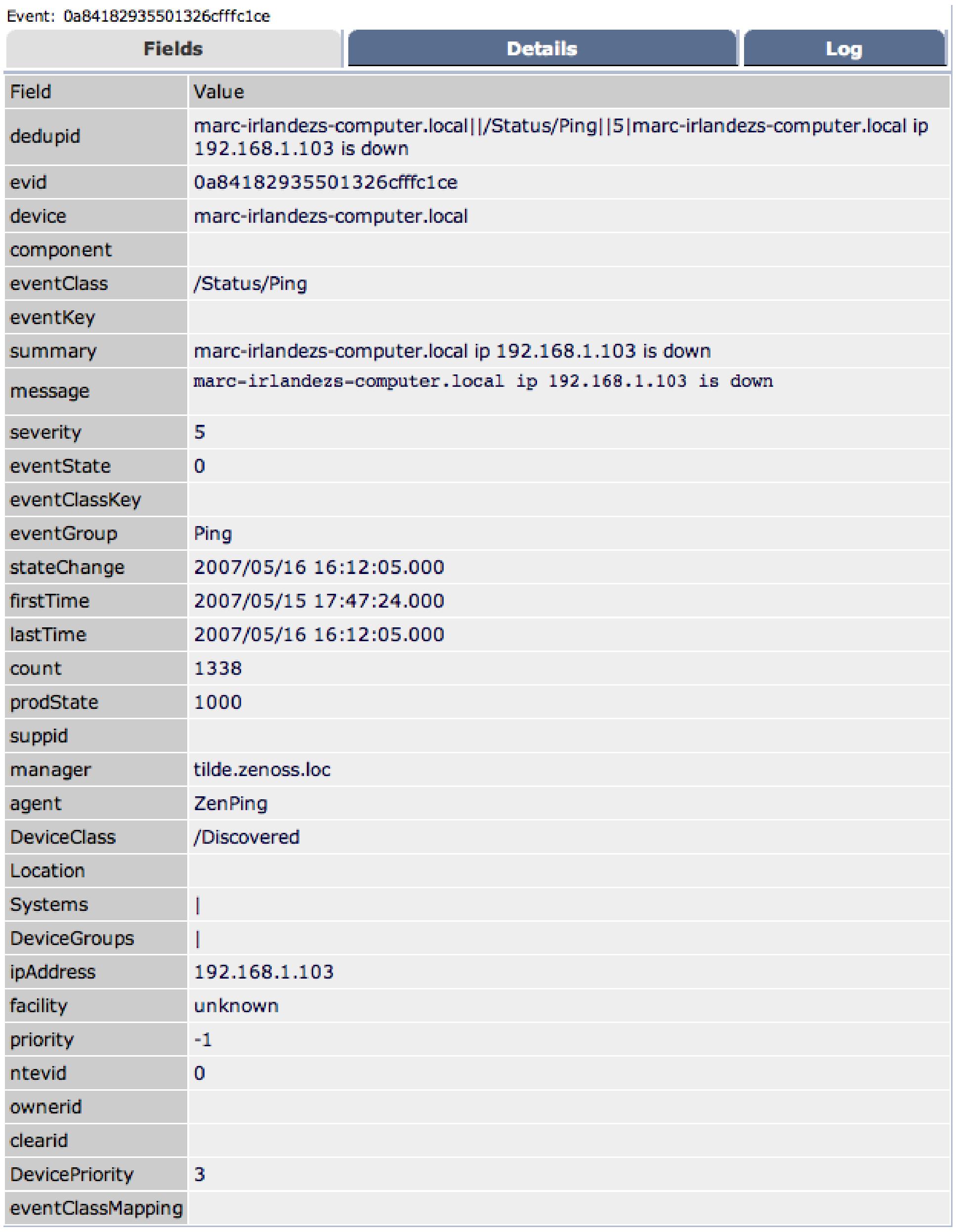
device	component	eventClass	summary	firstTime	lastTime	count
localhost		/Status/RRD	RRD files not updated: Devices/win2k-	2008/06/16 12:44:11.000	2008/06/18 10:54:11.000	554
localhost		/Perf/Snmp	threshold of zenperfsnmp cycle time exceeded: current	2008/06/13 14:13:10.000	2008/06/18 10:54:11.000	1289
appentbeta.zenoss.loc		/Status/Ping	ip 10.175.211.216 is down	2008/05/22 15:57:13.000	2008/06/18 10:33:06.000	3040
appcorebeta.zenoss.loc		/Status/Ping	ip 10.175.211.215 is down	2008/05/22 15:57:13.000	2008/06/18 10:33:06.000	3040
rackshack1		/Status/Ping	ip 207.218.223.103 is down	2008/05/31 18:16:01.000	2008/06/18 10:33:06.000	1939
darwin.zenoss.com		/Status/Ping	ip 64.34.197.72 is down	2008/05/29 11:34:59.000	2008/06/18 10:33:06.000	1939
hplegal.zenoss.loc		/Status/DNS	Domain hplegal.zenoss.loc was not found by the server	2008/06/16 09:23:22.000	2008/06/18 10:58:02.000	567
darwin.zenoss.com		/Status/-OSProcess	Command timed out on device darwin.zenoss.com:	2008/06/17 10:56:19.000	2008/06/18 10:57:36.000	271
cent5c.zenoss.loc	zenwinperf	/Status/-Heartbeat	cent5c.zenoss.loc zenwinperf heartbeat failure	2008/06/17 10:47:53.000	2008/06/18 10:57:23.000	1448
cent5c.zenoss.loc	zenwin	/Status/-Heartbeat	cent5c.zenoss.loc zenwin heartbeat failure	2008/06/17 15:51:25.000	2008/06/18 10:57:23.000	1145
cent5c.zenoss.loc	zenwebtx	/Status/-Heartbeat	cent5c.zenoss.loc zenwebtx heartbeat failure	2008/06/17 11:12:59.000	2008/06/18 10:57:23.000	1423
cent5c.zenoss.loc	zentrap	/Status/-Heartbeat	cent5c.zenoss.loc zentrap heartbeat failure	2008/06/17 15:50:24.000	2008/06/18 10:57:23.000	1146
cent5c.zenoss.loc	zensyslog	/Status/-Heartbeat	cent5c.zenoss.loc zensyslog heartbeat failure	2008/06/17 15:51:25.000	2008/06/18 10:57:23.000	1145

When looking at events, there are quite a few sorting and filtering schemes available. Events can be sorted by clicking on “device”, “component”, “eventClass”, “summary”, “firstTime”, “lastTime” and “count”. You can also filter by state and severity or by using the “Filter” field. “Filter” is a regular expression on all text seen in the event list. Events can be selected and either Acknowledged, moved to History, or Mapped into a specific location. You can also click on the component to navigate to the component where the event was triggered.

3.1. Viewing Event Details

You can view the details for any event in the system. To open the Event details window click on the magnifying glass in the right-most column in the Event Console. The Event Detail window opens showing the Fields tab.

Figure 12.5. Event Details Fields Tab



Event: 0a84182935501326cfffc1ce

Field	Value
dedupid	marc-irlandezs-computer.local||/Status/Ping||5|marc-irlandezs-computer.local ip 192.168.1.103 is down
evid	0a84182935501326cfffc1ce
device	marc-irlandezs-computer.local
component	
eventClass	/Status/Ping
eventKey	
summary	marc-irlandezs-computer.local ip 192.168.1.103 is down
message	marc-irlandezs-computer.local ip 192.168.1.103 is down
severity	5
eventState	0
eventClassKey	
eventGroup	Ping
stateChange	2007/05/16 16:12:05.000
firstTime	2007/05/15 17:47:24.000
lastTime	2007/05/16 16:12:05.000
count	1338
prodState	1000
suppid	
manager	tilde.zenoss.loc
agent	ZenPing
DeviceClass	/Discovered
Location	
Systems	|
DeviceGroups	|
ipAddress	192.168.1.103
facility	unknown
priority	-1
ntevid	0
ownerid	
clearid	
DevicePriority	3
eventClassMapping	

This event details screen highlights the minutia for the event.

4. Generating and Sending a Test Event

You can use Zenoss to add an artificial event to the event database to test your setups and mappings. To add a new event:

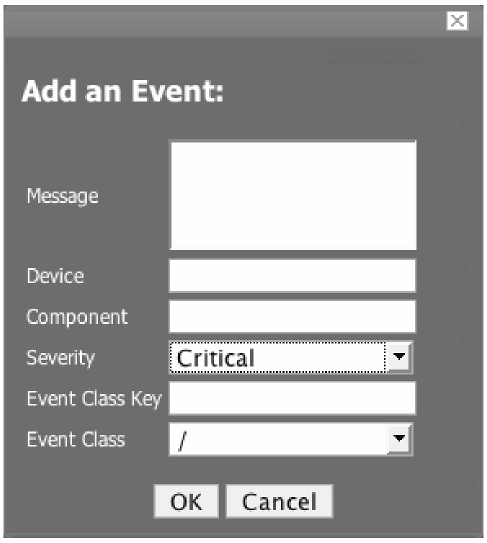
1. From the left navigation menu, select Events.

   The Events/Classes tab appears.

2. Open the Events page menu and select Add Event.

   The Add Event Screen appears.

Figure 12.6. Add Event Dialog


Add an Event:

Message

Device

Component

Severity: Critical

Event Class Key

Event Class: /

3. Fill out the fields according to the kind of event you want to send. The following fields are available:
   - Message
   - Device
   - Component
   - Severity
   - Event Class Key
   - Event Class
4. Click the OK button.

   The Event Console appears with your event at the top.

4.1. Sending Events from the Command Line

Zenoss allows you to send events TO Zenoss using a command line tool. This script only relies on python, so it can be moved easily to other systems. zensendevent uses the following options:

Usage:

```
zensendevent [options] summary words
```

Options:

- -d, --device

  The local fully-qualified domain name
- -p, --component

  Component from which this event is sent.
- -s, --severity

  Severity of this event: Critical, Error, Warn, Info, Debug, Clear. Default: Warn

- -c, --class

  Event class for this event

- --port

  xmlrpc server port. Default: 8081

- --server

  xmlrpc server. Default: localhost

- --auth

  xmlrpc server auth. Default: admin:zenoss

Example:

```
zensendevent -c /App/Fail -p sky -s Critical Onos\! very bad message\!
```

5. Event Classes

The first integration step performed on a received event is assignment of the event class. Event classes are added to the system using the hierarchical scheme found throughout Zenoss. Events are mapped to Event Classes by Event Class instances. Event Class instances are looked up by a non-unique key called EventClassKey.

5.1. Creating a New Event Class

To create a new Event class:

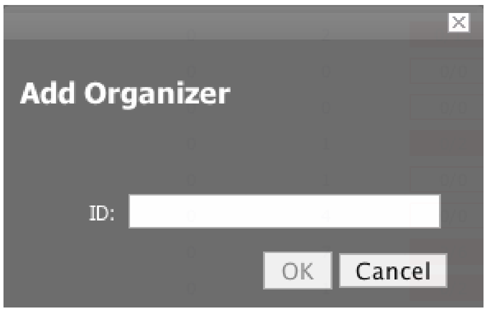
1. From the left navigation menu, click Events.

2. If it is not already open, click the Classes tab.

3. Open the SubClasses table menu and select the Add Organizer option.

   The Add Organizer dialog appears.

   Figure 12.7. Add Organizer Dialog

4. In the ID field, enter the name for the new Event Class.

5. Click OK.

   The Event class appears in the SubClasses list.

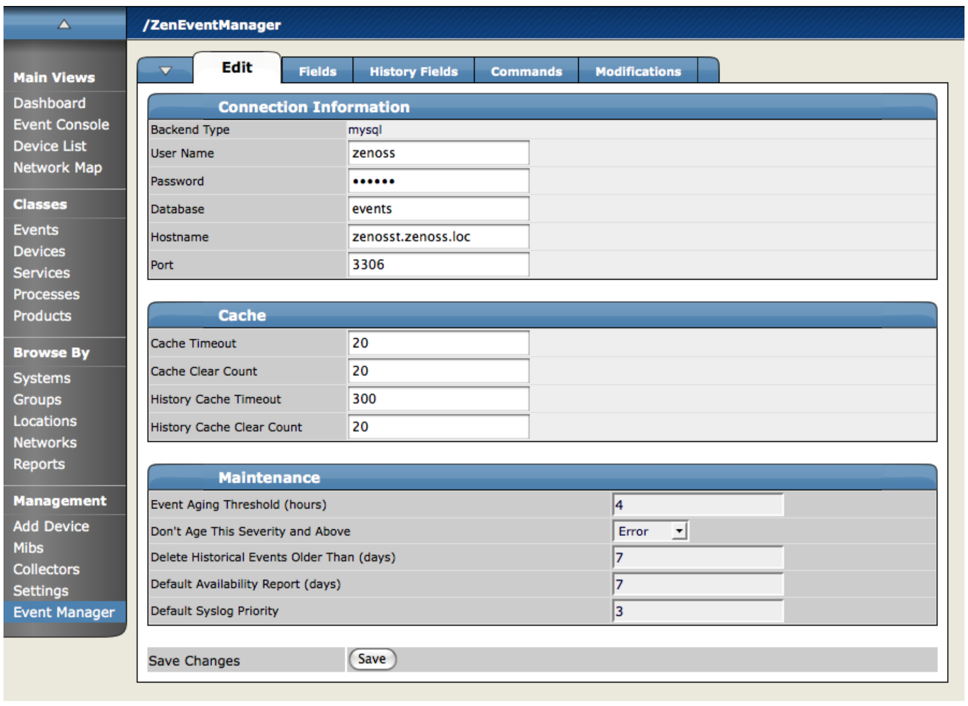
6. Event Manager Settings

You can specify specific settings for the Event Manager. The settings you can change are the connection to the MySQL Event Database, changing Event Cache timeouts and counts and make changes to the Event manager Maintenance Settings.

6.1. Accessing Event Manager Settings

From the left navigation menu, select Event Manager. The Event Manager Settings Edit tab appears.

Figure 12.8. Event Manager Edit Tab



Connection Information

Backend Type	mysql
User Name	zenoss
Password	••••••
Database	events
Hostname	zenosst.zenoss.loc
Port	3306

Cache

Cache Timeout	20
Cache Clear Count	20
History Cache Timeout	300
History Cache Clear Count	20

Maintenance

Event Aging Threshold (hours)	4
Don't Age This Severity and Above	Error
Delete Historical Events Older Than (days)	7
Default Availability Report (days)	7
Default Syslog Priority	3

Save Changes

6.2. Changing Event Database Connection Information

Navigate to the Event Manager and in the Connection information area make any necessary changes. The available connection information changes include:

- Backend Type – not editable at this point
- User Name – user name for the MySQL database
- Password – password for the user name identified above
- Database – which database to use
- Hostname – IP address or of the host
- Port – Port to use when accessing the event database

6.3. Changing Cache Settings for the Event Manager

Navigate to the Event Manager and in the Cache area make any necessary changes. The available Cache options include:

- Cache Timeout – sets the cache timeout for the event monitor. the lower the setting the more responsive your event console will be.
- Clear Cache Count – Clear the cache count for events (affects event counts, de-dup, etc.)
- History Cache Timeout – Sets the timeout for the History cache. Again the lower the number the more responsive your history will be.
- History Cache Clear Count – clears the counts for the history.

6.4. Change the Maintenance Settings for the Event Manager

Navigate to the Event Manager and in the Maintenance area make any necessary changes. The available fields include:

- Event Aging Threshold (hours) - This threshold tells Zenoss how long to wait before aging an event into the history table.
- Don't Age This Severity and Above - Whichever severity you specify here will not age out and be placed in the event history. So any event meeting the selected severity (or greater) will remain in the event list until acknowledged or moved into the event history manually.
- Delete Historical Events Older Than (days) - When an event in the event history is older than the number of days you enter here will automatically be purged (deleted) events from the event history.
- Default Availability Report (days) - Enter a number of days here for the automatically generated availability report for Zenoss. This reports gives a graphical summary of the actual availability and status of Zenoss going back the number of days you specify here.
- Default Syslog Priority - This is the default level of severity of an event in order to generate an entry into the syslog.

7. Acknowledging Events

Acknowledging Events is a way to let the system know that someone has seen the event and is either in progress fixing it or has fixed it or is saying the problem is under control. The acknowledged status for the event is displayed on the Dashboard.

To acknowledge an event:

1. Navigate to the event console.

2. Select the event you want to acknowledge from the Event Console by clicking the checkbox at the left of the event.

3. Open the Events page menu and select the Acknowledge Events option.

   Note the change in color (from deep red red to a pale red/pink) for the event you acknowledged.

4. Return to the dashboard, sort by Acknowledged and note the event appears in this list now.

8. Moving Events To History

Instead of deleting events, you manually send events to the event history. We call it “historifying” events. More detailing about automating when events are sent to the event history can be found in the Event Life Cycle Section.

To send an event to the event history:

1. Navigate to the Event Console.

2. Select the event you want to send to history from the Event Console by clicking the checkbox at the left of the event.

3. Navigate back to the Event console and select the same event (click the checkbox to the left of the event) scroll to the bottom of the page and click the History Button.

4. Open the Event Console page menu and select the Move Events to History option.

   A confirmation dialog appears.

5. Click OK.

   Note that the event disappears from the Event Console.

9. Clearing The Event History

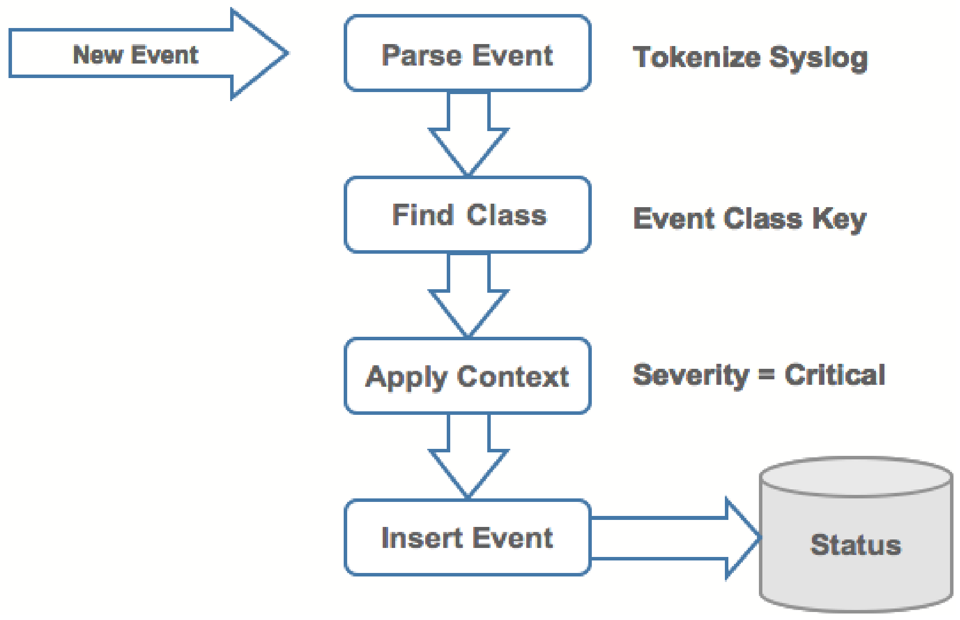
By default, the Event History is set to 0 which means to never age-out and delete any events in the Event History. You can change this setting by using the left navigation menu and choosing Event Manager. In the Maintenance area, set the number days to wait before deleting items from the history in the "Delete Historical Events Older Than (days)" field.

10. Event Class Mapping

Event Class Maps are the mechanism by which the events are integrated into the Zenoss system.

The following diagram shows an example of an event class mapping:

Figure 12.9. Event Class Mapping



In this example, the event comes into the system it is parsed, and then assigned to the appropriate event class. Then, as the event class key is found and associated with the event, then the context for that event class is applied to the incoming event. Then the status for the event is updated based on its classification.

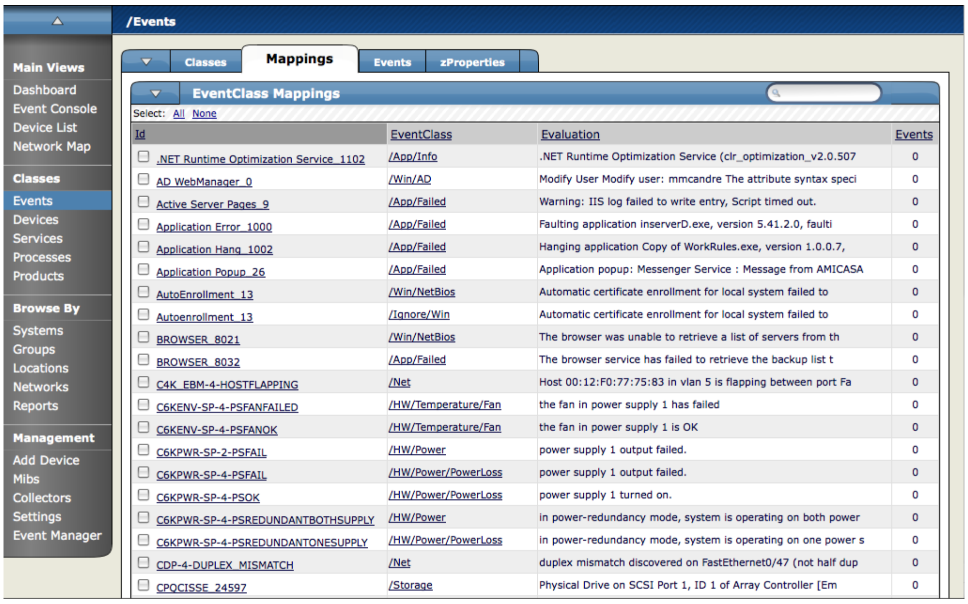
To create or update event class mappings:

1. From the left navigation menu, select Events.

2. Click the Mappings tab.

   The Mappings page appears.

Figure 12.10. Event Mapping Page

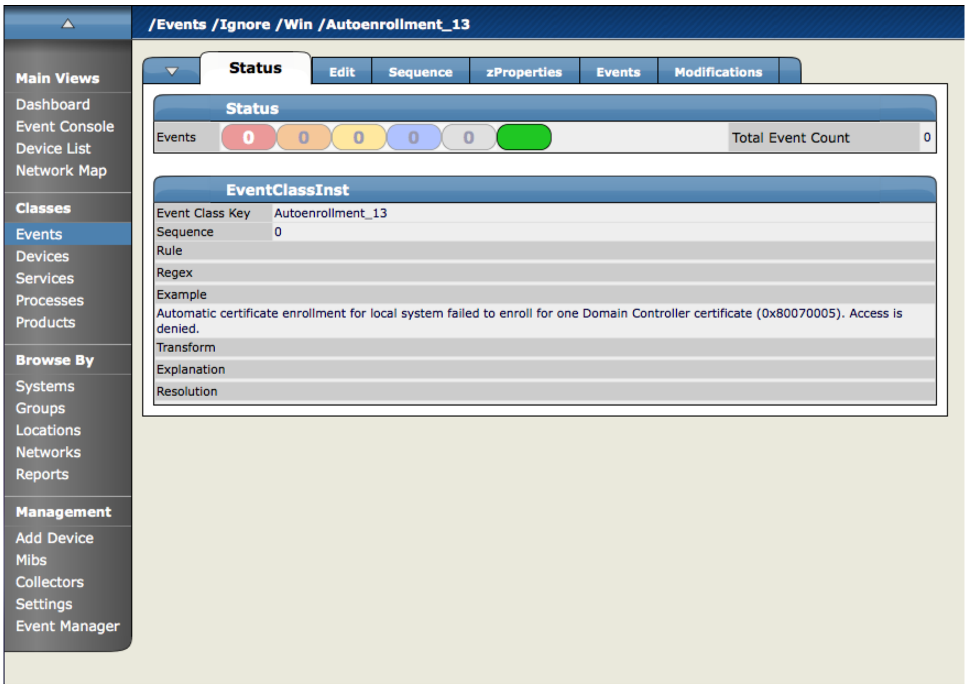


Id	EventClass	Evaluation	Events
.NET Runtime Optimization Service_1102	/App/Info	.NET Runtime Optimization Service (clr_optimization_v2.0.507	0
AD WebManager_0	/Win/AD	Modify User Modify user: mmcandre The attribute syntax speci	0
Active Server Pages_9	/App/Failed	Warning: IIS log failed to write entry, Script timed out.	0
Application Error_1000	/App/Failed	Faulting application inserverD.exe, version 5.41.2.0, faulti	0
Application Hang_1002	/App/Failed	Hanging application Copy of WorkRules.exe, version 1.0.0.7,	0
Application Popup_26	/App/Failed	Application popup: Messenger Service : Message from AMICASA	0
AutoEnrollment_13	/Win/NetBios	Automatic certificate enrollment for local system failed to	0
Autoenrollment_13	/Ignore/Win	Automatic certificate enrollment for local system failed to	0
BROWSER_8021	/Win/NetBios	The browser was unable to retrieve a list of servers from th	0
BROWSER_8032	/App/Failed	The browser service has failed to retrieve the backup list t	0
C4K_EBM-4-HOSTFLAPPING	/Net	Host 00:12:F0:77:75:83 in vlan 5 is flapping between port Fa	0
C6KENV-SP-4-PSFANFAILED	/HW/Temperature/Fan	the fan in power supply 1 has failed	0
C6KENV-SP-4-PSFANOK	/HW/Temperature/Fan	the fan in power supply 1 is OK	0
C6KPWR-SP-2-PSFAIL	/HW/Power	power supply 1 output failed.	0
C6KPWR-SP-4-PSFAIL	/HW/Power/PowerLoss	power supply 1 output failed.	0
C6KPWR-SP-4-PSOK	/HW/Power/PowerLoss	power supply 1 turned on.	0
C6KPWR-SP-4-PSREDUNDANTBOTHSUPPLY	/HW/Power	in power-redundancy mode, system is operating on both power	0
C6KPWR-SP-4-PSREDUNDANTONESUPPLY	/HW/Power/PowerLoss	in power-redundancy mode, system is operating on one power s	0
CDP-4-DUPLEX_MISMATCH	/Net	duplex mismatch discovered on FastEthernet0/47 (not half dup	0
CPQCISSE_24597	/Storage	Physical Drive on SCSI Port 1, ID 1 of Array Controller [Em	0

3. Click the name of the Event Class Mapping you want to Edit.

   The page for the Event mapping you have chosen appears.

Figure 12.11. IndividualEvent Mapping Page



Events	0	0	0	0	0	🟩	Total Event Count	0

Event Class Key	Autoenrollment_13
Sequence	0
Rule	
Regex	
Example	

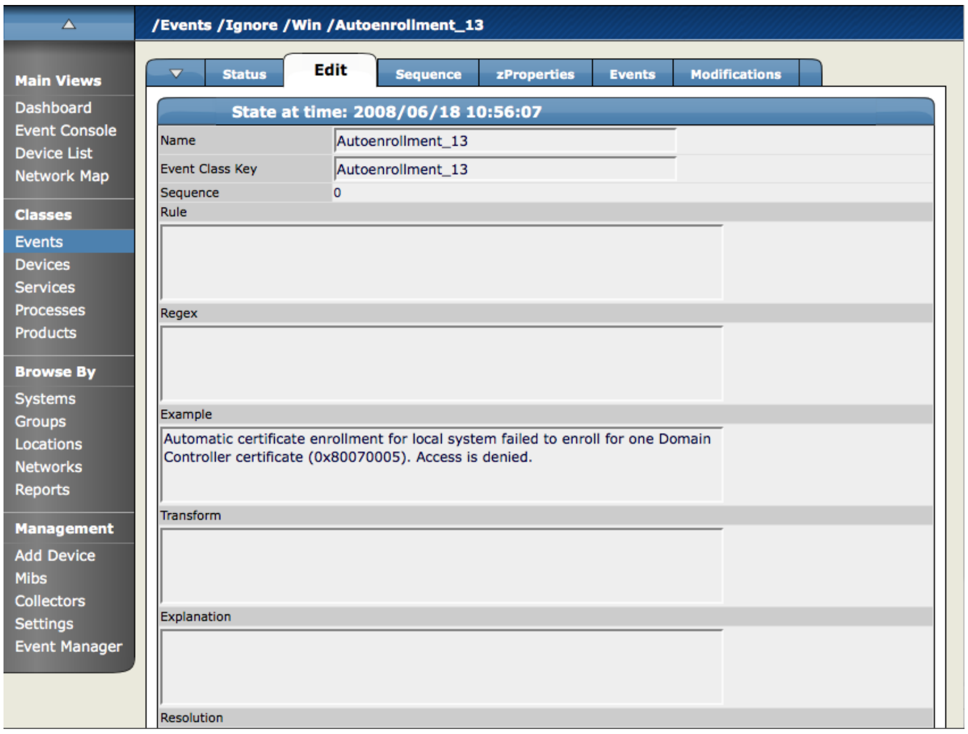
Automatic certificate enrollment for local system failed to enroll for one Domain Controller certificate (0x80070005). Access is denied.

Transform	
Explanation	
Resolution	

4. Click the Edit tab.

   The Edit Event mapping tab appears.

Figure 12.12. Edit Event Mapping Tab



State at time: 2008/06/18 10:56:07

Name: Autoenrollment_13

Event Class Key: Autoenrollment_13

Sequence: 0

Rule

Regex

Example: Automatic certificate enrollment for local system failed to enroll for one Domain Controller certificate (0x80070005). Access is denied.

Transform

Explanation

Resolution

5. Use the fields to define the Event Mapping.

   The fields in this tab are defined as follows:

   Name – Any name you want to call the Event Class Map.

   Event Class Key – The Event Class key is what is initially used to map incoming events to event classes. For syslog events, EventClassKey is most commonly the “Tag” or identifier of the syslog event. Often, the syslog tag maps to the process name from which the event came.

   Because the EventClassKey is non-unique, further matching may need to be performed to find a unique instance. This matching can be done through mechanisms, regular expression match, or Python expression evaluation. Because there will be list of instances against which these rules will be evaluated, the order of evaluation is important.

   Sequence – If there is more than one match you can use the Sequence field to define sequencing priorities. The “Sequence” tab allows all instances for a particular EventClassKey to be ordered.

   Rule – Enter a python expression to match to the event. This expression will be evaluated with the variable name “evt” bound to the current event. A detailed list of fields can be found in the Event Database Dictionary Appendix of the Admin Guide. An example of a rule using these fields would be:

   ```
   evt.priority>4
   ```

   Regex – The regex field is where you can enter regular expressions to match with events. When performing regular expression matches on an event extraction directives can be used to populate attributes of the event. These directives follow the Python format for named extractions in the form (?P<keyName>\S+).

Example - When creating a regular expression the original event text can be added to the example field and upon save the regular expression will be tested against this text. It is a great debugging tool for regex expressions.

Transform – one or more python statements. This allows you to modify the event through manipulation of the EVT variable. This section uses TALES Expressions, see the TALES Expressions section of this document for more information on using these expressions.

Explanation - Enter a textual description for matches for this event class mapping. Use in conjunction with the Resolution field.

Resolution - Use the Resolution field to enter resolution instructions for clearing the event.

If the EventClassKey lookup returns no results a second lookup will be performed using the key “defaultmapping”. Default mappings can be used to match large ranges of events by regular expression.

6. Once you have made all of your changes, click the Save button.

11. Applying Event and Device Context Using Event zProperties

Once an event has been mapped to its class, two things happen: (1) Its Event context is applied and (2) its Device context(s) are applied. EventClass context application is done by looking up the zProperty list zEventProperties.

Any property names found in zEventProperties are set in the same way that other zProperties are, except that when looking up the property, the prefix 'zEvent_' is added to the property name. When the zEventProperties value is changed, Zenoss creates a placeholder for each of the properties in the list on the zProperties screen.

A common event property to change is 'Severity' and, in fact, Severity is added by default to zEventProperties. To change the Severity of a classified event, change the value of zEvent_severity in the Event Class Path for the event.

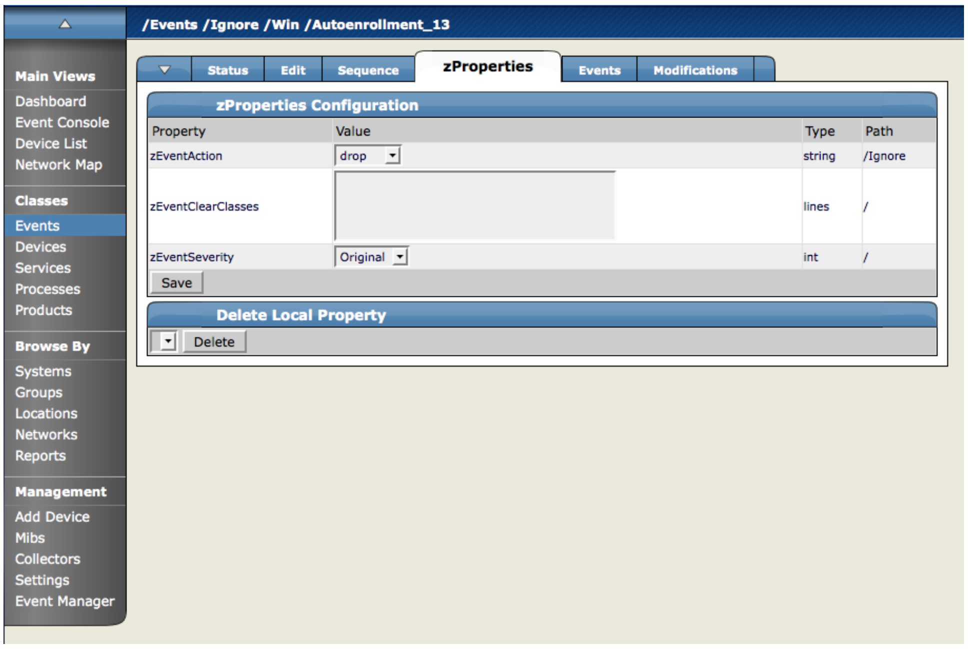
After the event context has been applied, then the device context is applied. During this process, the productionState, location, DeviceClass, DeviceGroups, and Systems, are all attached to the event in the event database. Once these properties have been associated with the event, Zenoss attempts to update the zEventProperties (as described above) but using the device class path instead of the event class path. This allows a particular device or class of devices to override the default values for any given event.

To edit the Event zProperties:

1. From the main screen for any Event Class Mapping, select the zProperties tab for the Event Class mapping you want to change.

   The zProperties tab for that mapping appears.

Figure 12.13. Event Mapping zProperties



Property	Value	Type	Path
zEventAction	drop	string	/Ignore
zEventClearClasses		lines	/
zEventSeverity	Original	int	/

2. Use the fields to define the Event zProperties.

   The three properties you can edit are:

   zEventAction – zEventAction tells what to do when there is a match for this class. Your choices are Drop, Status, and History.

   zEventClearClasses – allows you to enter matches that send clear events when they appear on this particular device.

   zEventSeverity – gives the default severity for events of this class.

12. Mapping Events Through the UI

As events come into the system, sometimes Zenoss has not seen them before and does not know how to classify them. As an administrator, you will classifying events and creating event maps through the UI. (So the event will be managed the same way each time.)

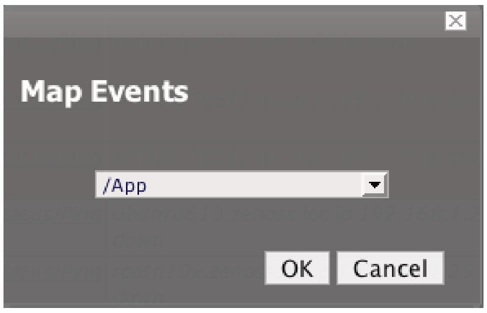
1. Navigate to the Event Console.

2. Select the event you want to map in the console by clicking in the checkbox to the left of the event.

3. Open the Event Console page menu and select the Map Events to Class option.

   The map events to class dialog appears.

Figure 12.14. Map Events Dialog



4. Select the event class where you want to map this event.

5. Click OK.

   The event class for this event and events with the same parameters will now take on the characteristics defined for the event class.

13. Using Mappings to Correlate Events

You can use Zenoss to create multiple event class mappings that can correlate two events and create an action based on the occurrence of the two events. An example would be creating a correlation such as the beginning of a failure with the end of a failure so that the clear event and then automatically move the event to the history database.

1. Navigate to one of the event classes you want to correlate.

   Click the Mappings tab. Create an additional mapping.

2. Click on the Classes tab again, then click on the mapping you just created, then click on its Edit tab.

3. Set Event Class Key back to the event class you want to correlate.

4. Make any changes to the Event Mapping fields you want such as adding the text “totally broken” in the Regex field.

5. Make a second mapping in the same class with a related name to the first so you know its correlated with the first mapping.

6. Set the Event Class Key the same event class key you set for the above event mapping.

7. Make any changes to the Event Mapping fields you want such as adding the text “everything is ok” in the Regex field.

8. Go to the zProperties tab and set zEventSeverity to Clear (in the case where you want the 2nd event to create a clear correlation). When this mapping matches the event will become a clear event and will clear all active events of the same class.

9. Once this is done, go to the “Sequence” tab and notice that all three mappings with the same event class key are listed here.

10. The order will control how the mappings are applied. Our first mapping will apply to all events with the given event class key and will prevent our new rules from becoming active.

11. Put this mapping at the end of the sequence by changing its sequence number to 3 and clicking Save.

14. Event Commands

Event Commands allow users to define arbitrary shell commands to be executed when events are received by Zenoss that meet the criteria defined in the event command. Event commands are carried out by the zenactions daemon. We're going to create an event command that uses the mappings we created in the last section. Our command will write text from our event into a log file.

14.1. Commands Tab

To access the Commands tab, from the left navigation menu, select Settings and then click the Commands tab. The Commands tab appears

<Commands Tab screen>

This tab shows all of the defined commands in Zenoss and you use the Define Commands table menu to Add or Delete any Commands.

14.2. Create an Event Command Example

This example will give you the framework for creating an event command.

1. Click on Event Manager and go to the Commands page.

2. Create a new event command called writeFile.

3. Edit the command with the following values.

   Set Enabled = True

   Default Command timeout = 60

   Delay = 0

4. Set the command as follows:

   ```
   echo "${evt/evid} ${dev/id} ${dev/manageIp} ${evt/message}" >> /tmp/cmdoutput
   ```

5. Set Clear Command

   ```
   echo "${evt/evid} ${evt/clearid} ${dev/id} ${evt/message}"" >> /tmp/cmdoutput
   ```

6. In the Where area, add a filter of Message contains "testing event commands".

7. Click Save.

14.3. Test the Event Command

The Add Event page is a good tool for testing event commands. Follow these steps to create an event that meets the criteria for the event command you just created.

1. Start watching the output file with the command:

   ```
   tail -f /tmp/cmdoutput
   ```

2. Create an artificial down event using the Add Event functionality and defining it as follows:

   Message = My Application is totally broken

   Device = <any device in your system>

   Event Class Key =

   Severity = Critical

3. Look at the output file. After the next zenactions cycle you should see the down event.

4. Now let’s clear the down event by sending an up event.

   Message = Now everything is ok

   Device = <any device in your system -same device you used above>

   Event Class Key =

   Severity = Info

15. SNMP Traps and Event Transforms

15.1. SNMP Trap Mapping

If an SNMP trap comes in and is classified as an "/Unknown" event type, this is because Zenoss does not know what you want to do with this event.

To map this event, from the Event Console, click the checkbox next to the unknown event, and from the page menu, select Map Events to Class.

The Event Mapping dialog appears. Each category does different things to the event: changing its severity, moving it to the history table, etc. For now, select "/App" and press the Map button.

To edit this mapping, from the left navigation, select Events and then click the Mappings tab. Search for and find your Event map and then click the name of the map. Now click the Edit tab.

This will take you to the edit screen for an Event "Mapper". These are the rules used to map this event to the "/App" category. This rule, since it matches the Trap by a very specific OID, is all you need.

In the "transform" section of the mapper, you can put some code to modify the summary. For example, lets say you want to set the summary string to "spam Filter Detects Virus". You would put this in the transform edit area:

```
evt.summary = "spam Filter Detects Virus"
```

A trap has a header with some standard (and mostly useless) information. But then it has a sequence of attribute/values You can see these values as event details if you click on the last column of the event.

You have indicated you want the value for the OID ".1.3.6.1.4.1.9789.1500.2.5" as the summary. If you had the MIB loaded, you could do this:

```
evt.summary = evt.spamFilterDetectsVirus
```

But... we have the OID and the data is still in there. We just need to use the slightly more cryptic:

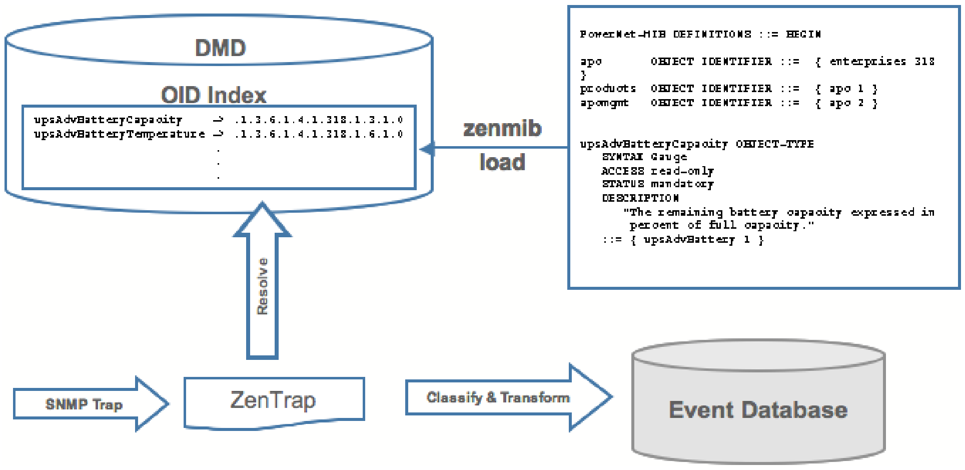
```
evt.summary = getattr(evt, ".1.3.6.1.4.9789.1500.2.5", "Unexpected missing OID")
```

The "device" object for the event has been made available, too:

```
evt.summary = getattr(evt, ".1.3.6.1.4.9789.1500.2.5", "Unexpected missing OID") + " from device " + devi
```

Zenoss uses MIBs to translate SNMP Traps that contain raw OID values Loading a MIB into Zenoss allows it to translate numeric OIDs like .1.3.6.1.2.1.1.6 into descriptive phrases like “sysLocation”. It also makes it easier to manipulate the events in an event mapping.

Figure 12.15. SNMP Trap Transform


DMD
OID Index
upsAdvBatteryCapacity → .1.3.6.1.4.1.318.1.3.1.0
upsAdvBatteryTemperature → .1.3.6.1.4.1.318.1.6.1.0
zenmib load
Resolve
SNMP Trap
ZenTrap
Classify & Transform
Event Database

```
PowerNet-MIB DEFINITIONS ::= BEGIN

apo       OBJECT IDENTIFIER ::=  { enterprises 318
}
products  OBJECT IDENTIFIER ::=  { apo 1 }
apcmgmt   OBJECT IDENTIFIER ::=  { apo 2 }


upsAdvBatteryCapacity OBJECT-TYPE
   SYNTAX Gauge
   ACCESS read-only
   STATUS mandatory
   DESCRIPTION
      "The remaining battery capacity expressed in
       percent of full capacity."
   ::= { upsAdvBattery 1 }
```

The following is a small demonstration MIB.

```
NOTIFICATION-TEST-MIB DEFINITIONS ::= BEGIN
IMPORTS
ucdavis FROM UCD-SNMP-MIB
NOTIFICATION-TYPE FROM SNMPv2-SMI
;
demonotifs OBJECT IDENTIFIER ::= { ucdavis 991 }
demo-notif NOTIFICATION-TYPE
OBJECTS { sysLocation }
STATUS current
DESCRIPTION "Just a test notification"
::= { demonotifs 17 }
END
```

15.2. Sending Test Traps

And here's how we send an SNMP trap.

1. From the command line, enter the following command:

   ```
   $ snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.2021.991.1.3.6.1.2.1.1.6 s "Device in Annapolis"
   ```

2. Save this demonstration MIB into a file.

3. Send the trap.

4. Open the Event Console and find the trap you just sent.

5. In the far right column of the event console, click the magnifying glass in the far right column for the event you just sent.

6. Click the Details tab.

   You should see:

   ```
   .1.3.6.1.2.1.1.6 Device in Annapolis
   ```

7. Send this event to the history.

You are now going to load some MIBs into Zenoss so we can get this OID translated into a more understandable format.

1. Copy your demonstration MIB into $ZENHOME/share/mibs/site.

2. Run zenmib to load it into Zenoss:

   ```
   $ zenmib run -v 10 DEBUG:zen.zenmib:TRAP-TEST-MIB.mib INFO:zen.zenmib:Unable to find a file providing t
   ```

3. Your MIB loaded, but it's missing some other definitions. They happen to be on the system somewhere else, so you can copy them in:

   ```
   $ cp /usr/share/snmp/mibs/SNMPv2-MIB.txt $ZENHOME/share/mibs/site $ cp /usr/share/snmp/mibs/UCD-SNMP-MI
   ```

4. Run zenmib again and get these loaded into Zenoss:

   ```
   $ zenmib run -v 10
   ```

5. Send the trap a second time:

   ```
   $ snmptrap -v 2c -c public localhost '' 1.3.6.1.4.1.2021.13.991 .1.3.6.1.2.1.1.6 s "Device in Annapolis
   ```

6. Now go check the event. Make sure the count is 1. If the count is 2, send the event to the history and send the trap again. Look at the Details tab. Now you should see something like this:

   ```
   sysLocation Device in Annapolis
   ```

   You should also see that the event summary changes from:

   ```
   snmp trap 1.3.6.1.4.1.2021.13.991 from localhost
   ```

   to:

   ```
   snmp trap ucdExperimental from localhost
   ```

   While this is an improvement, we would like to get the “sysLocation” value “Device in Annapolis” into the summary so that users do not have to drill down into the detail screen.

15.3. Transforming Events with Event Mappings

To modify events as they arrive, we will create an event map through the user interface as we did earlier. Create a new Event Class.

Go to the event console and create an event mapping in this class from the existing event.

On the map go to the “Edit” tab. We can filter what events this mapping will work against. For example you could filter on messages that contain the text ucdExperimental. Type “ucdExperimental” in the Regex field.

You can update the event with detail data in the Transform section. The entry field allows you to put in arbitrary Python scripts. The event is provided as “evt” and the device as “dev”. In this case, we'll extract the sysLocation event detail and make it the summary:

```
evt.summary = evt.sysLocation
```

Save this event mapping. Move all the events to history. Resend the trap. The summary for the trap should now read the device name in the location you have assigned.

If you have any problems with the Transform, check the zentrap.log file for errors that occurred.

15.4. Event Transforms Based on Event Class

When an event arrives into the Zenoss system, values (such as severity) can be changed. For example, the summary can be made more informative, or the severity changed according to some text within the summary.

Each Event Class allows for a short python script to be executed when an event arrives.

For example, a user may want full-filesystem threshold events on /data to be critical. The following python script in the Threshold Transform of /Events/Perf/Filesystem:

```
if evt.component == '/data' and evt.severity != 0:
   evt.severity = 5
```

Like event mappings for Event Class Keys, both "evt" and "device" objects are available within the script of the transform.

16. Custom Event Views

In this activity, you will create and edit a custom event view so you can save views of the event list that have been narrowed down according to filters you set and save. These custom event views are set on a per user basis.

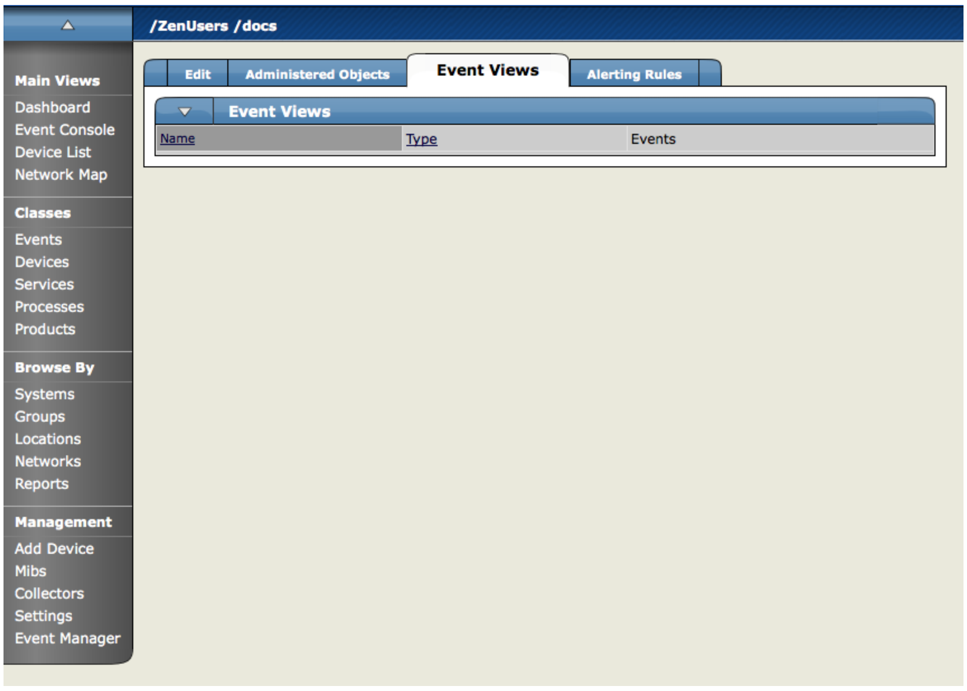
To create a new custom event view:

1. Click the Preferences link in the top right of the dashboard.

2. Click the Event Views tab.

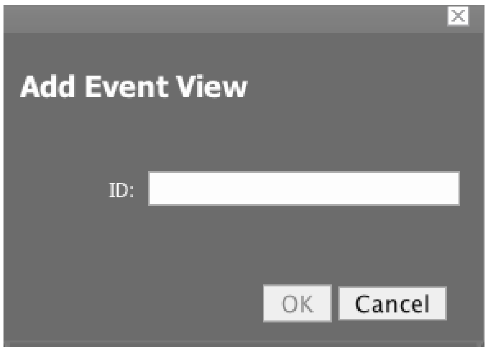
   The Event Views tab appears.

Figure 12.16. Custom Event Views - Initial Views



3. Open the Event View table menu and select the Add Event View option. The Add Event View dialog appears.

Figure 12.17. Add Custom Event View



4. In the ID text field, enter the name for the new Custom Event View.

5. Click OK.

   This custom event view appears in the list. Notice there is a custom alerting rainbow for this event view.

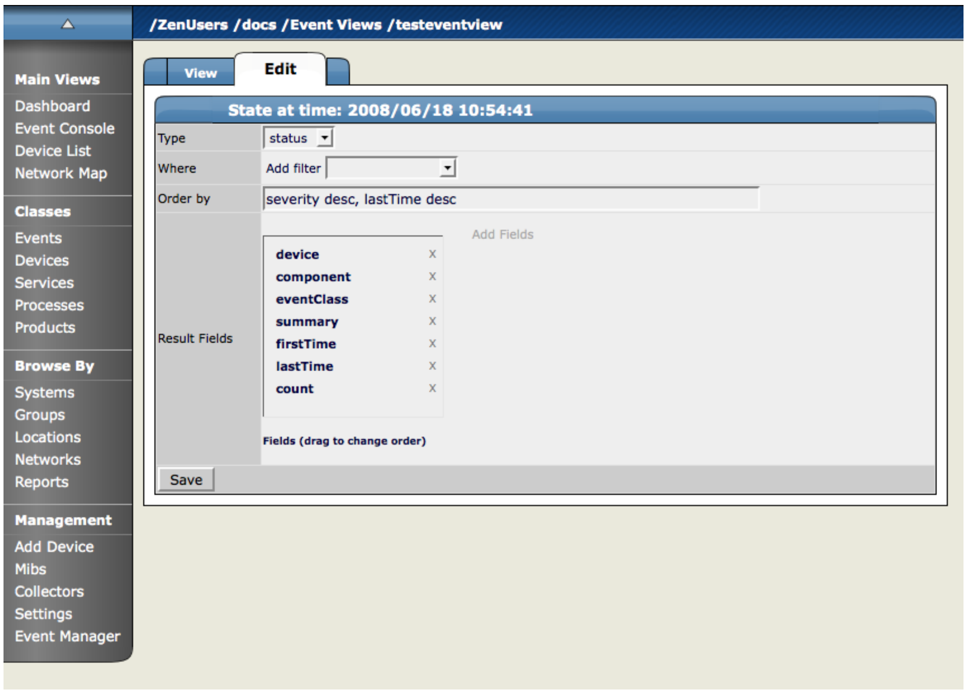
6. Click the link for the new event view you created.

   Notice the size of the list and the number of entries.

7. Click the Edit tab.

   The Edit Event views tab appears.

Figure 12.18. Custom Event Views - Edit Tab



State at time: 2008/06/18 10:54:41

Type: status

Where: Add filter

Order by: severity desc, lastTime desc

Result Fields: device, component, eventClass, summary, firstTime, lastTime, count

Add Fields

Fields (drag to change order)

8. Add conditions for this event view.

   Type - Here you can select if you want to show active events or the event history.

   Where - Use the Where area to add filters (similar to alerting rules where clauses.)

   Order by - his determines the order of the entries in the view.

   Result Fields: Result fields are the fields that appear in the view. For this activity, we will remove eventClass from the view. Click the X next to eventClass in the Result Fields area.

9. Click Save.

10. Click the View tab to see the results of this Custom Event view.

17. Use ZenMail and ZenPop to Turn Email Messages into Zenoss Events

ZenMail and ZenPop allow network administrators to turn email messages into events in Zenoss. This can be useful in situations where embedded systems (WAPs, NAS devices, RAID controllers) rely on email notification for events.

17.1. ZenMail

ZenMail serves as an SMTP server that you can bind to a specific TCP port. You can then configure your embedded system to send mail to the Zenoss server explicitly by using the Zenoss's server's IP address as the relay. Alternatively, ZenPop allows you to retrieve event emails from a POP server.

ZenMail supports the following configuration directives:

${ZENHOME}/bin/zenmail (no arguments): default operation. Binds to port 25 on all ports and listens for email messages to arrive. Ignores the TO field in the email and uses the FROM address as the device IP address.

${ZENHOME}/bin/zenmail --listenPort: Bind to the port provided. Useful in situations where an SMTP server is already running on the Zenoss server and you don't wish to interfere with the existing mail delivery system. Semantics are the same as the no-argument version (FROM address used as the device IP).

17.2. ZenPop

ZenPOP supports the following configuration directives:

--usessl: Issue the STARTTLS command to the POP server and attempt to transfer email messages using SSL encryption. This is required if retrieving mail from Google.

--nodelete: Do not issue the DELE command after retrieving all messages. This is typically used during initial testing so that you don't have to resend test messages to the POP account. Note that some email systems (most notably Google) do not actually delete messages when the DELE command is issued.

--pophost: The hostname or IP address of the POP server to retrieve messages from.

--popport: The TCP port the POP server listens on. Defaults to 110. Used in situations where the POP provider listens on another port (e.g. Google on port 995)

--popuser: The username that contains email messages to retrieve

--poppass: The password to use for the username provided

--cycletime: The time to sleep between polls. All emails are retrieved and then ZenPOP sleeps for this amount of time before waking up and attempting to pull new emails.

17.3. How Zenoss Translates Various Message Elements to the Event

17.3.1. How Zenoss uses the FROM field

If the FROM field is an IP address Zenoss associates the event with the device that has the same IP address. If the FROM field is a FQDN Zenoss resolves it to an IP address and performs the device association using the resolved IP address. The resolution of hostname uses A records and not MX records.

17.3.2. How Zenoss uses the TO Field

Zenoss completely ignores the TO field in the email message. ZenMail accepts email to any user and domain name combination. ZenPOP also drops the TO field and only pays attention to the FROM field.

17.3.3. How Zenoss uses the SUBJECT Field

ZenMail and ZenPOP use the SUBJECT as the summary for the event

17.3.4. How Zenoss uses the Message Body

ZenMail and ZenPOP use the first mime attachment as the event details. Secondary message bodies (typically HTML encoded versions of the message) are ignored, as are any additional attachments (typically files).

Chapter 13. Availability Monitoring

1. Monitoring Topology with ZenPing

The availability monitoring system within Zenoss provides active testing of the IT Infrastructure. The system currently consists of Zenping, Zenoss' Layer-3 aware topology-monitoring daemon, and Zenstatus, a TCP status tester.

ZenPing is configured automatically. You can use the About page, Status tab to stop and start Zenping. ZenPing does the high-performance asynchronous testing of the ICMP status. The most important element of this daemon is that Zenoss has built a compete model of the your routing system. If there are gaps in Zenoss' routing model, the power of ZenPing's topology monitoring will not be available. If there are these gaps, this issue can be seen in the zenping.log file.

Zenmodeler goes out and discovers the routes to each device in the Zenoss network. Zenoss tries not to use Internet routing tables and prefers to rely on Zenmodeler to discover the relationships on its own and create its own network map.

Basically if any known route is broken, there will only be one ping event that is generated by the outage. Any additional outages beyond that will only flag that device and the next time a ping sweep occurs the errors beyond the known router will not occur.

Zenmodeler works from the Zenoss system to further away from the server.

This monitoring model breaks down if the routers do not share their routing tables and interface information.

1.1. Controlling the Ping Cycle Time

1. From the left Navigation menu, select Monitors and select the Status monitor localhost and click the localhost link.

2. Notice the list of machines being pinged by this monitor.

3. In the “Edit” tab, change “Cycle Interval” to the desired interval.

4. On the next configuration cycle, the ping monitor will ping at the interval you set.

1.2. Using the Predefined /Ping Device Class

The /Ping device class is an example of a configuration for devices that should only be monitored for availability. Zenoss will not gather performance data for devices placed under this class; it will only ping them. You can use it as a reference for your own configuration; or, if you have a device that you want be monitored for availability alone, you can place it under this class.

2. Monitoring TCP Services

Use the Service menu to manage and monitor services that are running on your networks. To access the Services pages, from the left Navigational menu, choose Services. The Services Overview appears.

The Services overview shows the folders and sub-folders and lists all of the services that have been added to the system to monitor.

2.1. Zenstatus

ZenStatus performs monitoring of TCP services. It is configured by turning on monitoring of a service under the “Services” root on the Navigation Toolbar. Service monitoring can be turned on a service class but this can be overridden on any service instance. For example, “SMTP” will be monitored by default but it may not be a critical service on all boxes. If this is the case, it may be removed on specific devices. Also, if the service is configured to only listen on localhost (127.0.0.1) the service will not be monitored.

2.2. Adding a Service To Monitor

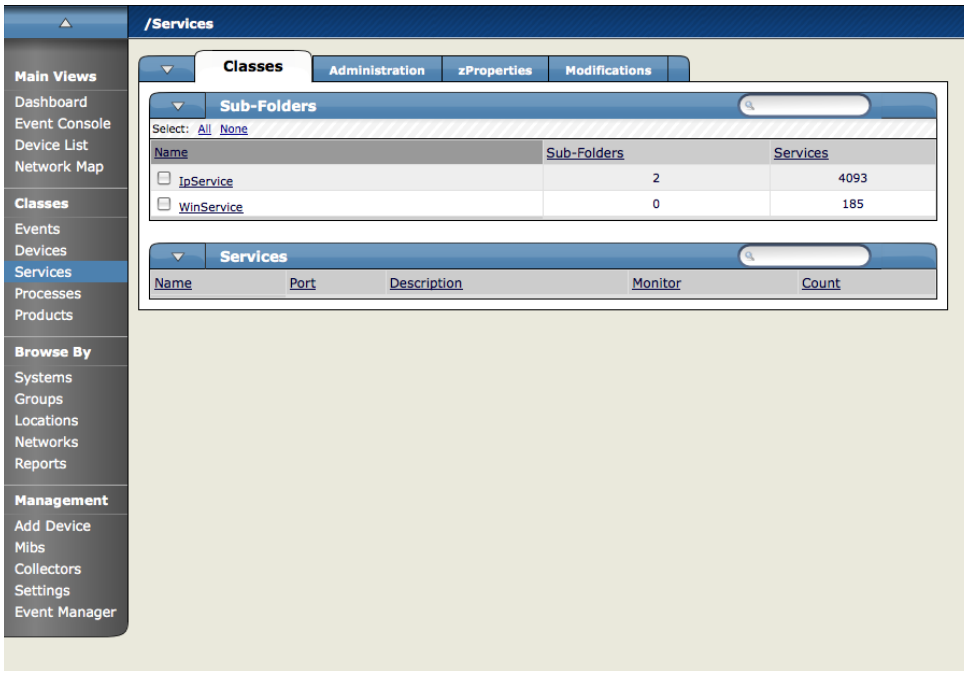
To add a service to monitor:

1. From the Services Overview page, in the Services text field, enter the name of the service you want to monitor.

2. Click the Add button.

   The service appears in the Services list.

   Figure 13.1. Services List - Classes tab



Name	Sub-Folders	Services
IpService	2	4093
WinService	0	185

Name	Port	Description	Monitor	Count

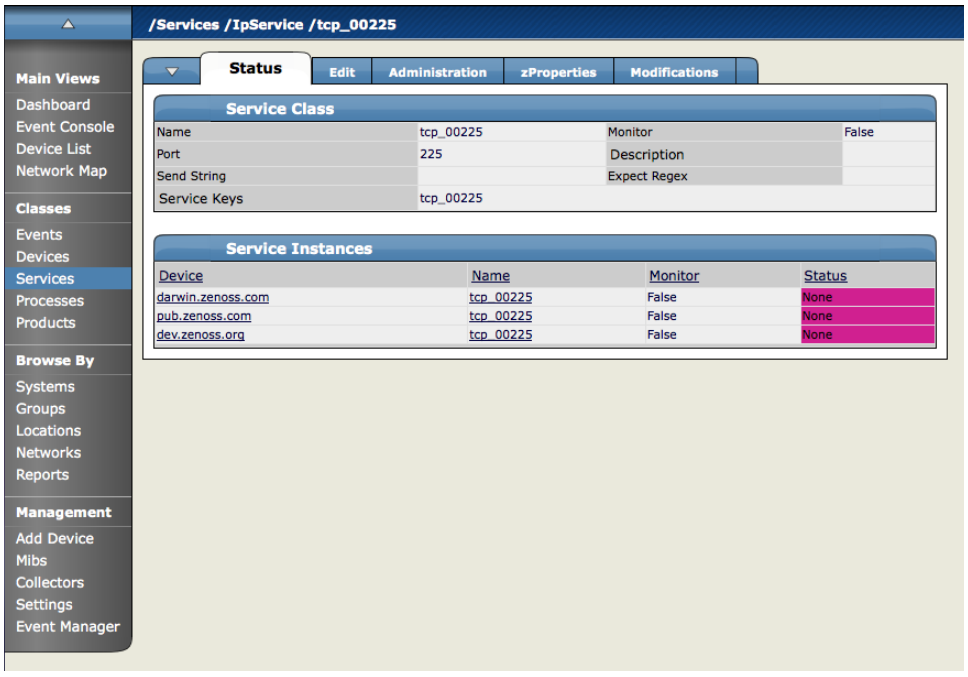
3. To set monitoring to True, click the Edit tab and set the Monitor pop-up to True. The service is now being monitored.

2.3. Monitoring Status Service Status Information

To view the status information associated with a given service, select the service from the Services list in the Services Overview page.

The Individual Service Status tab appears.

Figure 13.2. Individual Service Status Tab



Service Class

Name	tcp_00225	Monitor	False
Port	225	Description	
Send String		Expect Regex	
Service Keys	tcp_00225		

Service Instances

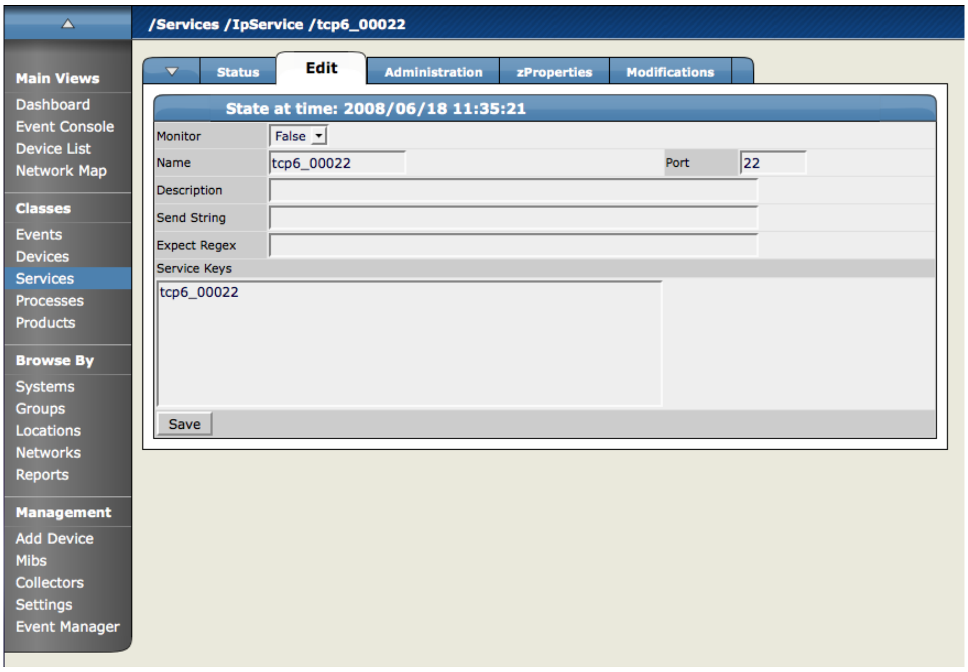
Device	Name	Monitor	Status
darwin.zenoss.com	tcp_00225	False	None
pub.zenoss.com	tcp_00225	False	None
dev.zenoss.org	tcp_00225	False	None

This tab shows you a summary of the details associated with this service. You can see the name of the service, whether its monitored or not, a Description, any associated Service Keys and a List of Devices where the service is currently running. To change any of this information, click the Edit tab.

2.4. Editing Service Information

To edit the information that appears on the Individual Service Status tab, from the Individual Services page, click the Edit tab.

Figure 13.3. Individual Service- Edit Tab



State at time: 2008/06/18 11:35:21

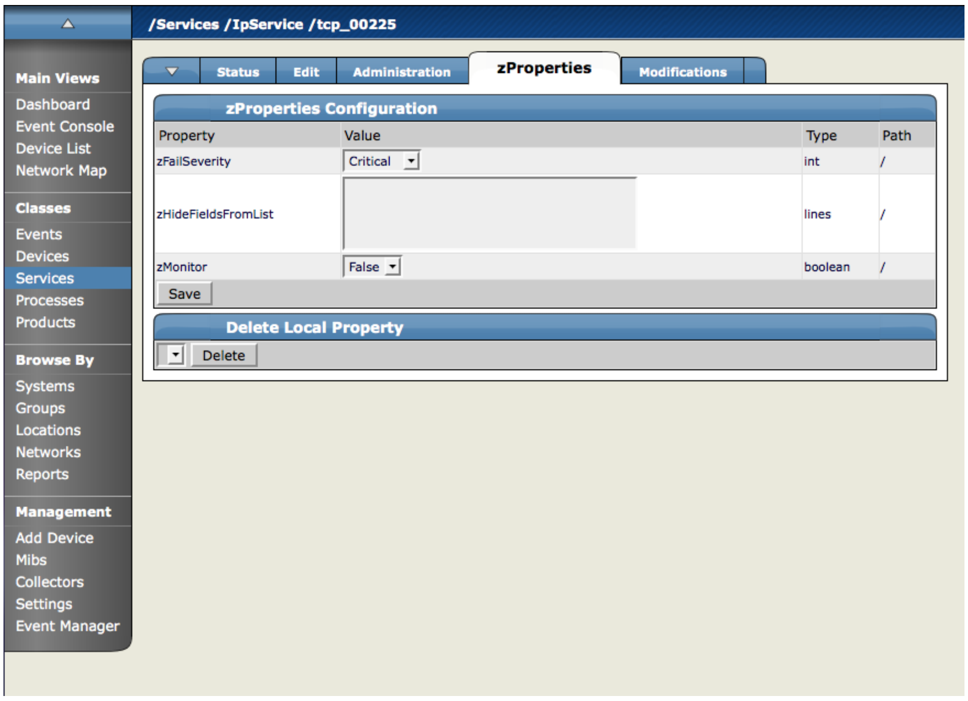
Monitor	False		
Name	tcp6_00022	Port	22
Description			
Send String			
Expect Regex			
Service Keys			
tcp6_00022			

From this screen, use the Monitor pop-up to select True to monitor the service and False to not monitor the service. You can also add any associated Service Keys or enter a brief Description.

2.5. Configuring Service zProperties

You can configure zProperties either for all services, for an individual service or for any services that fall further down in the service hierarchy tree. To configure them for all services click the zProperties tab from the Service Overview tab. To configure zProperties for an individual service, click on the service name in the Service Overview and then click the zProperties tab for that service. The Service zProperties Tab appears.

Figure 13.4. Individual Service - zProperties Tab



Property	Value	Type	Path
zFailSeverity	Critical	int	/
zHideFieldsFromList		lines	/
zMonitor	False	boolean	/

You can configure the following zProperties for an individual service from this tab:

- ZFailSeverity
- ZHideFieldsFromList
- zMonitor

  For more information about the Services zProperties, see the zProperties Appendix.

2.6. Using the Predefined /Server/Scan Device Class

The /Server/Scan device class is an example configuration for monitoring TCP services on devices using a port scan. You can use this device class as a reference for your own configuration; or, if you have a device that you want to be monitored for service availability alone, you can place it under this device class. Zenoss will not collect performance data for devices in this class.

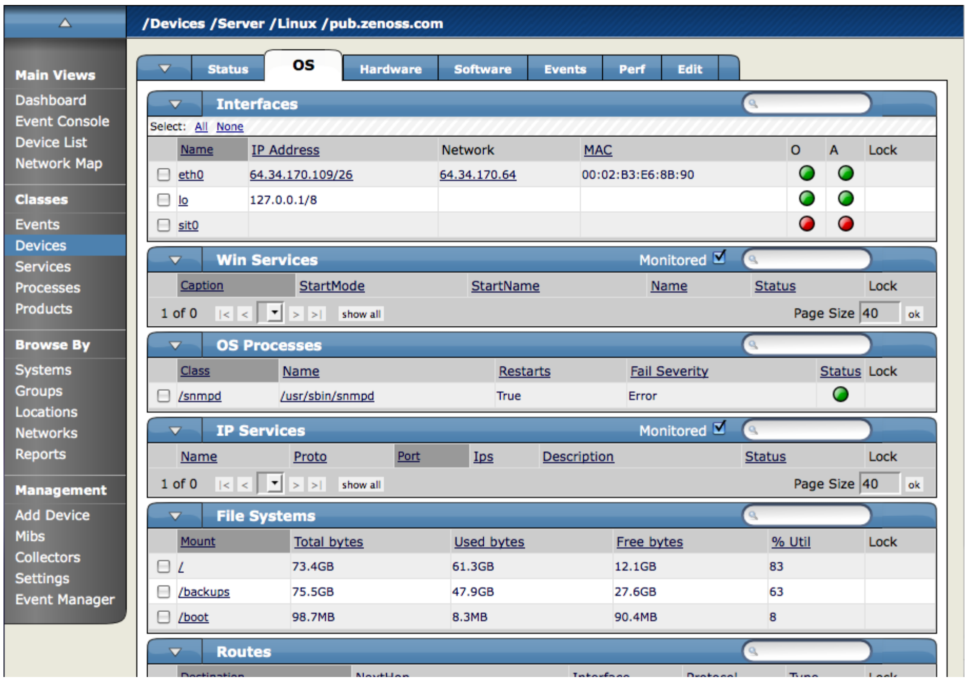
2.7. Monitoring a Service Using a Service Class

This section will show you how to set up monitoring of an IP service across a group of devices using a service class.

1. Navigate to the OS tab for a device you have loaded into the system.

The OS Tab for a device appears.

Figure 13.5. Showing Processes to Monitor



/Devices /Server /Linux /pub.zenoss.com

Interfaces

Select: All None

Name	IP Address	Network	MAC	O	A	Lock
eth0	64.34.170.109/26	64.34.170.64	00:02:B3:E6:8B:90	🟢	🟢	
lo	127.0.0.1/8			🟢	🟢	
sit0				🔴	🔴	

Win Services Monitored ✓

Caption	StartMode	StartName	Name	Status	Lock

1 of 0 show all Page Size 40 ok

OS Processes

Class	Name	Restarts	Fail Severity	Status	Lock
/snmpd	/usr/sbin/snmpd	True	Error	🟢	

IP Services Monitored ✓

Name	Proto	Port	Ips	Description	Status	Lock

1 of 0 show all Page Size 40 ok

File Systems

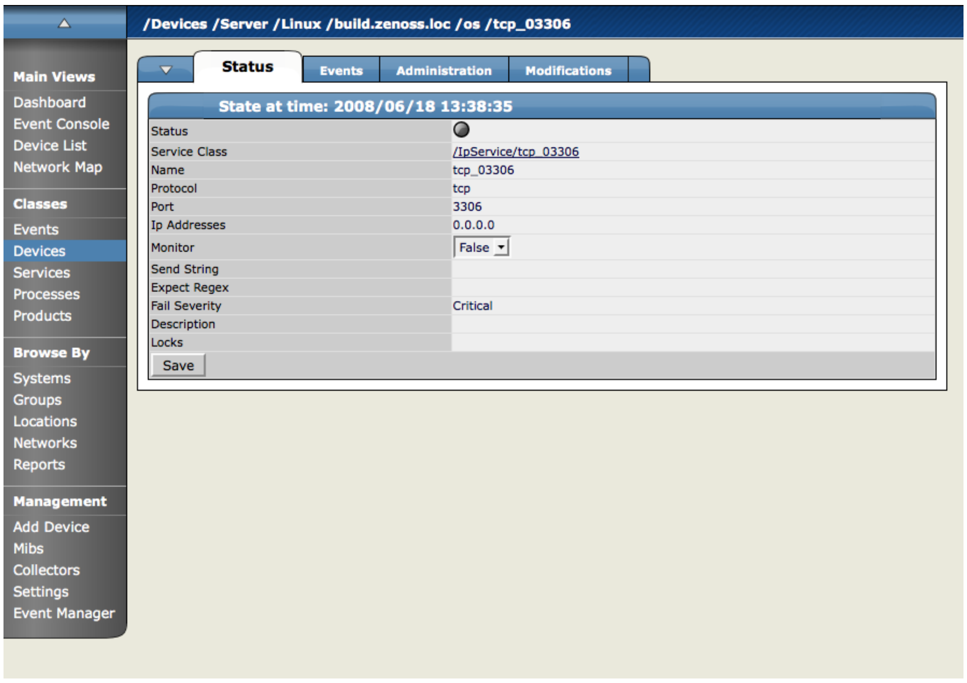
Mount	Total bytes	Used bytes	Free bytes	% Util	Lock
/	73.4GB	61.3GB	12.1GB	83	
/backups	75.5GB	47.9GB	27.6GB	63	
/boot	98.7MB	8.3MB	90.4MB	8	

Routes

2. In the IP Services area, click the link to the service you want to monitor.

   The service summary for the service you have selected appears.

Figure 13.6. Service Summary



/Devices /Server /Linux /build.zenoss.loc /os /tcp_03306

State at time: 2008/06/18 13:38:35

Status	●
Service Class	/IpService/tcp_03306
Name	tcp_03306
Protocol	tcp
Port	3306
Ip Addresses	0.0.0.0
Monitor	False
Send String	
Expect Regex	
Fail Severity	Critical
Description	
Locks	

3. From here set the Monitored flag to True to monitor this service for only this machine. You can also set this service to be monitored system-wide.

4. To monitor a service system wide, click the Click the Service Class link.

   This page will show you everywhere the service is running and whether the service is monitored or not.

5. Click the Edit tab and set Monitored to True.

   This will turn on monitoring for every instance of this service in the system.

6. Click Save.

7. Click the Status tab once again.

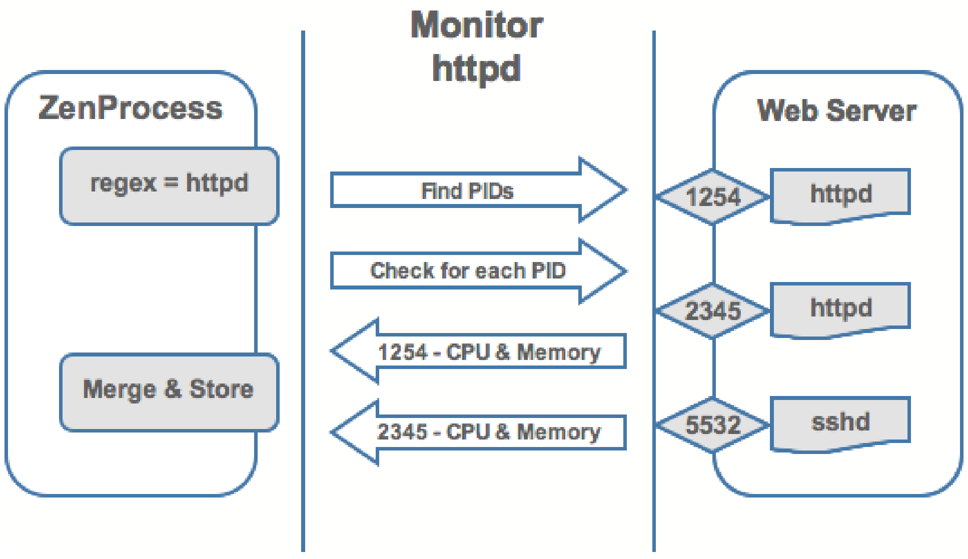
   Most of the instances of the Service will now be set to green, meaning they are monitored and up. The ones that remain unmonitored indicate that they have this service class set to not monitor at a local level.

3. Monitoring Processes

Zenoss is able to monitor any processes running on your network. You can configure Zenoss to monitor process as they occur throughout your system.

Zenoss process monitoring works according to how it appears in the following diagram:

Figure 13.7. Process Monitoring



You can see that ZenProcess uses a regex match to see if it can find PIDS matching the expression to see that these process are running on the selected device.

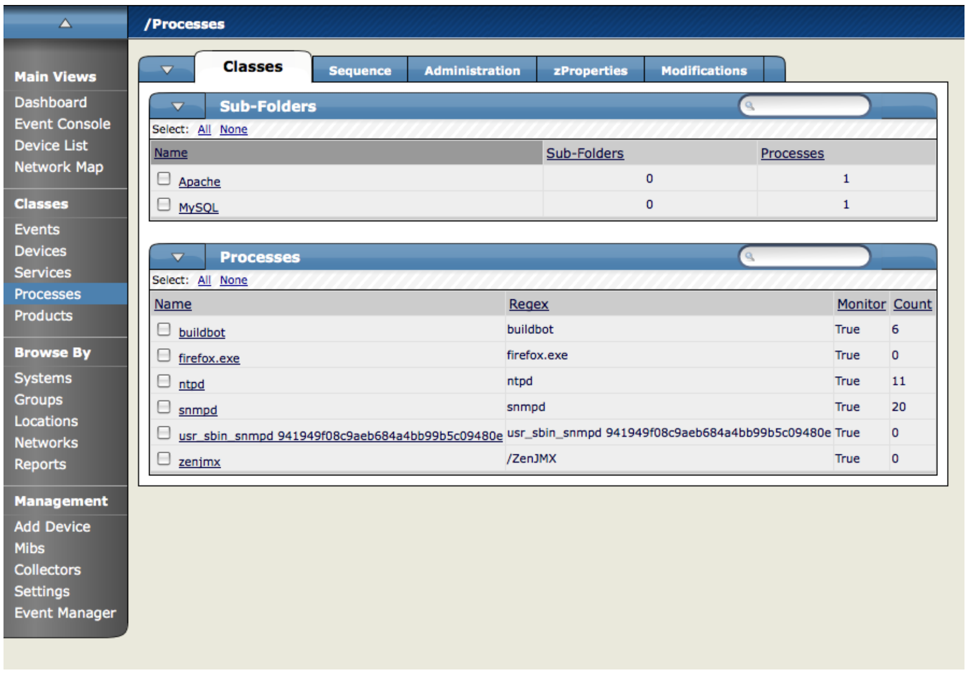
3.1. Adding Processes to Monitor

To add a process to monitor:

1. From the left navigation menu, select Processes.

   The Processes page appears.

Figure 13.8. Processes Page



Sub-Folders

Select: All None

Name	Sub-Folders	Processes
Apache	0	1
MySQL	0	1

Processes

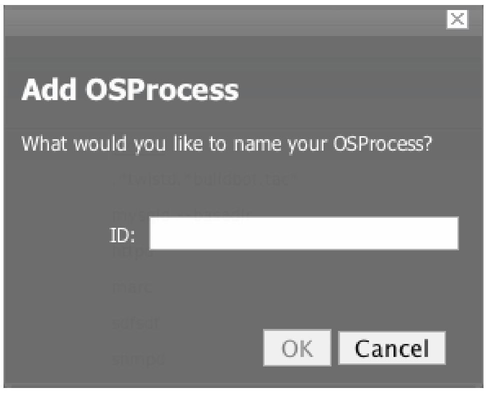
Select: All None

Name	Regex	Monitor	Count
buildbot	buildbot	True	6
firefox.exe	firefox.exe	True	0
ntpd	ntpd	True	11
snmpd	snmpd	True	20
usr_sbin_snmpd 941949f08c9aeb684a4bb99b5c09480e	usr_sbin_snmpd 941949f08c9aeb684a4bb99b5c09480e	True	0
zenjmx	/ZenJMX	True	0

2. From the Processes table menu, select Add Process.

   The Add OS Process Dialog Appears.

Figure 13.9. Add OS Process Dialog



Add OSProcess

What would you like to name your OSProcess?

ID:

3. Enter the regular expression (regex) name of the process you want to monitor in the Processes field and click the OK button.

   The process is added and the Processes window re-appears showing the process you just entered.

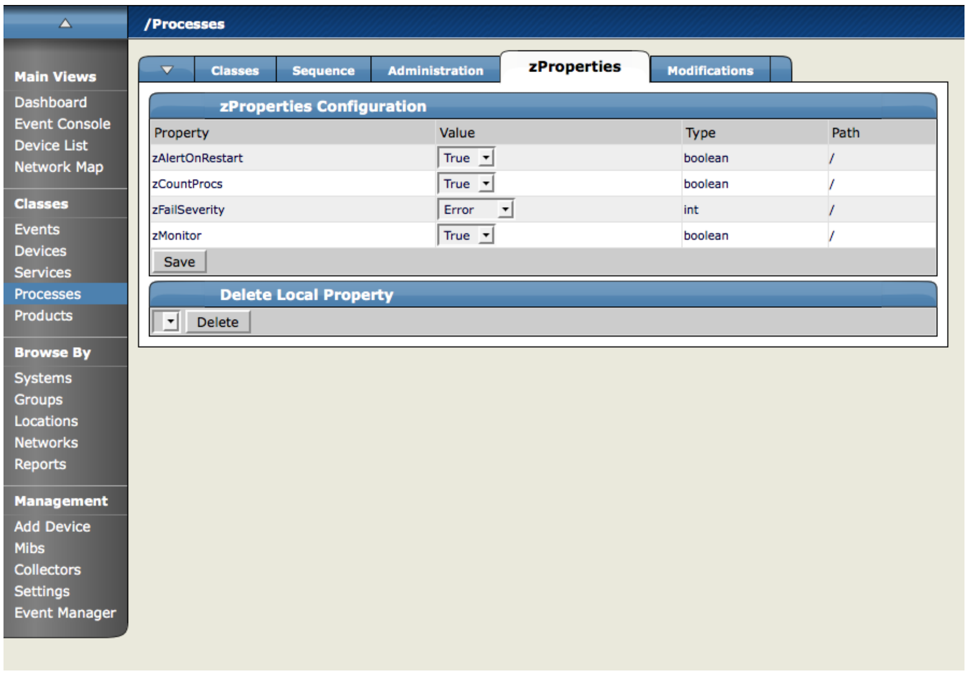
   Now you are monitoring this process, so after a remodel (which you can do manually or it occurs at 6 hour intervals), it will show every device (occurrence) where this process is running. As such, the process is now being monitored wherever it occurs.

   Clicking on a specific process will take you to an interface that shows all instances of that process running across machines that have it monitored. If the process has multiple instances the Zenoss will monitor the sum of CPU and memory utilization of all processes as well as the count of total processes running. However, if the process has only a single instance, CPU utilization and memory usage are graphed for the single process. To perform process monitoring the device SNMP agent must have a reasonable HOST-RESOURCES MIB.

3.2. Configuring Process zProperties

You can configure zProperties either for all processes for an individual process or for any processes that fall further down in the process hierarchy tree. To configure them for all processes click the zProperties tab from the Process Overview tab. To configure zProperties for an individual process, click on the process name in the Process Overview and then click the zProperties tab for that process. The Process zProperties Tab appears.

Figure 13.10. Processes zProperties tab



Property	Value	Type	Path
zAlertOnRestart	True	boolean	/
zCountProcs	True	boolean	/
zFailSeverity	Error	int	/
zMonitor	True	boolean	/

You can configure the following zProperties for either all processes or the selected process from this tab:

- ZAlertOnRestart
- ZCountProcs
- ZFailSeverity
- zMonitor

  For more information about the Services zProperties, see the zProperties Appendix.

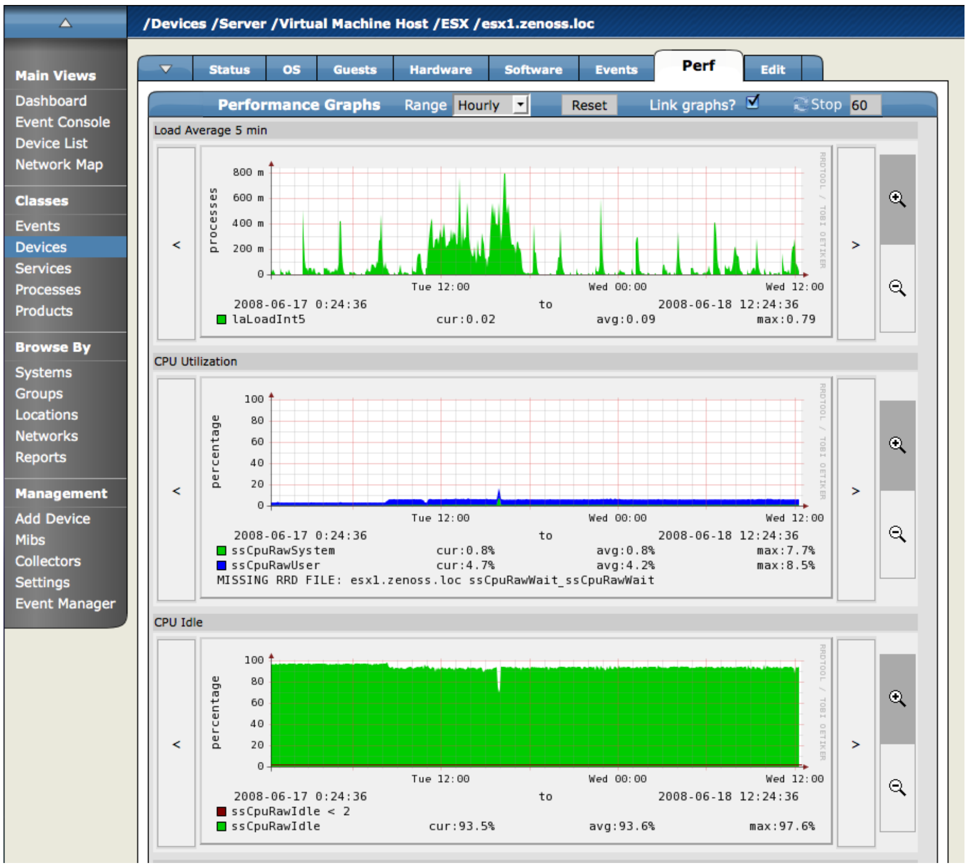
Chapter 14. Performance Monitoring

1. Performance Monitoring

Zenoss includes several methods for monitoring performance metrics of devices and device components. ZenPerf-SNMP collects data via SNMP from any device configured properly for SNMP monitoring. ZenCommand can login to devices via telnet or ssh and run scripts to collect performance data. ZenPacks can provide additional means of collecting performance data. Examples include ZenJMX; which collects data from enterprise Java applications, and HttpMonitor; which checks the availability and responsiveness of web pages. Regardless of the monitoring method used, configuration information is stored in Performance Templates.

The following image shows that graph as it appears in the Perf tab.

Figure 14.1. Perf Tab showing Load Average Graph



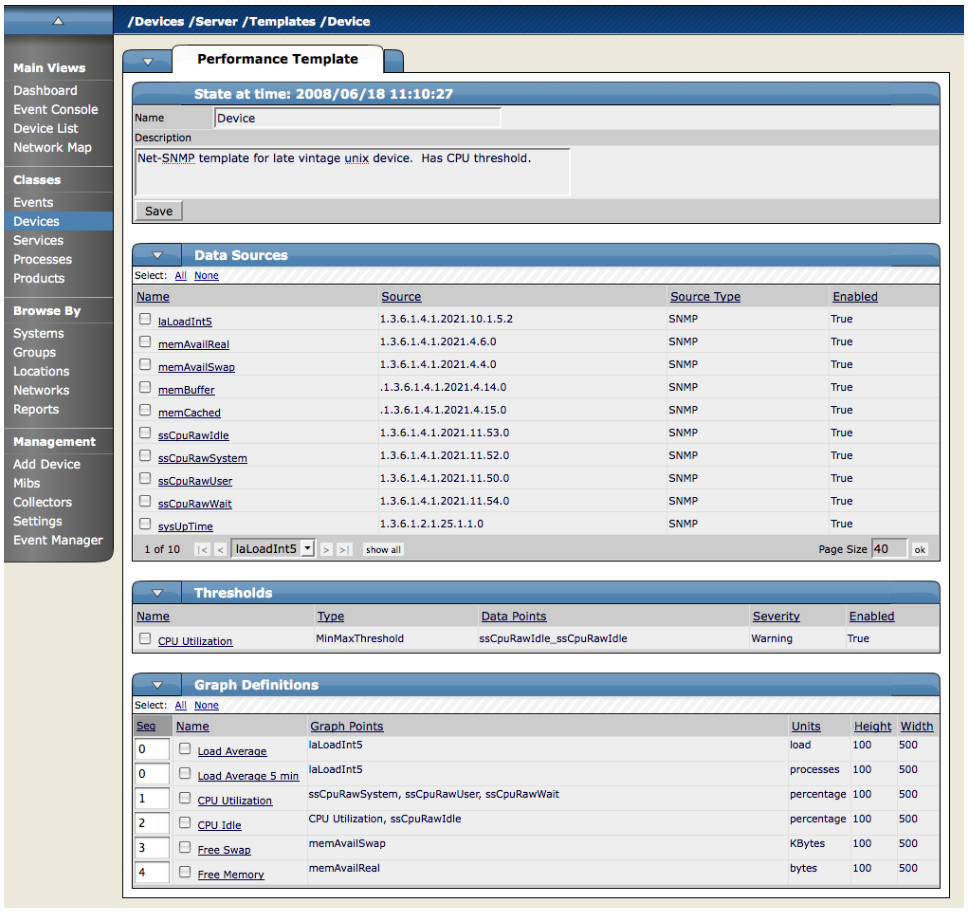
```
Load Average 5 min
2008-06-17 0:24:36          to          2008-06-18 12:24:36
laLoadInt5            cur:0.02      avg:0.09       max:0.79

CPU Utilization
2008-06-17 0:24:36          to          2008-06-18 12:24:36
ssCpuRawSystem        cur:0.8%      avg:0.8%       max:7.7%
ssCpuRawUser          cur:4.7%      avg:4.2%       max:8.5%
MISSING RRD FILE: esx1.zenoss.loc ssCpuRawWait_ssCpuRawWait

CPU Idle
2008-06-17 0:24:36          to          2008-06-18 12:24:36
ssCpuRawIdle < 2
ssCpuRawIdle          cur:93.5%     avg:93.6%      max:97.6%
```

The Performance Template for this Load Average 5 minutes graph (as for any other graph appearing on the Perf tab) can be found by opening the Device page menu, choosing More.. and then the Templates option and then clicking the name of the template.

2. Performance Templates

Performance Templates define the specific instructions for performance data collection for devices and device components. Templates contain three types of sub objects: Data Sources, Thresholds and Graph Definitions. Data Sources specify exactly which data points to collect and what method to use to collect them. Thresholds define expected bounds for collected data and specify events to be created in Zenoss if the data does not match those bounds. Graph Definitions describe how to graph the collected data on the device or device components Performance or Status page.

Figure 14.2. Performance Template for Load Average Graph



State at time: 2008/06/18 11:10:27

Name: Device

Description: Net-SNMP template for late vintage unix device. Has CPU threshold.

Data Sources

Name	Source	Source Type	Enabled
laLoadInt5	1.3.6.1.4.1.2021.10.1.5.2	SNMP	True
memAvailReal	1.3.6.1.4.1.2021.4.6.0	SNMP	True
memAvailSwap	1.3.6.1.4.1.2021.4.4.0	SNMP	True
memBuffer	.1.3.6.1.4.1.2021.4.14.0	SNMP	True
memCached	.1.3.6.1.4.1.2021.4.15.0	SNMP	True
ssCpuRawIdle	1.3.6.1.4.1.2021.11.53.0	SNMP	True
ssCpuRawSystem	1.3.6.1.4.1.2021.11.52.0	SNMP	True
ssCpuRawUser	1.3.6.1.4.1.2021.11.50.0	SNMP	True
ssCpuRawWait	1.3.6.1.4.1.2021.11.54.0	SNMP	True
sysUpTime	1.3.6.1.2.1.25.1.1.0	SNMP	True

Thresholds

Name	Type	Data Points	Severity	Enabled
CPU Utilization	MinMaxThreshold	ssCpuRawIdle_ssCpuRawIdle	Warning	True

Graph Definitions

Seq	Name	Graph Points	Units	Height	Width
0	Load Average	laLoadInt5	load	100	500
0	Load Average 5 min	laLoadInt5	processes	100	500
1	CPU Utilization	ssCpuRawSystem, ssCpuRawUser, ssCpuRawWait	percentage	100	500
2	CPU Idle	CPU Utilization, ssCpuRawIdle	percentage	100	500
3	Free Swap	memAvailSwap	KBytes	100	500
4	Free Memory	memAvailReal	bytes	100	500

2.1. The Templates Page

The Templates page lists all the Templates available to a particular Device or Device Class. For Devices Classes you can get to this page via the Templates tab. For Devices you use the page menu->More->Templates menu item. The Templates page shows Performance Templates defined that that particular Device or Device Class and all those defined further up the Device Class hierarchy. If more than one Template of the same name is defined then only the one to which this Device or Device Class can bind is shown (the others have been overridden by that one.)

3. Template Binding

Templates can be defined on Devices Classes and on individual devices. Before Zenoss begins performance collection for a device or component it must determine which Templates apply. This process is called template binding. The first step to binding is determining the list of Template names that apply to this device or component. For device components this is usually just the meta type of the component (e.g. FileSystem, CPU, HardDisk, etc.) For devices this list is the zDeviceTemplates zProperty. The second step to binding is finding Templates that match these names. For each name Zenoss searches first the Device then up the Device Class hierarchy looking for a template with that name. Zenoss uses the first template it finds with the correct name, ignoring others of the same name that might exist farther up the Device Class hierarchy.

The Templates page for any Device or Device Class shows the templates available for binding. This page shows a list of all templates that are defined at this point or higher in the Device hierarchy. Use the Bind Templates menu item to bring up the Bind Performance Templates dialog. This dialog lets you view and change which templates are currently bound. You can edit the list of template names a device will bind to either by using the Bind Templates dialog or by editing the zDeviceTemplates zProperty directly on the zProperties page. There is no way to edit the bound name for a device component.

Name Binding	Definition
Device	The device object itself. (These OIDs don't have an snmp index number.)
FileSystem	The file system object currently uses the host resources MIB.
Interface	Interfaces are bound using their interface type. For example: ethernetCsmacd
HardDisk	Hard disk object for I/O stats such as Windows boxes with Informant MIB.

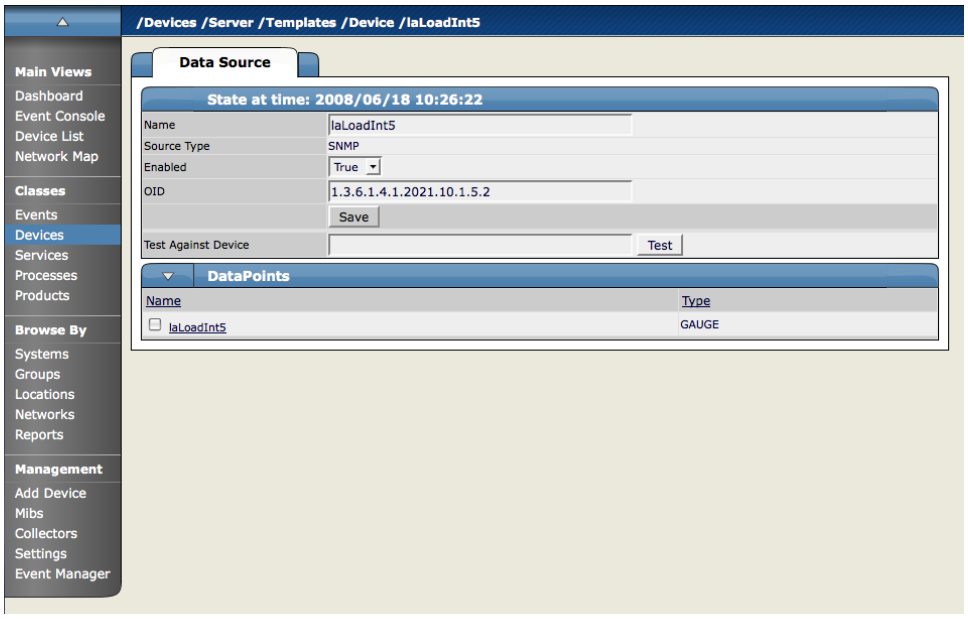
4. Data Sources

Data Sources specify exactly which data points to collect and how to collect them. Every Performance Template has one or more Data Sources. Zenoss provides two built in types of Data Sources: SNMP and COMMAND. Other types can be provided through ZenPacks. When creating a new Data Source with the Add New DataSource menu item you choose from a list of available Data Source types. All Data Source types will have a Name field and an Enabled True/False option. Which other configuration options are presented depends on the type of Data Source.

SNMP Data Sources define data to be collected via SNMP by the ZenPerfSNMP daemon. They contain one additional field to specify which SNMP OID to collect. Note that many OIDs need to end in .0 in order to work. Because SNMP Data Sources specify exactly one performance metric they usually have exactly one Data Point (see below.) See SNMP Monitoring section for further details.

COMMAND Data Sources specify data to be collected by a shell command that is executed on either the Zenoss server itself or on the monitored device via a telnet or ssh session. The ZenCommand daemon processes COMMAND Data Sources. A COMMAND Data Source might return one or more performance metrics and will usually have one Data Point for each metric (see below.) Shell commands used with COMMAND Data Sources must return data that conforms to the Nagios plug-in output specification. See the Monitoring Using ZenCommand section for further details.

Figure 14.3. Data Source



State at time: 2008/06/18 10:26:22

Name	laLoadInt5
Source Type	SNMP
Enabled	True
OID	1.3.6.1.4.1.2021.10.1.5.2
Test Against Device	

Name	Type
laLoadInt5	GAUGE

5. Data Points

A single Data Source might return data on one or more performance metrics. Data Sources can contain one or more Data Points to represent each individual metric retrieved by the Data Source. Data Points are created via the Add DataPoint menu item when viewing a Data Source. Each Data Point has the following fields:

- Name - In the case of an SNMP Data Point this is usually the same name as the Data Source. For COMMAND Data Points the name should be the same name used by the shell command in returning the data. See the plug-in output specification mentioned above for details.

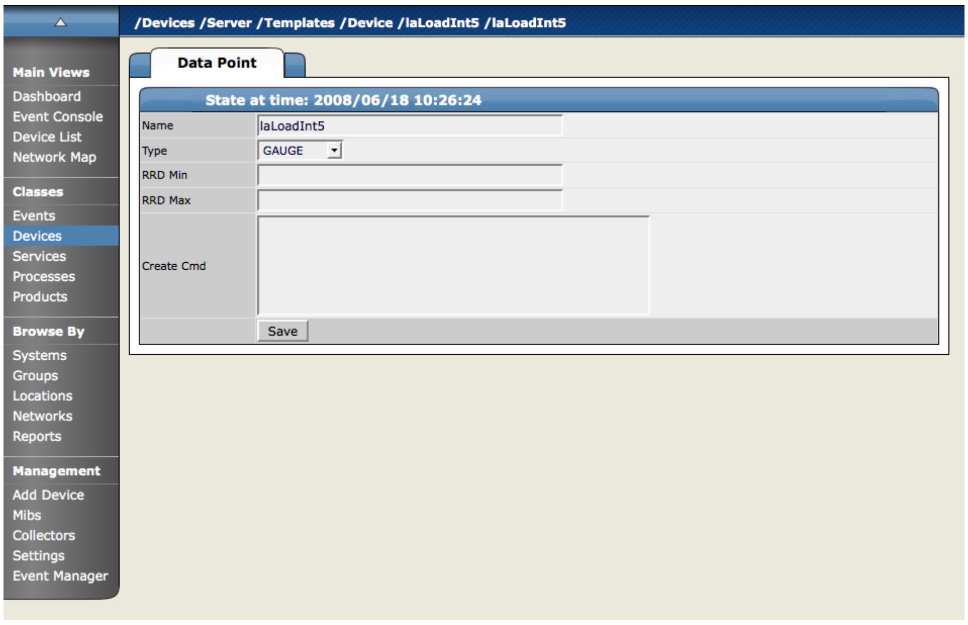
- Type - Zenoss uses RRDTool to store performance data. This field specifies which RRD data source type to use in storing the data for this Data Point. The available options are COUNTER, DERIVE, ABSOLUTE, GAUGE. A source declared as COUNTER will save the rate of change of the value over a step period. This assumes that the value is always increasing (the difference between the current and the previous value is greater than 0). Traffic counters on a router are an ideal candidate for using COUNTER. DERIVE is the same as COUNTER, but it allows negative values as well. If you want to see the rate of change in free disk space on your server, then you might want to use the DERIVE. ABSOLUTE also saves the rate of change, but it assumes that the previous value is set to 0. The difference between the current and the previous value is always equal to the current value. Thus it just stores the current value divided by the step interval (300 seconds in our example). GAUGE does not save the rate of change. It saves the actual value itself. There are no divisions or calculations. Memory consumption in a server is a typical example of gauge. When debating choosing COUNTER you may want to instead choose DERIVED and then a datapoint with a minimum of zero. This creates the same conditions as the COUNTER with one notable exception. COUNTER is a ‚Äúsmart‚Äù data type. It can wrap the data when a maximum number of values is reached in the system. An issue that can sometimes arise is a case when there is a loss of reporting and the system (when looking at COUNTER values) thinks it should wrap the data. This creates an artificial spike in the system and will create statistical anomalies.

- RRDMin - Any value received less than this number will be ignored.

- RRDMax - Any value received greater than this number will be ignored.

- Create Cmd - RRD expression used to create the database for this Data Point. If this is left blank then Zenoss will use a reasonable default that is appropriate for most situations. Details on the create command can be found at http://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html

Figure 14.4. Data Points



State at time: 2008/06/18 10:26:24

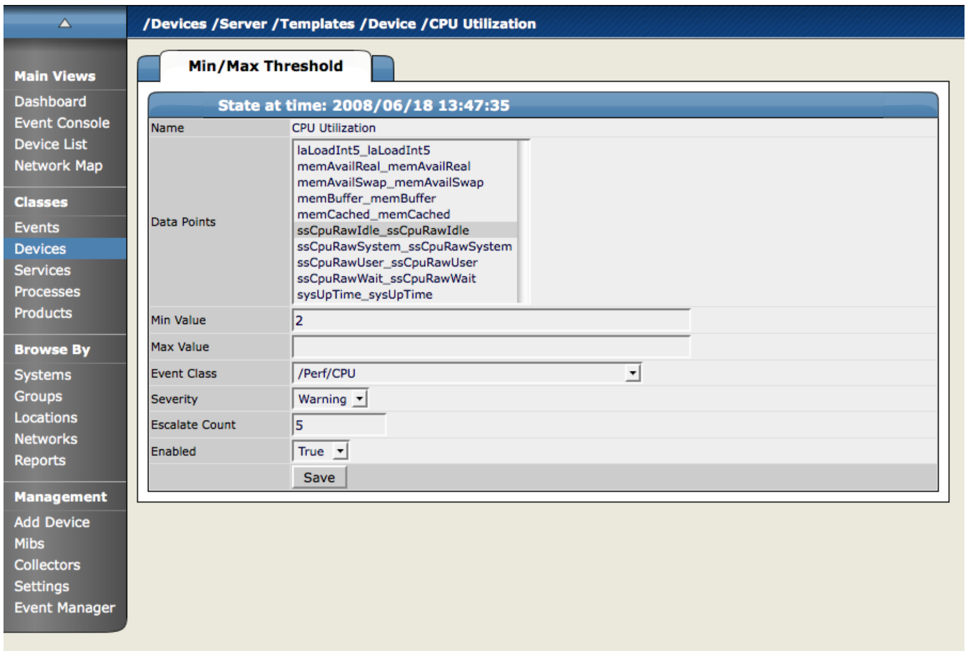
Name	laLoadInt5
Type	GAUGE
RRD Min	
RRD Max	
Create Cmd	

6. Thresholds

Thresholds define expected bounds for Data Points. When a Data Point goes outside a Threshold a Zenoss event is created. Zenoss provides one built in type of Threshold, the Min/Max Threshold. Mix/Max Thresholds inspect incoming data to see if it exceeds a given maximum or falls below a given minimum. ZenPacks can provide other types of Thresholds. The fields for a Min/Max Threshold are:

- Name - This name is displayed on the Performance Template page and is used in creating Threshold events.
- Data Points - Select one or more of the Data Points from this Template to which this Threshold should apply.
- Min Value - If this field is not empty then any time one of the selected Data Points falls below this value an event will be triggered. The field may contain a number or it may contain a Python expression. When using a python expression the variable "here" is references the device or component for which data is being collected. For example, an 85% threshold on an interface might be specified as: here.speed * .85 / 8. (The division by 8 is because interface speed is frequently reported in bytes/sec where the performance data is frequently bytes/sec.)
- Max Value - If this field is not empty then any time one of the selected Data Points goes above this value an event will be triggered. Like Min Value above, it may contain a number or a Python expression.
- Event Class - The event triggered when this Threshold is breached will be of this Event Class.
- Severity - The first event triggered when this Threshold is breached will have this severity.
- Escalate Count - If the threshold is broken this many consecutive times the severity of the event will be escalated one step.
- Enabled - Allows you to enable or disable this Threshold.

Figure 14.5. Threshold Definition



State at time: 2008/06/18 13:47:35

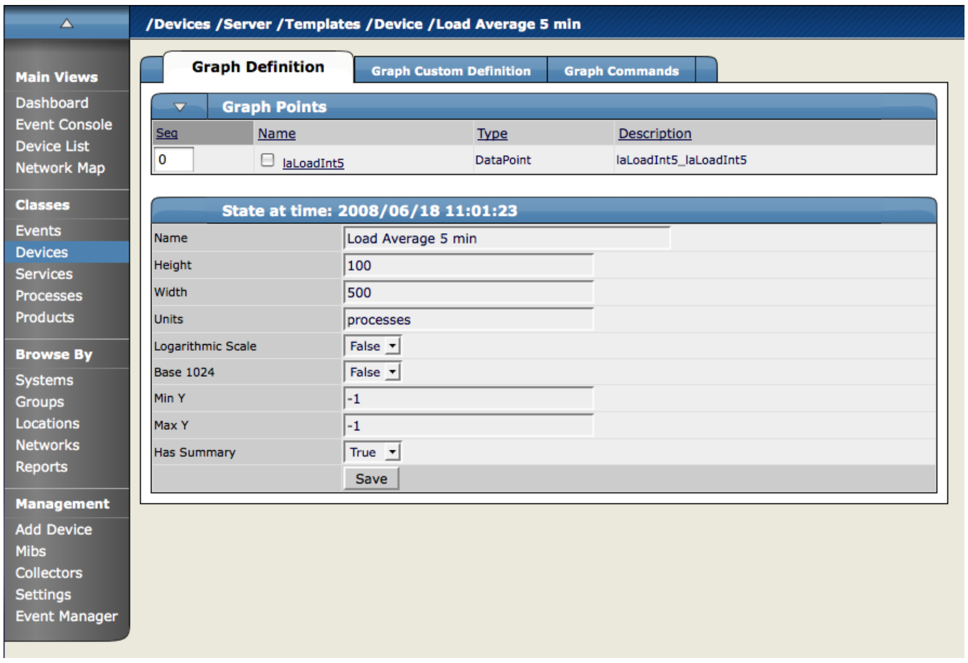
Name	CPU Utilization
Data Points	laLoadInt5_laLoadInt5 memAvailReal_memAvailReal memAvailSwap_memAvailSwap memBuffer_memBuffer memCached_memCached ssCpuRawIdle_ssCpuRawIdle ssCpuRawSystem_ssCpuRawSystem ssCpuRawUser_ssCpuRawUser ssCpuRawWait_ssCpuRawWait sysUpTime_sysUpTime
Min Value	2
Max Value	
Event Class	/Perf/CPU
Severity	Warning
Escalate Count	5
Enabled	True

7. Graph Definitions

A Graph Definition contains the settings to produce a graphic representation of performance data on the device's Perf page or on the component's Status page. These graphs can include any of the Data Points or Thresholds from this Template. New Graph Definitions are created via the "Add Graph..." menu item on any Performance Template page. There are several settings on the Graph Definition that determine the appearance of the graph:

- Name - The name of the graph as it will appear on the Perf or Status page.
- Height - The height of the graph in pixels.
- Width - The width of the graph in pixels
- Units - The label for the graph's vertical axis.
- Log - If True then the scale of the vertical axis will be logarithmic, if False then it is linear. This is useful in situations where the data being graphed grows exponentially. Note that only positive data can be graphed logarithmically.
- Base 1024 - Select True if the data you are graphing is measured in multiples of 1000 or 1024.
- Min Y - This sets the bottom value for the graph's vertical axis.
- Max Y - This sets the top value for the graph's vertical axis.
- Has Summary - If True then a summary of the data's current, average and maximum values is shown at the bottom of the graph.

Figure 14.6. Graph Definition



/Devices /Server /Templates /Device /Load Average 5 min

Graph Points

Seq	Name	Type	Description
0	laLoadInt5	DataPoint	laLoadInt5_laLoadInt5

State at time: 2008/06/18 11:01:23

Name	Load Average 5 min
Height	100
Width	500
Units	processes
Logarithmic Scale	False
Base 1024	False
Min Y	-1
Max Y	-1
Has Summary	True

7.1. Graph Points

Each Data Point or Threshold that is part of a graph is represented by a Graph Point. Several types of Graph Points are available that correspond to Data Points, Thresholds or Custom graphing commands. You can add as many Graph Points as you like to a Graph Definition. The Sequence field determines the order in which the Graph Points are drawn on the graph. You can edit the Sequence numbers and reorder the Graph Points using the Re-sequence Graph Points menu item. Note that Thresholds are always drawn before other Graph Points.

7.1.1. Data Point Graph Points

DataPoint Graph Points draw the value of Data Points from the Template on a graph. Add a new DataPoint Graph Point to the Graph Definition using the "Add DataPoint..." menu item. The Add GraphPoint dialog will appear allowing you to select one or more Data Points defined in this Template. One DataPoint Graph Point will be created for each DataPoint you select in this list. Additionally, there is a Include related thresholds checkbox. If this is selected then any Graph Points will be created for any Thresholds that have been applied to the selected Data Points as well. Clicking on the name of a Graph Point will take you to its edit page. This page will have different fields for different types of Graph Points. DataPoint Graph Points have the following fields:

- Name - This is the name that appears on the Graph Definition page and by default is also the name that will be used in the graph legend.

- Consolidation - This is the RRD function used to graph the Data Point's data to the size of the graph. For most purposes the default value of AVERAGE is appropriate.

- RPN - You may enter an RPN expression that alters the value of the data being graphed for the Data Point. For example, if the data is stored as bits but you would like to graph it as bytes you could use an RPN of "8,/" to divide by 8. You can read more details of RRDTool RPN notation at http://oss.oetiker.ch/rrdtool/tut/rpntutorial.en.html

- Limit - You can specify a maximum value for the data being graphed.

- Line Type - Select Line to have the data graphed as a line. Selecting Area will cause the area between the line and the horizontal axis to be filled with the same color as the line. You can select None if you want to use this Data Point for Custom RRD commands but don't want it to be explicitly drawn.

- Line Width - Enter the pixel width of the line.

- Stacked - If True then the Line or Area is drawn above the previously drawn data. At any point in time on the graph the value plotted for this data is the sum of the previously drawn data and the value of this Data Point at this time. For example, if you were measuring packets in and packets out you could stack them to get an idea of the total number of packets.

- Color - You may specify a color for the Line or Area. Some color names are recognized but more reliably you can use a three or six digit hex value.

- Format - This is the RRD format to use when displaying values in the graph summary. Information on RRDTool formatting strings can be found at http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html

- Legend - This is the name that will be used for this data in the graph legend. By default this is a TALES expression that specifies the Graph Point name. The variables available in this TALES expression are here (the device or component being graphed) and graphPoint (the Graph Point itself.)

- Available RRD Variables - This field lists the RRD variables already defined in this Graph Definition that could be used in the RPN field.

7.1.2. Threshold Graph Points

Threshold Graph Points draw the value of Thresholds from the Template on a graph. Add a new Threshold Graph Point to the Graph Definiton using the "Add Threshold..." menu item. The Add GraphPoint dialog will appear allowing you to select one or more Thresholds defined in this Template. One Threshold Graph Point will be created for each Threshold you select in this list. Threshold Graph Points have several settings: Name, Color and Legend. See the section on DataPoint Graph Points above for a description of these fields.

7.1.3. Custom Graph Points

Custom Graph Points allow you to insert specific RRD graph commands into the Graph Definition. For details on DEF, CDEF and VDEF commands see http://oss.oetiker.ch/rrdtool/doc/rrdgraph_data.en.html. For details on the other RRD commands see http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html

7.2. Custom Graph Definition

The Graph Custom Definition tab allows you to specify exactly your own set of RRD commands to draw this graph. The Graph Points specified on the Graph Definition tab are used to define data that is available to the commands you specify here but the Graph Points will not be drawn unless you explicitly draw them with the commands you specify here. The Available RRD Variables lists the values defined by the Graph Points that are available for your use.

7.3. Graph Commands

The Graph Commands tab shows an approximate representation of the RRD commands that will be used to draw this graph. It is mainly useful as a debugging tool when using Custom Graph Points or the Custom Definition tab.

8. Changing the Order of the Graphs

You can customize many of the features of the graphs including things like thresholds. As an example, you can change the sequence of the appearance of the graphs on the page.

1. Navigate to a device in your system.
2. From the page menu, select More >> Templates.
3. Click the “Create Local Copy” button.
4. Click on the name of the template.
5. In the Graphs area of the page, use the Seq boxes to order the graphs on the page.

9. Monitoring A File System and Changing a Threshold

Zenoss also monitors File System Utilization.

To see File system statistics:

1. Navigate to the OS tab for a device you have loaded into the system.
2. Click the “/” File System.

   Here you see a summary of the statistics for the file system for the device.

   You can change the threshold for events to occur in the local template for this file system. This example walks you through creating a low threshold and then generating events based on this new low threshold.
3. Click More and then Templates.
4. Click the Create Local Copy button to make a local copy of the template for this device.
5. In the Thresholds area, in the Add box, enter the name “Really Low Threshold” and click the Add button.

   The Really Low Threshold appears in the list.
6. Click on the link for the Really Low Threshold.
7. Choose usedBlocks use Blocks
8. In the Max Value area, enter the following expression:

   ```
   here.totalBlocks * .05
   ```

   Basically this sets the really low threshold at 5% of usage.
9. Make sure Enabled is set to True.

   You can leave the other values the same since we will gather this data from the same data-source and other attributes.

You have now created a new threshold that you can create and send events based upon.

Chapter 15. Monitoring Devices Remotely via SSH

1. Monitoring Devices Remotely via SSH

You can monitor device remotely via SSH. Monitoring devices remotely basically consists of 2 distinct installations. You must install the Zenoss plugins on each remote device you want to monitor as detailed in the following subsections.

1.1. Installing Zenoss Plugins on the Remote Machine

The Zenoss Plugins are packaged in 2 formats: native format and source distribution. The native format (RPM) is recommended in Red Hat based systems that support Red Hat Package Management. Using the RPM distribution allows system administrators to easily update the package when newer versions are released. The source distribution is assembled using setuptools, and is commonly used by developers rather than system administrators. The benefit of using the source distribution is that you do not need root privileges to install the Zenoss plugins.

1.1.1. Zenoss Plugin Installation Technique: RPM

The RPM for the Zenoss Plugins is a noarch RPM, which means it can be installed on any architecture (i386, amd64, ia_64). The only external dependency needed to install the zenoss plugins RPM is python itself. Most Linux distributions include python in their standard loads.

To install the zenoss plugins RPM use the following command:

```
$ sudo rpm -Uvh zenoss-plugins-*.rpm
```

Where 'zenoss-plugins-*.rpm' equals the latest Zenoss plugin RPM file.

1.1.2. Zenoss Plugin Installation Technique: setuptools

As noted above, the setuptools installation method allows non-root users to install and use the zenoss plugins. The full set of setuptools arguments are supported, but most people will want to use the default build and install commands:

```
$ python setup.py build
```

```
$ sudo python setup.py install
```

The above commands will install the Zenoss Plugins into directories that are accessible to all users. If you are unable to install system software (because you are a nonprivileged user) you can still install and use the Zenoss Plugins. However, you must go through some extra leg work to get the plugins installed. For a more detailed discussion of how to install the plugins using a non-privileged account see the following URL:

http://dev.zenoss.org/trac/wiki/FAQ#HowdoIinstalltheZenossPlugins

An alternative to downloading the source tarball, exploding it, and running setup.py is to use setuptool's built-in command 'easy_install'. To automatically download and install the zenoss plugins using easy_install run the following command:

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

Where 'Zenoss-Plugins' is the name of the current Zenoss plugin file.

1.1.3. Testing the Plugin Installation

The entry point to the Zenoss Plugins is the zenplugin.py command. When run without any arguments, zenplugin.py reports the proper usage of the script providing insight into which options should be run for troubleshooting.

The Zenoss Plugins detect platform specific runtime values using plugins. For example, the CPU plugin for the linux2 platform uses /proc to read values. In comparison, the CPU plugin for the freebsd5 platform uses a different technique. In order to test the installation you will need to determine which plugins are available for your platform. To do this run the following command:

```
$ zenplugin.py --list-plugins
```

After determining a list of supported plugins for your platform run the zenplugin.py with the plugin name as the argument. The following command line illustrates:

```
$ zenplugin.py cpu
```

1.1.4. Troubleshooting the Plugin Installation

1.1.4.1. "Command not found" when running zenplugin.py

If you receive a "command not found" when running the zenplugin.py command check to make sure that the directory zenplugin.py was installed into is included in your PATH. If you installed via rpm you can use the command "rpm -ql zenoss-plugins | grep zenplugin.py". If you installed via setuptools pay close attention to the "Installing..." messages to see the full directory paths.

1.1.4.2. "platform 'XXX' is not implemented. no plugins exists"

This message indicates that no development work has been put towards implementing plugins for your particular platform. If you receive this message and would like the plugins to support your platform mail the output of the following command to the development team:

```
$ python -c 'import sys; print sys.platform'
```

1.1.5. Changing Zenoss to Monitor the Devices Remotely Using SSH

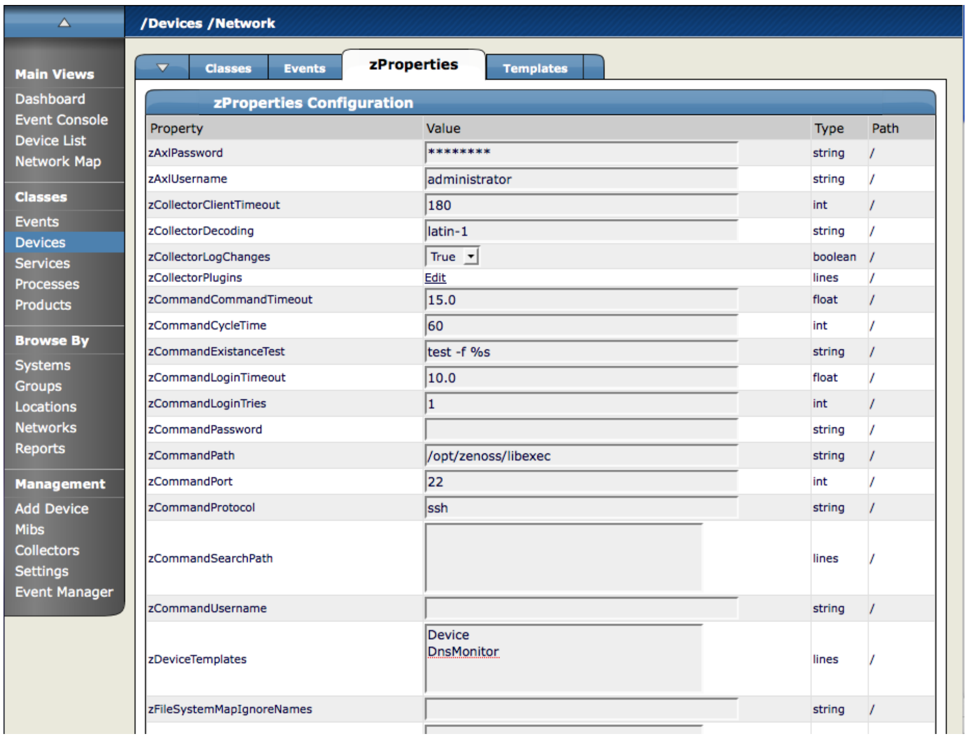
You must change the Zenoss properties for the group where you want to collect remote information using SSH.

1. Navigate to the device class path you want to monitor remotely. You can apply this monitoring per device or per device class path.

2. Change the zProperties value for the group. Click the zProperties tab.

   The zProperties tab appears.

Figure 15.1. Device Group zProperties Tab



Property	Value	Type	Path
zAxlPassword	********	string	/
zAxlUsername	administrator	string	/
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		string	/
zCommandPath	/opt/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device DnsMonitor	lines	/
zFileSystemMapIgnoreNames		string	/

You must make changes to the following zProperties:

- zCollectorPlugins
- zCommandPassword
- zCommandPath
- zCommandUsername
- zSnmpMonitorIgnore
- zTransportPreference

Here are the values we have setup on our remote devices. These have a pre-shared key (with no password) setup from the collector to the remote boxes (it can also use password authorization if the password is entered into zCommandPassword.

zProperties	Value
zCollectorPlugins	snmp|portscan
zCommandPassword	The SSH password for the remote machine.
zCommandPath	The path to zenplugin.py
zCommandUsername	The SSH Username for the remote machine.
zSnmpMonitorIgnore	True
zTransportPreference	command

NOTE: It takes two passes to get full modeling. The first gets the platform type so we know what plugins to run on the second pass which gives us interfaces, filesystems, etc. Run the command:

```
$ zenmodeler run -d enter_server_name_here
```

Run the command a second time to use the plug ins the command gathered on the first pass.

1.1.6. Using the Predefined /Server/Cmd Device Class

The /Server/Cmd device class is an example configuration for modeling and monitoring devices using SSH. The zProperties have been modified as described above, and Device, Filesystem and Ethernet interface templates that gather data over SSH have been created.

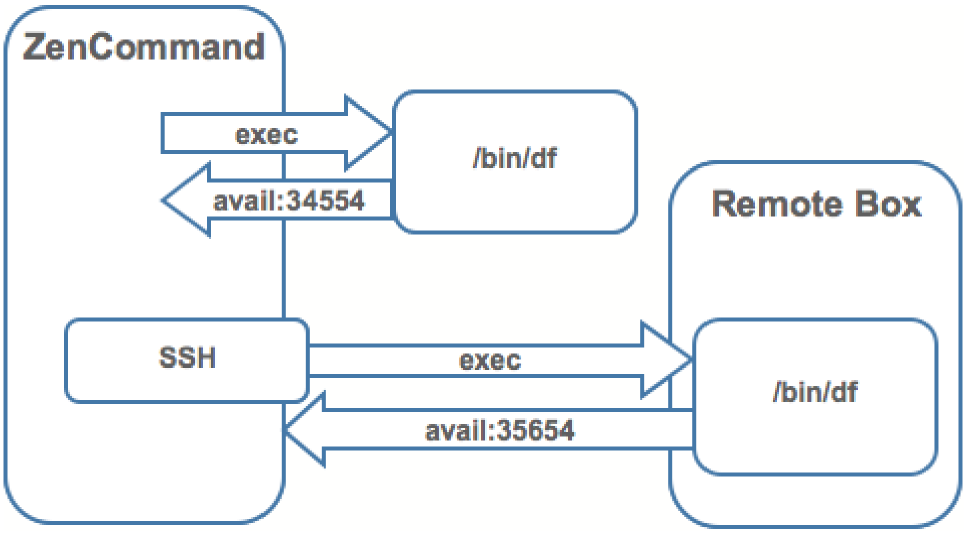
You can use this device class as a reference for your own configuration; or, if you have a device that needs to be modeled or monitored via SSH/Command, you can place it under this device class to use the preconfigured templates and zProperties. You will still need to set the zCommandUsername and zCommandPassword zProperties to the appropriate SSH login information for each device.

Chapter 16. Monitoring Using ZenCommand

1. About ZenCommands

Zenoss has the ability to run Nagios® or Cactii plug-ins though the process ZenCommand. In prior versions of Zenoss, this was called Zenagios. ZenCommand can run plugins locally and remotely using a native SSH transport. When run, Zenoss will track the return code of each plug-in and create events with the output from the plug-in. Zenoss can track performance information from a plug-in also.

Figure 16.1. Running ZenCommands



2. Example: Writing a ZenCommand (check_http example)

You can use a ZenCommand plugin (check_http) to check for specific content of a web page (implicitly checking for server/page 200 status as well).

An example of setting up a ZenCommand plugin to check specific content is:

1. 1.Make sure you are the zenoss user.

2. Test the plugin from the command line. Once its working we will integrate it with Zenoss. check_http -h will display all plugin options. Here is the command to test the product directory.

   ```
   $ZENHOME/libexec/check_http -H www.zenoss.com
   ```

   If the check_http command is correct, the output will look similar to the following.

   ```
   HTTP OK HTTP/1.0 200 OK - 0.723 second response time |time=0.722758s;;;0.000000 size=7932B;;;0
   ```

   You can see that this command plugin returns a valid value we are ready for the next step.

3.

4. Add the device you want to check (one running a www. website) to the Zenoss system setting the discovery protocol to 'none'.

5. Navigate to the device in the Zenoss system and use the page menu to select More and then Templates.

6. Click Create Local Copy button.

   You have now overridden the default Device template with one specific to this device.

7. Drill into the template and remove the sysUpTime data source since we won’t be using SNMP for this device.

8. Add a new description to the template something like “Web testing template”.

9. Add a new Data Source called rootWebCheck.

10. Drill into rootWebCheck and set the following variables:

    - Source Type = COMMAND
    - Component = rootWebCheck
    - Cycle Time = 30

11. Debug your ZenCommand by running zencommand in the foreground with debugging on.

    ```
    zencommand run -d www.website.com -v10
    ```

    Where www.website.com is the site you are trying to monitor.

12. The command template field is a TALES expression so we can do some substitution in our command that will make it generic for any device we add to this class. First let’s set the -H flag to be the IP of the device against which this command will be run.

    ```
    check_http -H ${here/manageIp}
    ```

13. Now let’s add a check looking for content on the page. The -r flag will run a regular expression against the web page to check for text. We will look for three pieces of text:

    ```
    check_http -H ${here/manageIp}-r textstring1
    ```

    Where textsrting1 is some text you know to be displayed on the resulting webpage.

14. For this example, you want this command to be generic, so make a custom field for the regex that can be changed per device. Set the default to “.*” which will match everything. Go to /Devices/Custom Schema and add a new field:

    - Label = Web Match Regex
    - Name = cWebMatchRegex
    - Type = string
    - Default = .*
    - Visible = True

15. Now go back our template and change the command to be.

    ```
    check_http -H ${here/manageIp}-r ${here/cWebMatchRegex}
    ```

16. Now add the regex value into cWebMatchRegex that we used in the example above.

17. Test your ZenCommand from the command line.

3. Collect Data from A ZenCommand

To collect and display some data from the ZenCommand example above, you can log the data to see something like response time in a graphical format.

1. Go to the /Web/Device template.

2. Go to the Data Source you created in the check_http example above.

3. At the bottom there is a table of Data Points add one called “time”

4. The default type of GAUGE is fine so there is no need to modify the Data Point.

5. Test your command again and you should see a log message that starts with:

   ```
   DEBUG:zen.zencommand:storing responseTime = 1.0
   ```

6. Make a graph to display your data. In the Device template, create a graph called “Web Response Time”.

7. For this graph, set the following values:

   - Data Sources = rootWebCheck_responseTime
   - Units = Seconds
   - Min Y = 0

8. Now look at the Perf tab for the device www.website.com (whatever website you were using to check) to see the graph. Graph data won’t show up until collection as run several times. Restart zencommand so that the new configuration will take place immediately.

4. Plugin Format for ZenCommands

Nagios® plugins are configured using a Command Template that is much like the RRDTemplates used for performance monitoring. So a template named “Device” will bind to all devices below the template definition. Within each template is a list of commands that will run. The commands can be any program that follows the Nagios® plug-in standard (inputs are command line arguments output is first line of stdout plus a return code) as defined in:

http://nagiosplug.sourceforge.net/developer-guidelines.html

A Nagios® command has several fields:

- name – name of the command object
- enabled – should this command be used on a give device
- component – the component name to use when zencommand sends events to Zenoss
- event class – the event class to used when sending events to Zenoss
- severity – the default severity to use when sending events to Zenoss
- cycle time – how often a command should be run in seconds

- command template – the command to run

  The command template string is built using Zope TALES. Several variables are passed when evaluating the template. They are:

- zCommandPath – The path to the zencommand plug-ins on a given box it comes from the zProperty zCommand-Path. zCommandPath will be automatically added to a command if a path is absent from the beginning of the beginning of the command.
- devname – The device name of the device against which the command is being evaluated
- dev – The device object against which the command is being evaluated
- here – The context of evaluation. For a device, this is equivalent to dev for a component such as a filesystem or interface this is the component object
- compname – If this command evaluates against a component, this is its name as a string
- now – The current time

  Template values are accessed like shell variables. They are the same as the expression syntax used in the section of this guide called TALES Expressions. Here are some examples:

  Run an http check against all devices using the url /zport/dmd

  ```
  check_http –H ${devname}-u /zport/dmd
  ```

  In a template named FileSystem the following command will run against all FileSystems on a device.

  ```
  check_disk –w 10% -c 5% -p ${compname}
  ```

5. Testing ZenCommands

ZenCommand Data Sources can be tested using the zentestcommand shell script. From the command line, simply run:

```
zentestcommand –d devicename --datasource=datasourcename
```

Where devicename is the device you wish to run the command on and datasourcename is the name of a data source on a template associated with the device. zentestcommand will print the results of the command to stdout.

Chapter 17. Monitoring Windows Devices

1. Device Preparation for Windows Devices

Zenoss requires you to do some device preparation to Windows devices you want to monitor. Make sure the following items have been checked for the devices.

1. Check that the SNMP agent enabled.

   This is done through Add/Remove programs -> Add/Remove Windows Components -> Management and Monitoring Tools.

2. Make sure DCOM is enabled for WMI connections.

3. Optionally, you can use SNMP-Informant to have CPU, Memory and Disk I/O stats.

SNMP-Informant agents grab information from Windows devices via WMI on the server where they are installed and then converts system, state and operational data into SNMP OIDs for broadcast. Zenoss can then process the SNMP OID information and generate Events and alerts based on this information. SNMP-Informant is not necessary in Zenoss Enterprise as this functionality is built in. Not using an agent such as SNMP-Informant will limit the amount of information that Zenoss can process.

2. Setting Windows zProperties

This section will give you the important zProperties to set for getting information from windows servers into Zenoss. For each Windows server, navigate to the zProperties for that device and set the following zProperties:

- zWmiMonitorIgnore - Use this to turn on or off all WMI monitoring. If it is set to False, then Windows monitoring is ON. If Ignore is set to True, then WMI Monitoring is OFF. You can set this at the Server/Windows Class level. That way any device placed in this class has Windows monitoring automatically turned on.

- zWinUser - Must be the local admin. The format for zWinUser is:

  - .\Username - The format to use when the account is a local account.

  - DOMAIN\Username - The format for a Domain account.

- zWinPassword - Enter the password used to remotely login to the Windows machine.

3. Testing WMI on a Windows Server

Use the following steps to test the WMI connections on the Windows server.

1. Run wbemtest

2. Click “Connect…”

3. In the Namespace field enter \\HOST\root\cimv2

4. Enter login information in “User” and “Password” fields.

5. Click the “Query…” button.

6. Enter “select * from win32_service” should return a dialog with list of services on the box.

4. Other Optional Windows Configuration Items

There are a few other options for potentially gathering additional information from Windows devices. If you have any of the following installed, Zenoss can get additional information from:

- Dell Open Manage agent gives detailed OS and hardware information.

- HP Insight Management Agent gives detailed OS and hardware information.

5. Modeling Services on Windows Devices

ZenWin is used for Windows Service Monitoring over WMI. It runs under windows and monitors the up/down availability of windows services.

ZenWinModeler like the ZenModeler uses collector plugins to retrieve data from devices. Rather than the methods used by ZenModeler, the ZenWinModeler uses WMI alone to collect its data. The WinServiceMap WMI plugin is available and is included in zCollectorPlugins on the /Server/Windows device class by default. WinServiceMap retrieves the all the monitorable services on a device whether it is up or down.

All Windows services are, by default, NOT monitored. If you would like to monitor a specific windows service, it is very simple to turn it on.

Navigate to the Windows device and click the OS tab.

Click the service you want to monitor and change the value of monitor to “True”.

If you do not see the service you would like to monitor, click on the WinServices table menu.

Select the “Add WinService...” menu item.

6. Collecting Windows Eventlog Events

ZenEventlog is used collect (WMI) event log events. Use the following zProperties to define how Windows Event Log events are processed and monitored:

- zWinEventLog - This zProperty tells Zenoss whether or not to read the event log into Zenoss.

- zWinEventLogMinSeverity - Sets minimum severity to collect from the win event log. Important to note that the higher the number, the lower the severity. 1 being the most severe and 5 being the least severe.

7. Monitoring Windows Performance with SNMP-Informant

Zenoss can use information from SNMP-informant to get SNMP information from Windows devices. First, you must install the free version of snmp-informant from: http://www.snmp-informant.com

To make sure SNMP Informant is running and setup correctly, run this command to walk the SNMP Informant MIB:

```
snmpwalk -v1 -c<community> <server> 1.3.6.1.4.1.9600
```

This command will return some performance information if SNMP-informant is configured and running correctly.

Once this is configured properly SNMP information is gathered and used by the Zenoss system the same as any other device sending SNMP traps.

8. Running Commands on Windows Servers Using Winexe

You can use winexe commands to run commands on monitored Windows servers from within Zenoss.

Usage:

```
$ZENHOME/bin/winexe [options] //host [command]
```

Options	Use
--uninstall	Uninstall winexe service after remote execution
--reinstall	Reinstall winexe service before remote execution
--system	Use SYSTEM account
--runas=[DOMAIN\]USERNAME%PASSWORD	Run as user (BEWARE: password is sent in cleartext over net)

Help Options	Use
-?, --help	Show this help message
--usage	Display brief usage message

Common samba options	Use
-d, --debuglevel=DEBUGLEVEL	Set debug level
--debug-stderr	Send debug output to STDERR
-s, --configfile=CONFIGFILE	Use alternative configuration file
--option=name=value	Set smb.conf option from command line
-l, --log-basename=LOGFILEBASE	Basename for log/debug files
--leak-report	enable talloc leak reporting on exit
--leak-report-full	enable full talloc leak reporting on exit
-V, --version	Print version

Connection Options	Use
-R, --name-resolve=NAME-RESOLVE-ORDER	Use these name resolution services only
-O, --socket-options=SOCKETOPTIONS	socket options to use
-n, --netbiosname=NETBIOSNAME	Primary netbios name
-W, --workgroup=WORKGROUP	Set the workgroup name
--realm=REALM	Set the realm name
-i, --scope=SCOPE	Use this Netbios scope
-m, --maxprotocol=MAXPROTOCOL	Set max protocol level

Authentication Options	Use
-U, --user=[DOMAIN\]USERNAME[%PASSWORD]	Set the network username

Authentication Options	**Use**
-N, --no-pass	Don't ask for a password
--password=STRING	Password
-A, --authentication-file=FILE	Get the credentials from a file
-S, --signing=on|off|required	Set the client signing state
-P, --machine-pass	Use stored machine account password (implies -k)
--simple-bind-dn=STRING	DN to use for a simple bind
-k, --kerberos=STRING	Use Kerberos
--use-security-mechanisms=STRING	Restricted list of authentication mechanisms available for use with this authentication

Chapter 18. Windows Performance Monitoring (Zenwinperf) (Commercial)

1. About Windows Performance Monitoring

ZenWinPerf is an Enterprise ZenPack that allows performance monitoring of Windows servers without an intermediary Windows server doing the data collection. ZenwinPerf provides the WinPerf Data Source, which uses a Windows performance counter rather than an SNMP OID to specify the value to collect. WinPerf Data Sources are processed by the zenwinperf daemon.

2. Zenwinperf Daemon

The zenwinperf daemon does the actual work of opening the winexe/typeperf connections and sending the data back to Zenoss. This daemon appears on the Zenoss Daemons page and can be started, stopped and restarted from there. From the command line it can be controlled via the zenoss script or by issuing commands directly to the daemon which is symlinked at $ZENHOME/bin/zenwinperf.

2.1. ZenwinPerf zProperties

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

- zWinPerfCycleSeconds - This is how frequently (in seconds) winperf datasources are collected. By default this is set to 300.
- zWinPerfTimeoutSeconds - Deprecated. If no data is collected on a connection after this many seconds the connection is closed and a new one created. This value should be greater than zWinPerfCycleSeconds. By default this is set to 600.
- zWinPerfCyclesPerConnection - Deprecated. This is the number of collection cycles requested of typeperf. After (at most) this many cycles typeperf will exit and zenwinperf will create a new connection to the server and a new call to typeperf. (This value is passed as the -sc value to typeperf.) By default this is set to 12.

2.2. How to Create a WinPerf Data Source

There is an example template with WinPerf Date Sources on the /Devices/Server/Windows device class called ZenWinPerf Example. To create your own WinPerf Data Sources follow these steps:

1. Navigate to either a new or an existing Performance 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 click OK.
3. Enter a Windows performance counter in the Perf Counter field. See below for more details on Windows perf counters.
4. Click the Save button. (Notice that a datapoint is created with the same name as the perf counter you selected.)
5. If you wish you can test the counter by entering a device id in the Test Device field and clicking the Test button.

2.3. Windows Performance Counters

ZenWinPerf uses a Windows utility called typeperf to obtain performance data. Information on performance counters can be found with the typeperf documentation here: http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/nt_command_typeperf.mspx?mfr=true

3. /Device/Server/Windows/WMI Device Class

The ZenWinPerf ZenPack includes a /Device/Server/Windows/WMI class that has several new device templates bound and several collector plugins from ZenPacks.zenoss.WinModelerPlugin enabled.

Any Windows device placed in this device class will use WMI as its primary data modeling and data collection mechanism.

A WMI connection cannot be established without a valid set of credentials. These credentials are set with two zProperties: zWinPassword and zWinUser. Like all Windows credentials, the domain should be specified in the zWinUser entry, so use .\username for an account that isn't in the domain but just on the local computer.

The following collector plugins are enabled by default with this new device class:

- zenoss.wmi.WinServiceMap
- zenoss.wmi.CpuMap
- zenoss.wmi.FileSystemMap
- zenoss.wmi.IpInterfaceMap
- zenoss.wmi.IpRouteMap
- zenoss.wmi.MemoryMap
- zenoss.wmi.ProcessMap
- zenoss.wmi.WindowsDeviceMap
- zenoss.wmi.SoftwareMap

The SoftwareMap plugin collects data from the Win32_Product class. The WMI provider for this class is not installed by default on some versions of Windows, especially the Standard Editions of the Server products. To enable to the modeling of software inventory, the WMI Installer Provider must be installed. This can be done by going to the Add/Remove Programs applet in control panel, choosing Windows Components, selecting Management and Monitoring Tools, and then finally the WMI Installer Provider component.

3.1. Device Templates

The following device templates are bound to the new /Devices/Server/Windows/WMI class: Device, FileSystem, and ethernetCsmacd.

3.1.1. Device Template

The Device template provides the following Data Sources:

- cpuPercentProcessorTime
- diskAvgQueueLength
- memoryAvailableKBytes
- # memoryPagesPerSec

- sysUpTime

And the following graphs:

- CPU
- Free Memory
- Paging
- Avg. Disk Queue Length

3.1.2. File System Template

The File System template includes the following Data Sources:

- diskTime
- freeMegabytes
- freeSpace

And then the following graphs:

- Utilization
- IO ServiceTime

3.1.3. ethernetCsmacd Template

The ethernetCsmacd template includes the following Data Sources:

- bytesReceivedSec
- bytesSentSec
- packetsReceivedErrors
- packetsReceivedSec
- # packetsSentErrors
- packetsSentSec

And then the following graphs:

- Packets
- Errors
- Utilization

Chapter 19. SNMP Monitoring

1. SNMP from the Command Line

OID represent the data points where the data for the graphs comes from. Sometimes the reason that a graph is not appearing is because the OID for the particular graph is not valid for the device. You can test this validity using the command line to see if you can return a value. To test the validity of an OID data point giving performance data:

1. SSH to the Zenoss instance.

   Use Username: root

   Password: zenoss

2. Run the command snmp get for one of the OIDs

   In this case lets use the command:

   ```
   $ snmpget -v1 -cpublic build .1.3.6.1.4.1.2021.4.14.0
   ```

   If the OID is valid it will return a value.

   Here are some basic SNMP Operators to gather certain information.

   a. Walk a basic system MIB.

      ```
      snmpwalk -v1 -cpublic <device name> system
      ```

   b. Walk an interface description

      ```
      snmpwalk -v1 -cpublic <device name> ifDescr
      ```

   c. Get a single value.

      ```
      snmpget -v1 -cpublic <device name> ifDescr.2
      ```

   d. Detailed description of an OID value.

      ```
      snmptranslate -Td RFC1213-MIB::ifDescr
      ```

   e. Convert a name to a raw OID.

      ```
      snmptranslate -On RFC1213-MIB::ifDescr
      ```

   f. Convert a raw OID to a short name

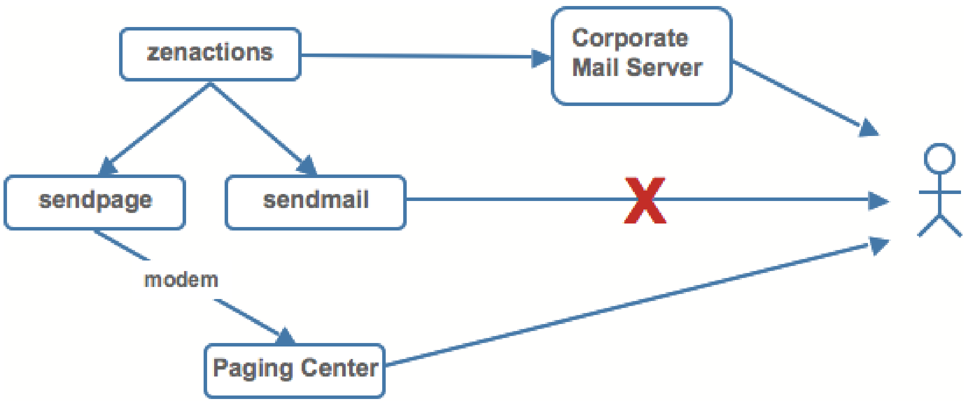
      ```
      snmptranslate -OS .1.3.6.1.2.1.2.2.1.2
      ```

Chapter 20. Alerting Rules

1. Creating and Using Alerts

The daemon ZenActions provides functionality for sending emails or pages based on events received. It continuously evaluates every user's paging rules against the event database. Each user has their own set of alerting rules.

Figure 20.1. Sending Alerts



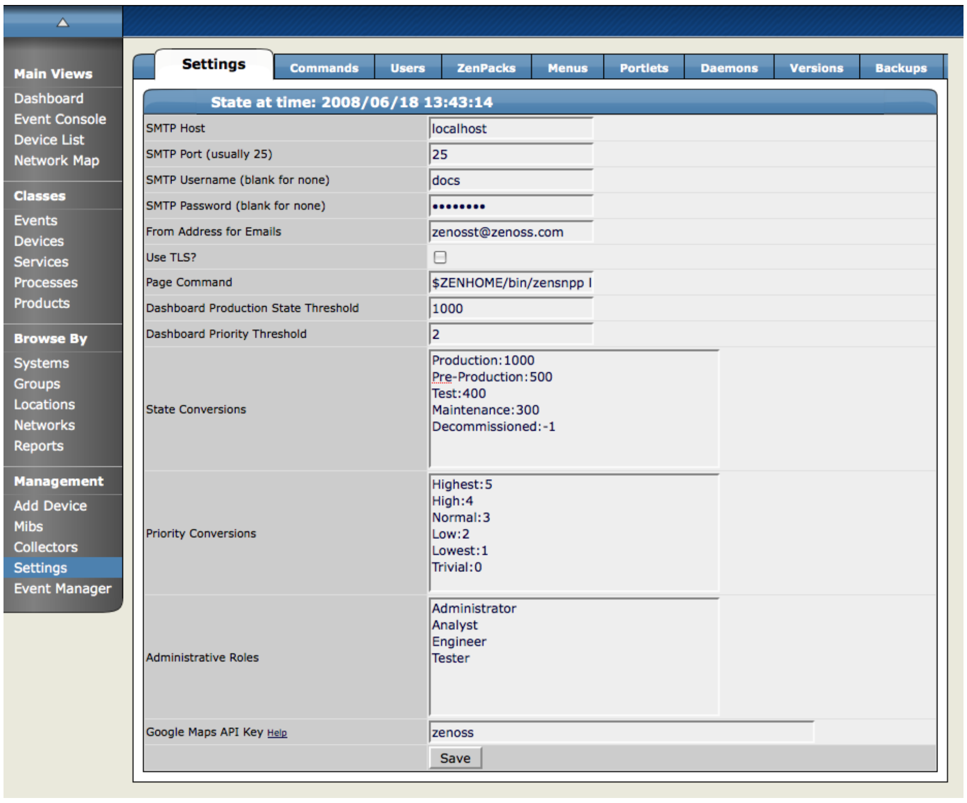
1.1. Setting SMTP Settings For Alerts

To use email and pager alerts, you must have Zenoss pointing to an SMTP relay with the proper settings.

1. From the navigation menu on the left side of the Dashboard, select Settings.

   The Settings page appears.

Figure 20.2. Settings Tab Showing SMTP Settings



State at time: 2008/06/18 13:43:14

SMTP Host	localhost
SMTP Port (usually 25)	25
SMTP Username (blank for none)	docs
SMTP Password (blank for none)	••••••••
From Address for Emails	zenosst@zenoss.com
Use TLS?	☐
Page Command	$ZENHOME/bin/zensnpp l
Dashboard Production State Threshold	1000
Dashboard Priority Threshold	2
State Conversions	Production:1000 Pre-Production:500 Test:400 Maintenance:300 Decommissioned:-1
Priority Conversions	Highest:5 High:4 Normal:3 Low:2 Lowest:1 Trivial:0
Administrative Roles	Administrator Analyst Engineer Tester
Google Maps API Key Help	zenoss

2. To set up the mail servers, you must configure the SMTP Host, the SMTP Port, SNPP Host, and the SNPP Port.

   Now you are prepared to create and use Alerting Rules for the Zenoss system.

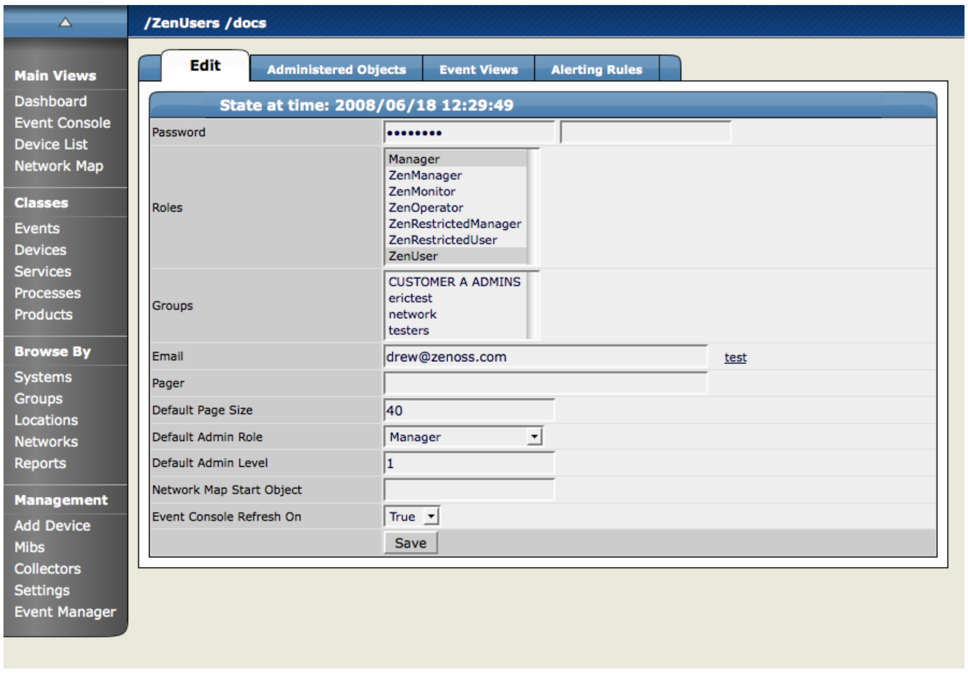
2. Creating a New Alerting Rule

Alerting rules are created on a per user basis. You can add additional recipients for rules, but upon creation, the rules are tied to a user account.

1. From the upper right corner of the Zenoss Dashboard, click the Preferences link.

   The Preferences page appears.

Figure 20.3. Preferences Tab - Edit Tab



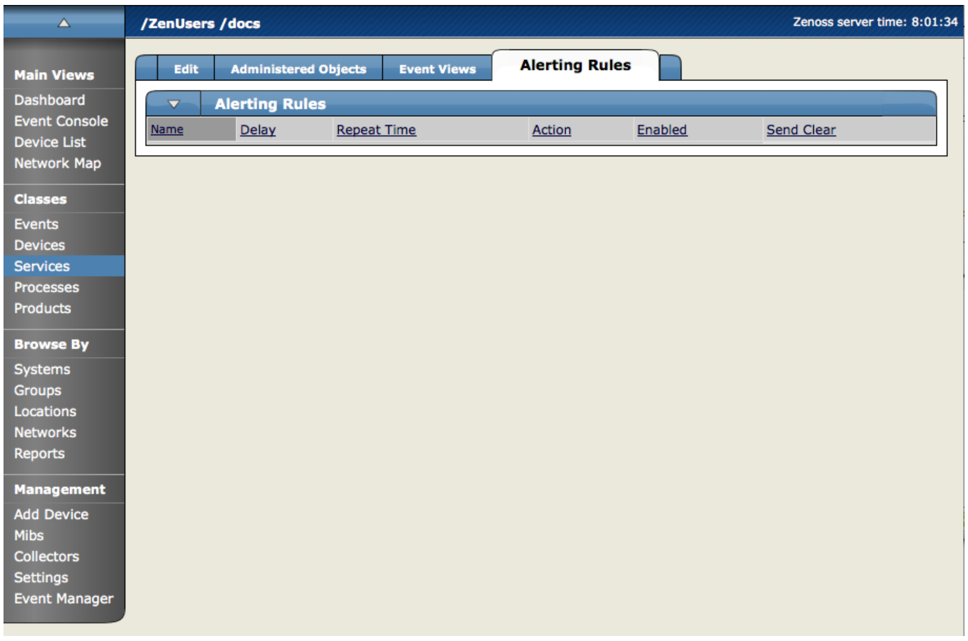
State at time: 2008/06/18 12:29:49

Password	••••••••
Roles	Manager ZenManager ZenMonitor ZenOperator ZenRestrictedManager ZenRestrictedUser ZenUser
Groups	CUSTOMER A ADMINS erictest network testers
Email	drew@zenoss.com
Pager	
Default Page Size	40
Default Admin Role	Manager
Default Admin Level	1
Network Map Start Object	
Event Console Refresh On	True

2. Select the Alerting Rules tab.

   The Alerting Rules tab appears.

Figure 20.4. Alerting Rules Tab

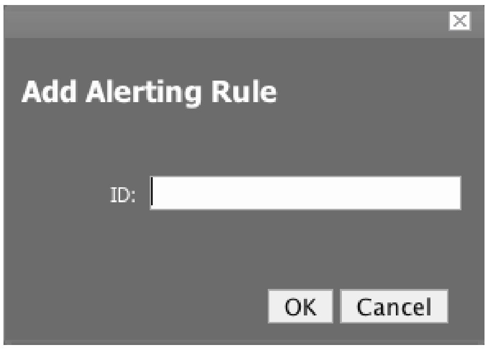


Name	Delay	Repeat Time	Action	Enabled	Send Clear

3. From the Alerting rule table menu, select Add Alerting Rule.

   The Add Alerting Rule dialog appears.

Figure 20.5. Add Alerting Rule Dialog



4. In the ID field, enter a name for the alert.

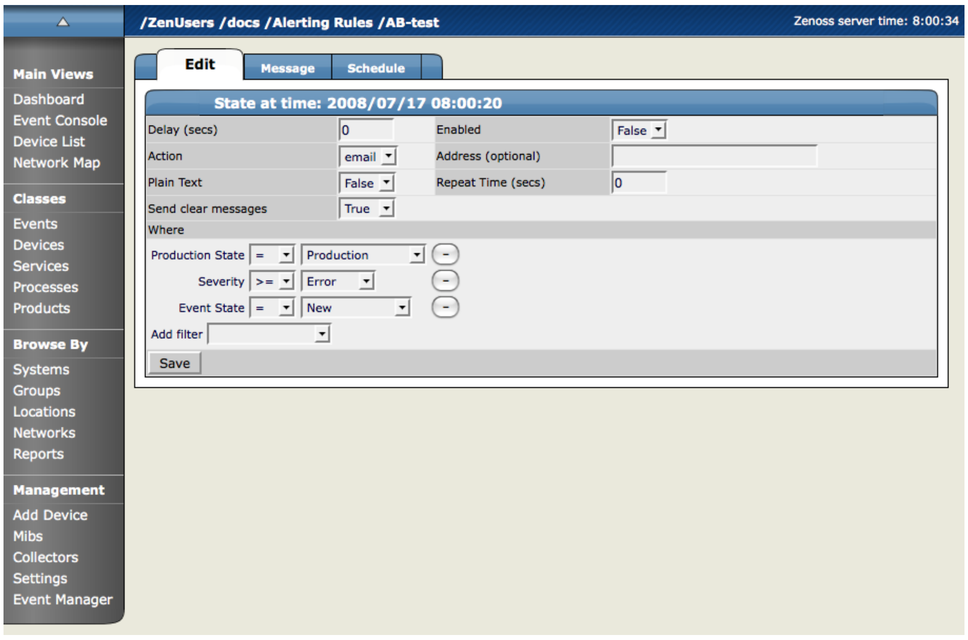
5. Click the OK button.

   The main Alerting Rules page appears showing the alert you just created.

6. Click the name of the Alert you just created.

   The Alert Details page appears.

Figure 20.6. Alert Details Edit Page



State at time: 2008/07/17 08:00:20

Delay (secs)	0	Enabled	False
Action	email	Address (optional)	
Plain Text	False	Repeat Time (secs)	0
Send clear messages	True		

Where

Production State	=	Production
Severity	>=	Error
Event State	=	New
Add filter		

2.1. Define and Enable This Alert

Set the attributes from the Alert Details page, and clicking the Edit tab.

1. Use the Delay field to set the number of seconds to wait before sending the alert. If an event clears before delay time no alert is sent.

2. To enable the alert, set Enabled to True.

3. Use the Repeat Time to set the time for repeating the alert to send the alert every x seconds until the event is acknowledged.

4. In the Action field, select whether you want the system to send email or a page.

   If action is defined as email the event will be emailed. If the default action is set to page, you will need to have an SNMP paging server setup (see the external libs dir of install directory and the sendpage lib does this). Many wireless phone systems have SMTP to SMS gateways so in many cases you use email to act like a page as well.

   By default email alerts will be sent to the email address for this user and pager alerts will go to the specified pager address.

   You can override this by filling in the Address (optional) field.

5. The Where area of the tab sets the thresholds for the Alert.

   The default rule that is created contains the thresholds for an event occurrence where the Event State is “New”, Severity is “greater than Error”, and Production State is “Production”. You can change these thresholds by changing the values in the pop-up menus.

6. You can also add additional filters to the Where area by choosing a filter from the Add Filter menu. Adding a filter creates an additional pop-menu in the Where area where you can choose additional values to filter the event. To Remove any of the filters for the alert, click the (-) minus button.

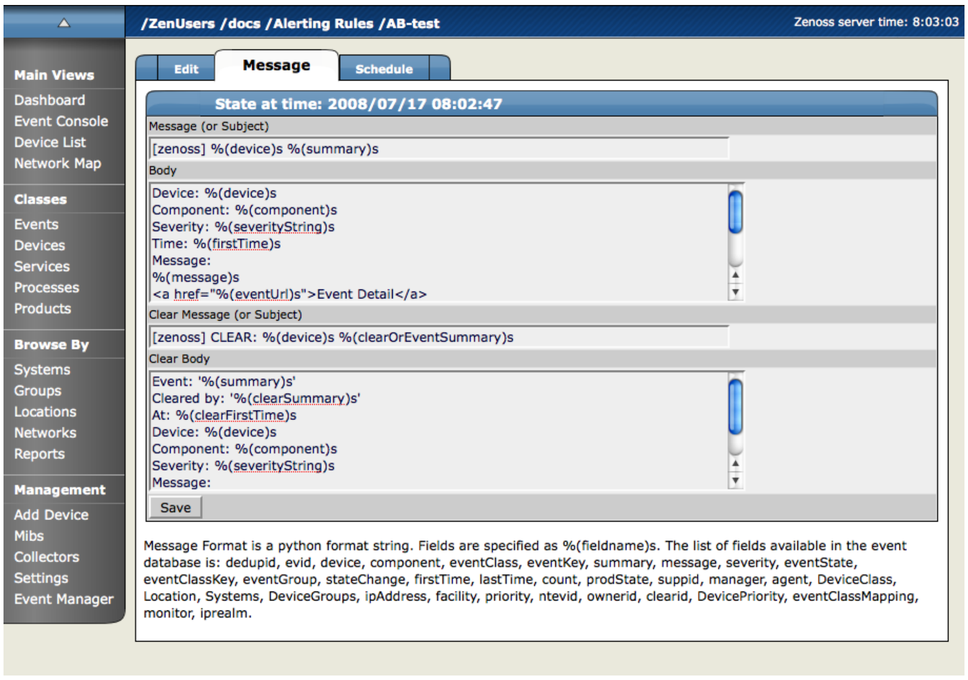
7. Click Save to save the values you entered on this tab.

2.2. 1.1.1 Create the Content of the Alert Message

1. From the Alerting Rules Page, click the Message tab to customize the message that is sent to the specified address.

   The Message tab appears.

Figure 20.7. Alerting Rules Message Tab



State at time: 2008/07/17 08:02:47

Message (or Subject)

```
[zenoss] %(device)s %(summary)s
```

Body

```
Device: %(device)s
Component: %(component)s
Severity: %(severityString)s
Time: %(firstTime)s
Message:
%(message)s
<a href="%(eventUrl)s">Event Detail</a>
```

Clear Message (or Subject)

```
[zenoss] CLEAR: %(device)s %(clearOrEventSummary)s
```

Clear Body

```
Event: '%(summary)s'
Cleared by: '%(clearSummary)s'
At: %(clearFirstTime)s
Device: %(device)s
Component: %(component)s
Severity: %(severityString)s
Message:
```

Message Format is a python format string. Fields are specified as %(fieldname)s. The list of fields available in the event database is: dedupid, evid, device, component, eventClass, eventKey, summary, message, severity, eventState, eventClassKey, eventGroup, stateChange, firstTime, lastTime, count, prodState, suppid, manager, agent, DeviceClass, Location, Systems, DeviceGroups, ipAddress, facility, priority, ntevid, ownerid, clearid, DevicePriority, eventClassMapping, monitor, iprealm.

2. Use the Message tab to specify the email message subject and body. You actually have two (2) message to create. The first (called Message) is the message to send when the thresholds for the alert are met or exceeded. The second message is the one to send when the event has cleared (called Clear Message).

   The fields for the subject and message areas are python format strings.

   At the bottom of the page there is a list of the fields available for an alert. A clear events fields are accessed by prefixing the field name with “clear”. For instance the field prodState becomes clearProdState. There is also a special field clearOrEventSummary which will print the clear summary or if it does not exist the original alert summary. This is useful for the subject of a clear alert. In the case where an alert has no clear (it was deleted for the UI for instance) a meaning full subject will still be created.

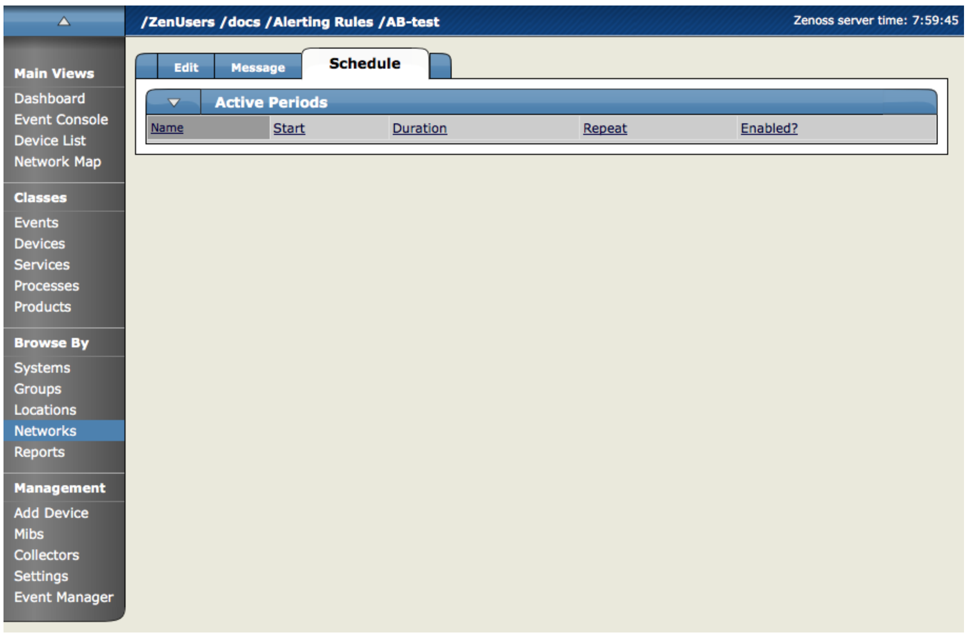
3. Click Save to save the data you entered on this page.

2.3. Create a Schedule for Sending the Alert

1. From the Alerting Rules page, click the Schedule tab to set up a schedule for the Alert.

   The Schedule tab appears.

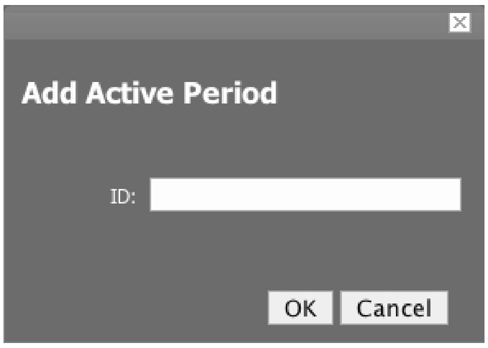
   Figure 20.8. Alerting Rules - Schedule Tab



2. To add a new Schedule for the Alert, from the Active Periods table menu, select Add Rule Window.

   The Add Active Period dialog appears.

Figure 20.9. Add Active Period Dialog



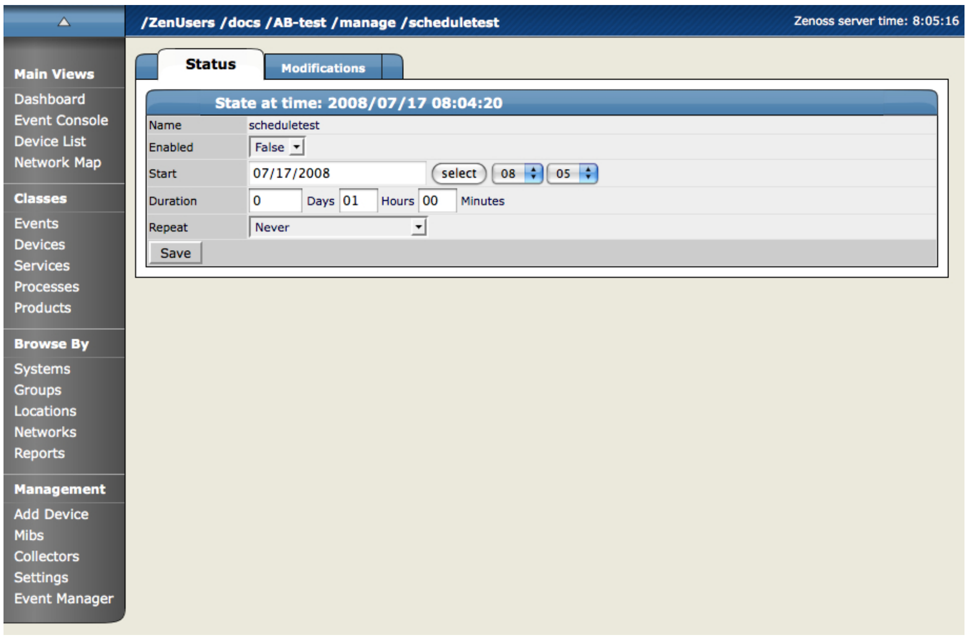
3. Enter a name for the schedule in the ID field and click the OK button.

   The Schedule you added appears in the Active Periods list.

4. Click the name of the new Schedule to set the details for the schedule.

   The Schedule Details page appears.

Figure 20.10. Alerting Rules - Schedule Details Page



State at time: 2008/07/17 08:04:20

Name	scheduletest
Enabled	False
Start	07/17/2008 select 08 05
Duration	0 Days 01 Hours 00 Minutes
Repeat	Never

5. If you want to restrict this Alert to only monitor at certain times for certain durations, set the Enabled field to True.

6. In the Start area, enter the date you want the alert to start, or click the Select button to choose the date from a calendar.

7. In the fields to the right of the date, select an hour and minute for the Alert to start.

8. Use the Duration area to specify the length of time you want to Alert to be listening based on the start time.

9. If you want the Alerting period to repeat you can choose a time frame from the Repeat pop-up menu. You can choose from:

   - Never
   - Daily
   - Every Weekday
   - Weekly
   - Monthly
   - First Sunday of the Month

10. Choose a number of times to repeat the selected interval.

11. Click Save.

    You have now saved all of the options for creating a new alert.

3. Escalation of Messaging in Zenoss

You can create a messaging hierarchy using some of the different managing tools available in Zenoss.

3.1. Creating an Alerting Hierarchy

You can create a alert Hierarchy based on event status and delays using Alerting Rules. For example, you send an alert to Person A (for example the first level, on call tech) as soon as any event of a given priority occurs, if that event is not Acknowledged or Suppressed (changing the status of the event) within a given time frame (For example, an hour) you create an additional alerting rule to send email to the next person in the Alert Hierarchy, for example Person B.

To create this hierarchy, first, you create an Alerting Rule for the Default case (the initial state). This case is "Any when any new event of whatever priority occurs, alert Person A. To create this default rule, just create the Alerting rule as usual. Set the Delay at 0 (Zero) seconds. Now for the next level in the alerting hierarchy (Person B) you want to say "OK, if Person A does not acknowledge or suppress the event within an hour - then send an Alert to the next person in the hierarchy (Person B). To create this hierarchy in Zenoss, you have to create an additional alerting rule for Person B. There are 2 ways of doing this - you can A) Create an additional rule for the User account you are currently logged in as or B) You can just add the additional (Person B's email address in the Address (optional field). This added address overrides the User account email so email will only be sent to the newly added email address.

For the 2nd Alerting Rule, set the delay to the number of seconds you want to wait after an event has come in and has not been acknowledged or cleared. For our example, we have determined that time-frame to be one hour. One hour is 3600 seconds. So in the Delay box for the Alerting Rule, enter 3600.

Now we need to pass a condition that will keep this alert from firing on all events, even the ones that Person A acknowledges or Suppresses before the hour expires. In the Add filter area, select Event State and then select the event state that will keep this rule from being executed on ALL events, even ones acknowledged by Person A. In this case, an Event is considered to have a state of New unless it has been either Suppressed or Acknowledged so let’s leave the setting at New. So now this alerting rule will send email to Person 2's email if an event remains in the New State for more than 3600 seconds (one hour)

So this section defines a small 2 person hierarchy, but you can use the filters for alerting rules to create much more sophisticated alerting hierarchies and scenarios. Device priority is a new attribute that will work great for defining which person gets which alerts unless a device has a certain priority level in which case everyone gets the alert.

4. Adding Delay and Schedules to Alerting Rules

You can use Delays when creating alerting rules to set up on-call schedules and elevation hierarchies. Using delay will allow you to specify that if an event is not acknowledged in a certain amount of time, then send email to the next person in the hierarchy. You accomplish this by filtering on event state ('New' not acknowledged) and adding a delay. Create an alerting rule for the tier 1 support person that does not have a delay so they find out immediately and can acknowledge the event if possible.

1. Create a second alerting rule (this one will be for the tier 2 person in the hierarchy) and enable it.

2. Use the 'where' clause to say this rule is only in effect for events that have not been acknowledged.

   Delay = 300 (in seconds, 5 minutes)

   In the Where area,

   Production State = Production

   Severity >= Error

   Event State = New

   This rule now says fire this alert if there is an event in the system that is New (not acknowledged) for 5 minutes send email to this user.

3. Click the Message tab and in the Message (or subject) field enter the following:

   [Zenoss-delayed] %(device)s %(summary)s

4. In the Clear message (or Subject) area, enter the following.

   [Zenoss-delayed] CLEAR: %(device)s %(clearOrEventSummary)s

5. Click the Schedule tab to edit the schedule. You can tell the rule to only be active when this user is on call (remember each alerting rule is user based).

6. In the Add field enter a name for the new schedule.

7. The new schedule appears in the list. Click the name of the new schedule.

   Name = name of the new schedule

   Enabled = True

   Start = whenever you want the rule to start

   Duration = how long you want this rule to be in effect

   Repeat = number of times to repeat the schedule

   Every = how many of the time periods to repeat for

8. Click Save.

Chapter 21. UI Commands

1. About UI Commands

User commands allow users to execute arbitrary shell commands from within Zenoss. These commands can then be run manually against a single device or a group of devices. The user commands are executed on the Zenoss server, not on the remote device (unless the user command explicitly uses ssh to connect to the remote device.) User commands can be defined on any device class, system, group or location. They can also be defined globally from the Settings page under the Manage tab.

2. Defining UI Commands

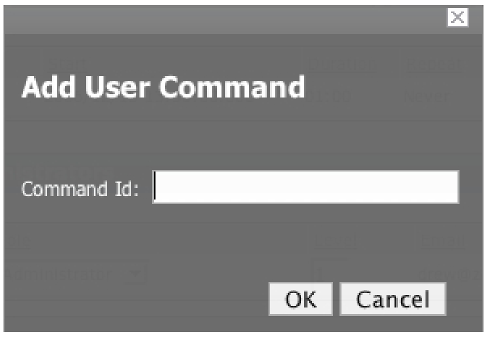
1. To access the Define Commands area, from any device you have loaded into your system, open the Device page menu, select More >> Administration.

   The Administration tab appears.

2. From Define Commands table menu, select Add User Command.

   The Add User Command dialog appears.

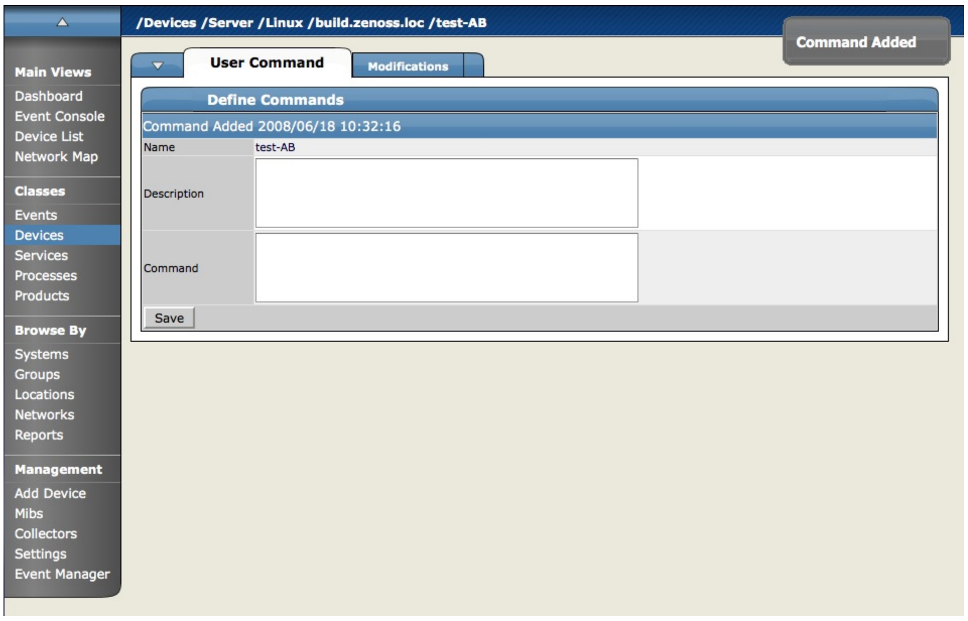
   Figure 21.1. Add User Command Dialog

3. In the Command Id field, enter a name for the command and click OK.

   The Define Command page appears.

Figure 21.2. Define User Command Dialog



Command Added 2008/06/18 10:32:16

Name	test-AB
Description	
Command	

4. In the Description field, brief description of what the command will do.

5. In the Command section, enter the TALES expression based command you want to be run on the selected device or devices.

6. Click the Save button.

   The Command is saved and added to the command drop-down menu so you can choose to run it at any time.

2.1. UI Command Example: Echo Command

This section will walk you through creating an echo UI command. You can see the use of TALES expressions in the definition of this command as in many other commands in Zenoss.

1. Go to the Settings - Manage tab.

2. Add a new command called “echoDevice”

3. Echo the name and IP of the device

   ```
   echo name = ${here/id} ip = ${here/manageIp}
   ```

   In a TALES expression ‘here’ is the object that the expression is executed against. Some TALES expressions in Zenoss have other variables like evt for event and dev or device for the device. See the TALES Expression appendix for more information on the syntax of the various TALES expressions.

4. Go to a device and run this command.

5. Now go back to the editing of this command and add some more information to the command.

   ```
   echo name = ${here/id} ip = ${here/manageIp} hw = ${here/getHWProductName}
   ```

6. Now try running the command against a group of devices and see the command outputs.

3. Running Commands from the Zenoss Web UI

Zenoss allows commands to be run though the user interface. It comes with several built in commands such as ping and traceroute. Commands can be run on a single device or on a group of devices.

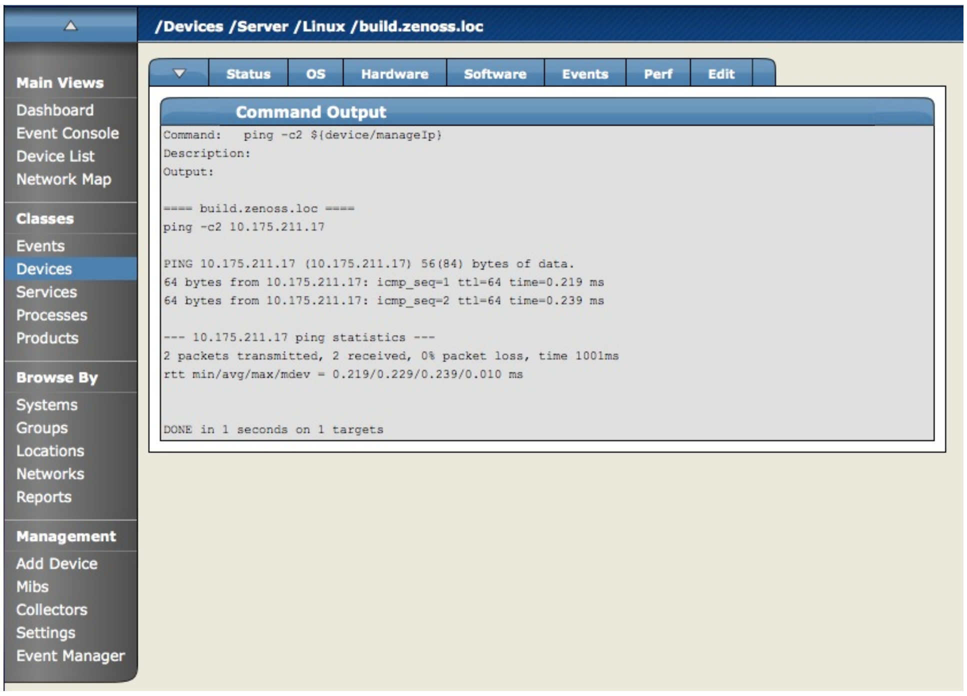
You can run a command one time on a device or group of device from the Zenoss System.

1. Navigate to the Device or Device Group where you want to run the command (and where you have the command defined).

2. 1.From the Device page menu, select Run Commands and then the command name you want to run.

   The command is run and the output appears on the screen.

Figure 21.3. Command Output



```
Command:   ping -c2 ${device/manageIp}
Description:
Output:

==== build.zenoss.loc ====
ping -c2 10.175.211.17

PING 10.175.211.17 (10.175.211.17) 56(84) bytes of data.
64 bytes from 10.175.211.17: icmp_seq=1 ttl=64 time=0.219 ms
64 bytes from 10.175.211.17: icmp_seq=2 ttl=64 time=0.239 ms

--- 10.175.211.17 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.219/0.229/0.239/0.010 ms


DONE in 1 seconds on 1 targets
```

4. Auto-Running Commands Based on Events

Zenoss allows you to set up commands to automatically be run against a device based on an event entering into the system.

To run events based on Events:

1. 1.From the Zenoss Dashboard, from the left navigation menu, in the Management area, select Event Manager.

   The Event Manager Edit tab appears.

2. Click the Commands tab.

   The Commands tab appears.

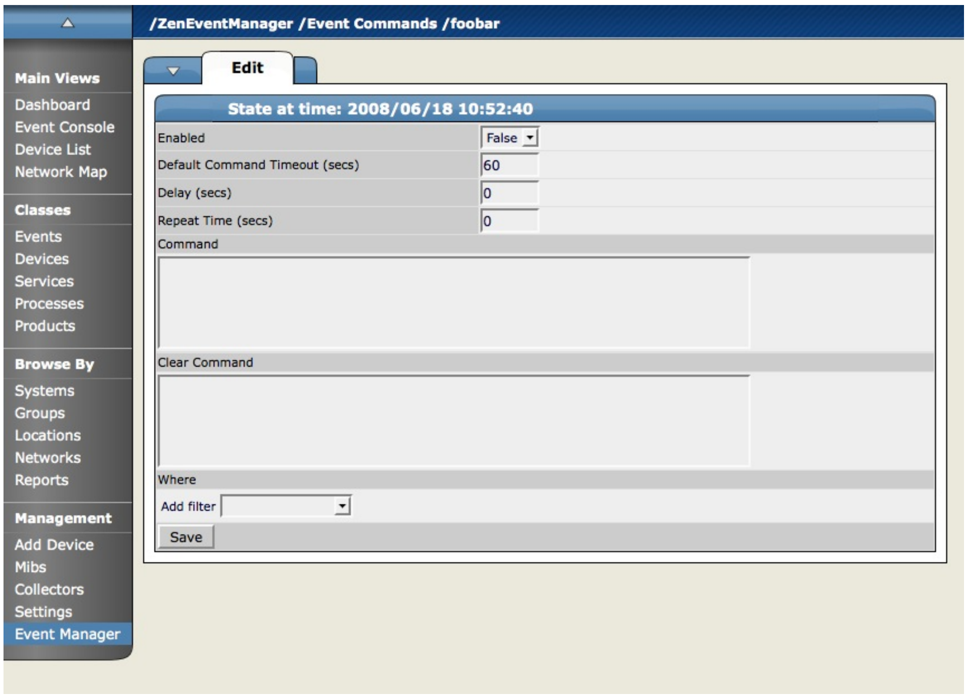
3. Enter a name for the new Command, and click the Add button.

The new command appears in the command list.

4. Click on the name of the new command in the command list to give the command its attributes.

   The Edit Command window appears.

   Figure 21.4. Edit Event Command Page



State at time: 2008/06/18 10:52:40

Enabled	False
Default Command Timeout (secs)	60
Delay (secs)	0
Repeat Time (secs)	0
Command	
Clear Command	
Where	
Add filter	

5. Enable the command by setting the Enabled box to True.

6. Now set the other attributes for the command change the default timeout if necessary. Default timeout for the command is 60 seconds.

7. You can also change the delay for the command so there is a gap between when the conditions for the command to be run are met and when the command actually runs.

8. In the Command area, enter the command you want to run when the conditions are met. These commands use TALES expressions. (See the section on TALES Expressions for more detail.)

9. In the clear Command area, enter a command to be run once the event has cleared.

   Commands are built using TALES Expressions. Documented in the Appendix. When the conditions in the Where clause are met, the command you create runs. The power of this command and where and what you can run is only limited by the power of the scripts you create and the permissions on the machine where you want to execute the commands.

   This is optional, but could be useful if, for example, you were running an SMS modem separate from your alerting rules and wanted to send on message in the Command area to send an SMS message if Ping went down, and then when the Ping Down event was clear, you wanted to send an additional message from the SMS modem that it is up.

10. At this point you use the “Where” area to set the conditions for this Command to be run.

11. In the Add Filter drop down select the filters for your Command.

12. You can filter on the Following Event cases:

    - Agent
    - Count
    - Device Class
    - Device Priority
    - Event Class Key
    - Facility
    - Manage
    - ntevid
    - Priority
    - Component
    - Device
    - Device Groups
    - Event Class
    - Event State
    - IP Address
    - Message
    - OwnerId

    You can add as many conditions as you want to further narrow the cases in which the Command will be executed.

13. Once you have completed setting up the conditions, click the Save button. Now the command is entered and will run when the conditions are met.

Chapter 22. Production States and Maintenance Windows

1. About Production States and Maintenance Windows

Zenoss has the capability to support “Maintenance Windows” or time periods (either scheduled or on the fly) where the monitoring and alerting rules are changed. A set of rules governing monitoring, display and alerting can be collectively defined as “Production States”. When there are temporary changes in Production State, and then a reversion to the original state, this is a Maintenance Window.

2. Production States

Production State is an important concept in Zenoss. It determines whether or not a device is monitored and can be used to control several elements of the event system such as whether or not an event will produce a remote alert (email or page). Typically devices start off their life in state “Pre-Production.” In this state, devices are monitored by default but no remote alerting occurs and events aren’t shown on the Dashboard. Once a device is in full “Production” state monitoring is occurring and remote alerts are sent. If service needs to be performed on a device its state can be set to “Maintenance” to temporarily block any remote alerts.

There are three factors that affect and define the Production State for devices:

1. Whether or not the device is being monitored.

2. Whether or not you want alerting to occur.

3. Whether or not the device appears on the dashboard.

The available Production States are merely combinations of the above:

- Production - you want all three: monitoring, alerting and dashboard.
- Pre-production - you may want monitoring but not alerting or the appearance of the device notices on the dashboard
- Test - you may want monitoring and alerting (sent to one email) and but not displaying device info on the dashboard.
- Maintenance - you want monitoring and collection to occur, and maybe or maybe not the device on the dashboard, just not alerting occurring
- Decommissioned - no monitoring, no dashboard, no alerting

2.1. Defining Production States for Devices

You can set the Production State for a device or group of device(s) by going to the Edit Tab for the device or device group and changing the Production State drop-down to whatever state you want the new Production State to be. The default Production state when you add a device is Production. If you change the Production State for a hierarchy of devices, the Production state propagates down the hierarchy except when you define an exception to the Production State further down the hierarchy.

3. Maintenance Windows

Zenoss Maintenance Windows allow scheduled production state changes of a device or all devices in a System, Group, or Location. Maintenance Windows are defined on the Manage tab of these objects.

A Maintenance Window has a Start Time, Duration, Repeat Cycle and Start and End Production State. A typical Maintenance Window changes the Production State to Maintenance at its start, and to Production at its end. The Start Production State for the Maintenance Window is the state that the monitoring for the device (or group of devices) is in when the Maintenance Window begins or “opens”. For example, if your device(s) are running along in a Production State of “Production” (meaning you are monitoring and alerting on the devices normally) and the Maintenance Window opening time arrives, the Production State changes to the maintenance Window’s Start Production State.

For example, if the Start Production State is set to Maintenance, this means you want monitoring and data collection to continue to occur for the device, but you don’t want alerts to occur or any warnings to appear on the dashboard. You can use this time to reboot the machine or make configuration changes that would normally create alerts and not have them actually send alerts. You can either schedule a Maintenance Window or change the Production State for the device manually at the time you want to make the changes. When the Maintenance window closes, the device(s) change to the End Production State for the Maintenance Window. You define the End Production State for the Maintenance Window. This refers to the Production State that you want the device(s) to revert to when the Maintenance Window ends. So the Start Production State is the state you want when the Maintenance Window opens - if I were setting up a Maintenance Window, I would define the window such that when its time for the Maintenance Window to occur, I want the Start Production State to be Maintenance, and then when the Maintenance Window time frame expires, I want the Stop Production State to be Production, meaning its back to monitoring and alerting as normal. This would save sending out known alerts as you rebooted or created other, known, alerting events.

3.1. Creating and Using Maintenance Windows

You can create a Maintenance Window for an individual device or for any grouping of devices in any hierarchy you create and define. You can create these windows either for a single device or create a window per device class or system, and have the window propagate to all devices that fall below where you create the window.

To create a new maintenance window:

1. Navigate to the device or group of devices where you want to define the maintenance window. Open the page menu and select More and then Administration.

   The Administration page appears.

2. From the Maintenance table menu, select Add Maint Window.

   The Add Maintenance Window appears.

3. In the ID field enter a name for the maintenance window and click OK.

   The name appears in the Maintenance Window list.

4. To define the window, click the name of the Maintenance Window.

   The Maintenance Window Status Tab appears.

5. Define the attributes for this Maintenance window.

   - Name - Name of the maintenance window.
   - Enabled - True or False as to whether you want this maintenance window active.
   - Start - The time for the window to become active.
   - Duration - The length of time for the window to be in effect starting from the Start time.
   - Start Production State - Defines the production state for the window before it opens.
   - Stop Production State - Defines the production state for the object once the maintenance window closes.

6. Click Save.

Chapter 23. Reporting

1. About Reporting in Zenoss

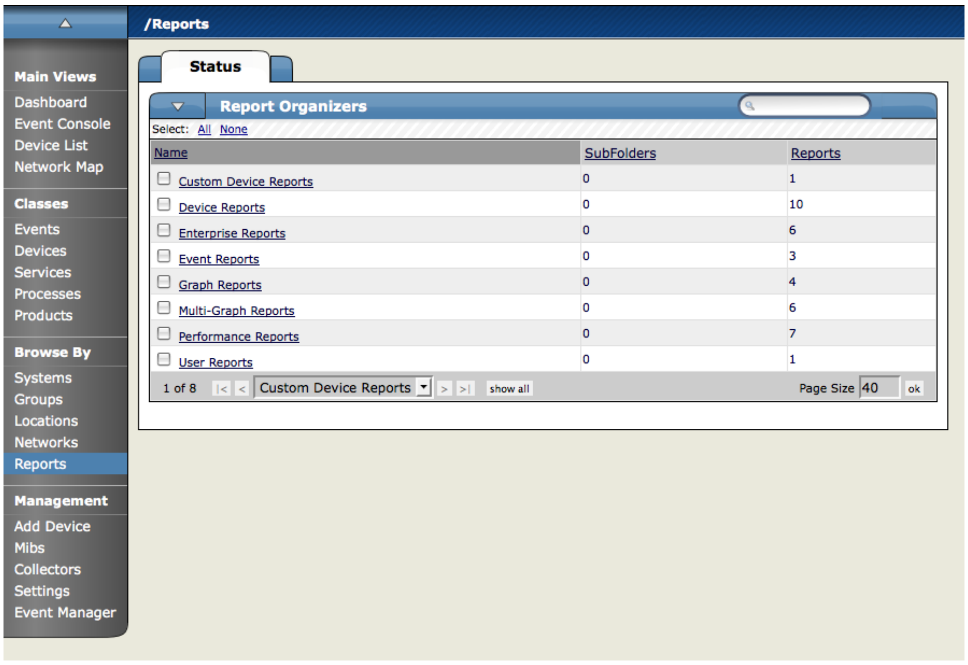
Zenoss also excels at aggregating and reporting on data over time for any of the data you have set it up to monitor.

The kinds of Reports that are available include:

- Summary Reports for performance information (CPU, memory, i/o pages, load average, network interface utilization)
- Performance threshold summary reports
- Go through the history events and calculate the percentage uptime for device and is components.
- Availability report of any event class
- Ping Issues
- SNMP Issues
- SLA Reporting
- Graphing of trend over time: hourly, weekly, monthly, yearly
- Other Performance Reports

  To see the Reports area of Zenoss, from the left navigation click Reports. The Reports organizer list appears.

Figure 23.1. Report Organizer List



Name	SubFolders	Reports
Custom Device Reports	0	1
Device Reports	0	10
Enterprise Reports	0	6
Event Reports	0	3
Graph Reports	0	4
Multi-Graph Reports	0	6
Performance Reports	0	7
User Reports	0	1

1 of 8 | Custom Device Reports | show all | Page Size 40 | ok

2. Organizing Reports

Reports can be organized just like any other elements in the Zenoss system. You can create categories, sub-categories and Organizers to organize the reports as you see fit. You can create any kind of hierarchy and relationships the same way you can do for devices or events.

3. Navigating and Sorting Report Results

Each report result can be sorted by column heading by clicking on the column heading you want to sort by. You can also sort this report based on any of the column headings or filter based on any keyword you enter in the filter field.

4. Exporting Reports

You can export any of the included reports in Zenoss by going to the Exporting of Reports as comma separated value (.csv) files. To export the reports, from the bottom of any of the reports, click the ”Export All” button.

4.1. Add An Export Button to a Report

Adding an Export All button to a report is fairly straightforward. The overall format of the report markup looks something like this:

```
<tal:block tal:define="
objects python:here.ZenUsers.getAllThingsForReport();
objects python: (hasattr(request, 'doExport') and list(objects)) or objects;
tableName string: thisIsTheTableName;
batch python:here.ZenTableManager.getBatch(tableName,objects, sortedHeader='getUserid'
);
exportFields python:['getUserid', 'id', 'delay',
'enabled', 'nextActiveNice', 'nextDurationNice',
'repeatNice', 'where'];
">
<tal:block metal:use-macro="here/reportMacros/macros/exportableReport"> <tal:block metal:fill-slot="repor
```

The normal report markup goes here:

</tal:block>

</tal:block>

</tal:block

The first definition is a call to some method that retrieves the objects for the report. This might be a list, tuple or an iterable class.

If we are doing an export then we need this to be a list, so the second tal:define line makes sure we have a list in the event that we are doing an export. It's good to not do this if we are not doing an export. Large reports might run into performance issues if an iterable is converted to a list unnecessarily.

tablename is defined here for use by the getBatch() call that follows.

exportFields is a list of data to be included in the export. These can be attribute names or names of methods to call. See DataRoot?.writeExportRows() for more details on what can be included in this list.

Within the <tal:block metal:fill-slot="report"></tal:block> block goes the report markup you would use when not including the export functionality.

Note: If the Export All button is mysteriously not doing anything you may need to be using zenTableNavigation/macros/navtool rather than zenTableNavigation/macros/navbody in your report. The former includes the <form> tag, the latter does not. If you are not providing a <form> tag then you need to use navtool so the export button is within a form.

5. Reports Included With Zenoss

The basic Report categories (the ones included when you build Zenoss)

5.1. Device Reports

Device Reports aggregate and display data over many devices and device attributes.

- All Monitored Components – The All Monitored components shows all of the components currently being monitored. This is NOT all components in the system, only the ones that Zenoss is currently collecting performance data on. The information that appears in this report is: the device name, Component, the type of component (when available), the description, and the status of each device.

- Model Collection Age – The Model Collection age report shows information about the history of the modeling that Zenoss does of each device. The information available in this report includes the device name, Device Class, the time when the device was first seen by Zenoss, the last time Zenoss Collected Data on this device, and any changes made to the configuration at that time.

- Device Changes - The Device Change report shows information about the history of any changes that Zenoss detects when modeling each device. This report only shows devices with changes. The information available in this report includes the device name, Device Class, the time when the device was first seen by Zenoss, the last time Zenoss Collected Data on this device, and any changes made to the configuration at that time.

- Ping Status issues – The Ping Status Report shows the device name, the device class, the Product name, the current state of the device and the Ping and SNMP status. The only devices that will show up here are devices that actually have or have had Ping issues.

- SNMP Status Issues - The SNMP Status Report shows the device name, the device class, the Product name, the current state of the device and the Ping and SNMP status. The only devices that will show up here are devices that actually have or have had SNMP issues.

- New Devices – The New Devices report shows devices that have been recently added. The report shows the Device Name, the Device Class, when the device was first seen and the model collection age.

- New Devices – The New Devices report shows devices that have been recently added. The report shows the Device Name, the Device Class, when the device was first seen and the model collection age.

5.2. Event Reports

Event reporting gives you aggregate data about events, event mappings and event classes.

- All Heartbeats – The All Heartbeats Report shows all heartbeats for monitored devices. Heartbeats are reported by component and number of seconds.

- All Event Classes – The All Event Classes report shows all of the event classes that reside in the Zenoss system. It also breaks these classes down by SubClasses, the number of instances of that class in the system and the number of events in the system associated with each Event Class.

- All Event Mappings – The All Event Mappings shows all of the event mappings that you have created throughout the system. You can sort them by Event Class key, Evaluation or number of events per event class.

5.3. Performance Reports

Performance Reporting allows you to roll up performance data across the Zenoss system.

- Aggregate Reports – The Aggregate reports shows the performance graphs for all of the devices in the system in graphical format. Common performance stats include PU Usage, Aggregate Free Memory, Aggregate Free Swap, and Network Output/Input. You can edit the graph parameters by clicking on the graph. You can change the Width, height, Min Y and Max Y axis. You can also specify which devices are included in the aggregate and the time span for the graph.

- File System Utilization Report – The File System Utilization Report shows the Total Bytes, used Bytes, Free Bytes and % Utilization for each device. You can customize the report through the interface for such attributes and start/end Date and Summary type; either average or Maximum.

- CPU Utilization Report – The CPU Utilization report shows all of the Monitored Interfaces, the list of devices and the load average and % Utility. You can customize the report through the interface for such attributes and start/end Date and Summary type; either average or Maximum.

- Threshold Summary – The Threshold Summary Report identifies the devices that approach or exceed their thresholds and reports them in a list. You can see the device, the component, the event class, count, duration and percentage. You can filter this list by Event Class or to see all Event Classes, leave the event class Selection List blank. You can also change the start and end date for the reporting data.

- Availability Report – The Availability Report reports availability of devices in percentage form. You can filter on device, component, event class or severity. You can also further limit the time frame for the availability.

  The availability report provides an **estimate** of the time a device or component is available for monitoring things such as Service-level Agreements (as an example).

This report summarizes the amount of time a device or component can be considered "unavailable". The definition of unavailable differs depending on what type of service you require. If a web server goes down, the web service is not available, even though the device may actually be reachable (pingable) on the network.

As Zenoss collects events against devices and components, these events are classified into categories that are useful for determining if a particular service is unavailable. For example, the class "/Status/IpService" can provide all the events related to a failed IP Service. The availability report adds up all of the time (lastTime - firstTime) for all of the events based on device, component and/or eventClass. This defines the "unavailable" period. The availability is the remainder of the period.

This is an optimistic estimate of availability because events that occur only once and events that have the same first and last time, are not counted. Another assumption the estimate makes is that devices without any matching events during the Availability time range are assumed to be 100% available.

- Memory Utilization – The Memory Utilization Report provides system wide information about the memory usage for devices in the system. This report shows Total memory, Available memory, Cache memory, Buffered memory, and percent of memory utilized.

5.4. User Reports

User Reports refer to report based on user account information and changes within the Zenoss system.

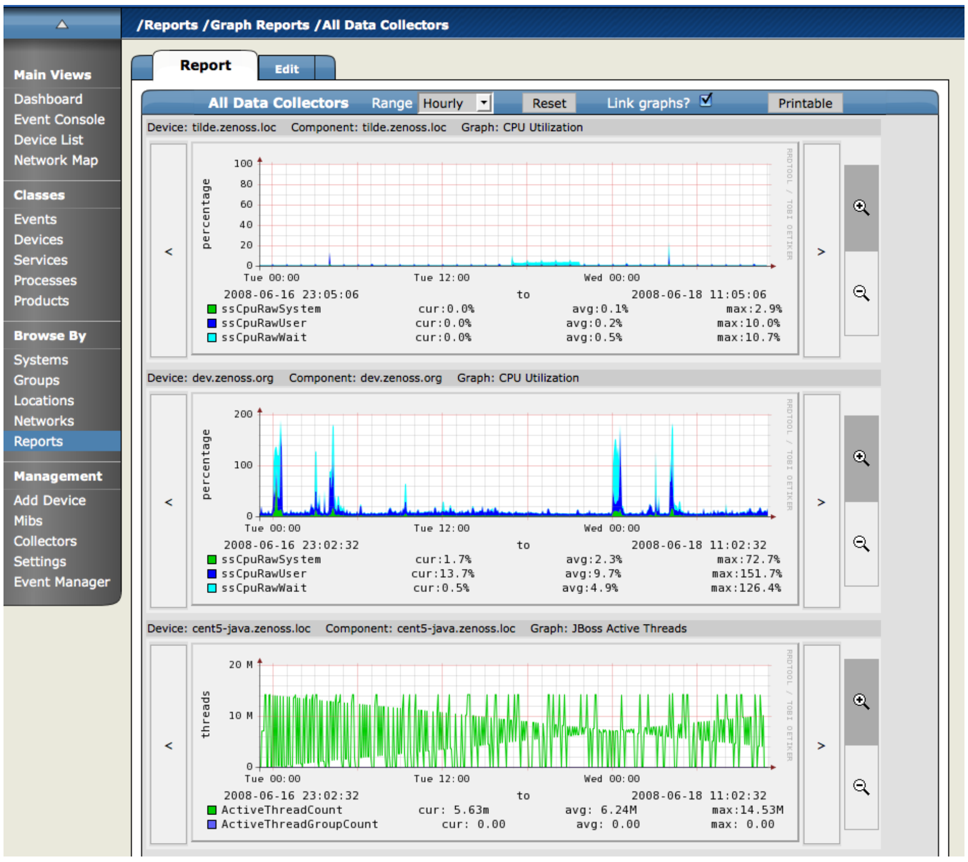
- Notification Schedules – The Notification Schedules Report shows all of the alerting rules and their associated details with each one.

6. Graph Reports

Graph Reports allow you to assemble graphs from specific Devices and Device Components into a single report. Click on Reports in the left navigation menu, then on the Graph Reports organizer to view or create Graph Reports. Graph Reports have a normal view which is similar to the graph views for Devices and Device Classes. Graph Reports also have a Printable view more appropriate for printing which can be brought up by clicking on the Printable button at the top of the report view.

Note that Graph Reports can only display graphs that already exist on Device or Components within Zenoss. You cannot define new graphs or alter existing graphs from within a Graph Report. If you need this type of functionality you probably want MultiGraph Reports instead.

Figure 23.2. Graph Report


/Reports /Graph Reports /All Data Collectors
Main Views
Dashboard
Event Console
Device List
Network Map
Classes
Events
Devices
Services
Processes
Products
Browse By
Systems
Groups
Locations
Networks
Reports
Management
Add Device
Mibs
Collectors
Settings
Event Manager
Report
Edit
All Data Collectors
Range
Hourly
Reset
Link graphs?
Printable
Device: tilde.zenoss.loc Component: tilde.zenoss.loc Graph: CPU Utilization
percentage
2008-06-16 23:05:06 to 2008-06-18 11:05:06
ssCpuRawSystem cur:0.0% avg:0.1% max:2.9%
ssCpuRawUser cur:0.0% avg:0.2% max:10.0%
ssCpuRawWait cur:0.0% avg:0.5% max:10.7%
Device: dev.zenoss.org Component: dev.zenoss.org Graph: CPU Utilization
percentage
2008-06-16 23:02:32 to 2008-06-18 11:02:32
ssCpuRawSystem cur:1.7% avg:2.3% max:72.7%
ssCpuRawUser cur:13.7% avg:9.7% max:151.7%
ssCpuRawWait cur:0.5% avg:4.9% max:126.4%
Device: cent5-java.zenoss.loc Component: cent5-java.zenoss.loc Graph: JBoss Active Threads
threads
2008-06-16 23:02:32 to 2008-06-18 11:02:32
ActiveThreadCount cur: 5.63m avg: 6.24M max:14.53M
ActiveThreadGroupCount cur: 0.00 avg: 0.00 max: 0.00
Tue 00:00
Tue 12:00
Wed 00:00

6.1. Creating a New Graph Report

From within the Graph Reports organizer or a sub-organizer use the Add Graph Report menu item to create a new report. After entering a name for the new report you will be taken to the edit page. The fields for a Graph Report are:

- Name - The name of the report is displayed at the top of the report.
- Title - This description is displayed in the list of reports for the report organizer. It is also available for use in the comments.
- Number of Columns - This is the number of columns the graphs will be displayed in on the report.
- Comments - The comments are displayed at the top of the Printable version of the report. This is a TALES evaluated string that may contain html formatting. The variables available to the TALES expression are now (the current datetime) and report (the report object itself.)

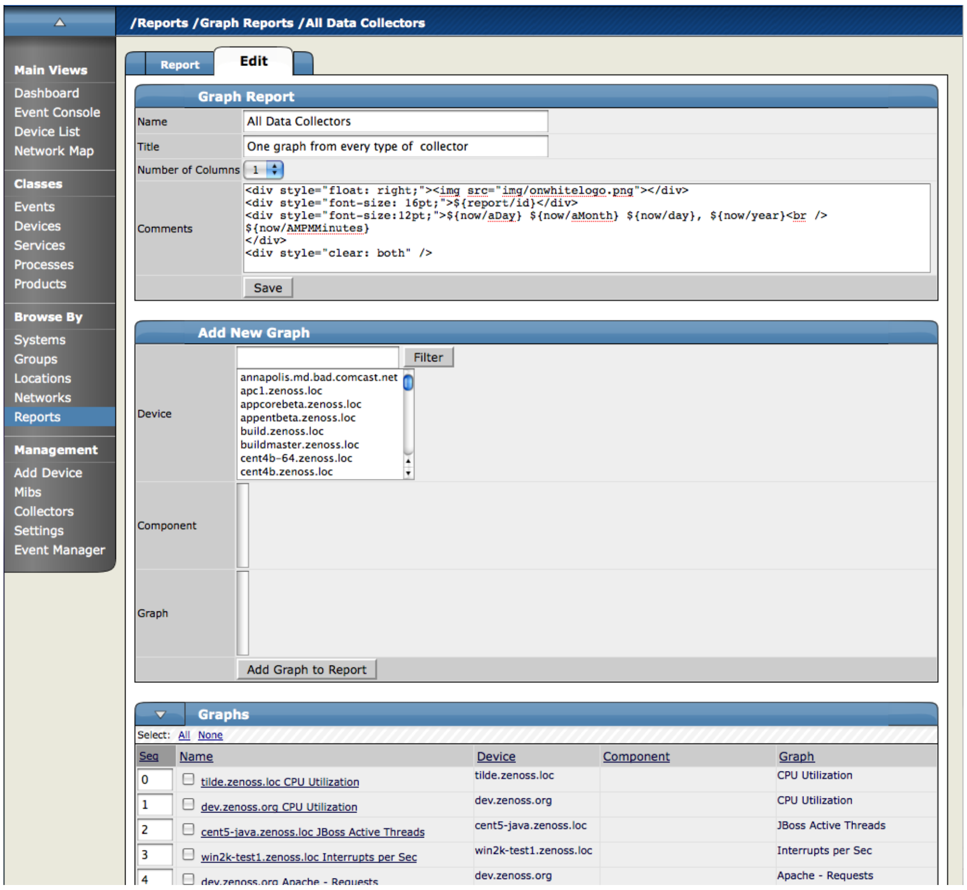
6.2. Adding Graphs

The Add New Graph section of the Edit page allows you to add one or more graphs to the report. The first step is to select one or more Devices. The list of Devices can be narrowed by entering a search string to the left of the Filter button and pressing Return or clicking the Filter Button. When you select one or more Devices the Component list will display the names of all Components defined on at least one of the selected Devices. The Graph list will list the Graphs available on one or more of the listed Devices. When one or more Components are selected the Graph list will display Graphs from the selected Components rather than the selected Devices.

At any point you may select one or more Graphs and use the Add Graph to Report button. Zenoss steps through each selected Component (if any are selected) or Device (if no Components are selected) looking for graphs with the given names. Matching graphs are added to the Graph Report.

Graph Reports maintain a static list of graphs which does not change when graphs are added or deleted from Performance Templates. For example, take DeviceA which has only one graph called Graph1 and DeviceB which has two graphs named Graph1 and Graph2. On the Graph Report edit page if you selected DeviceA and DeviceB the list of graphs would include Graph1 and Graph2. Selecting both graphs and clicking the Add Graph to Report button would add three graphs to the report: DeviceA's Graph1, DeviceB's Graph1 and DeviceB's Graph2. If at some later date you created a Graph2 on one of DeviceA's Performance Templates it would not automatically appear on the Graph Report, you would have to edit the report to specifically add it. Similarly, if one of the graphs was removed from DeviceB's Template (or if DeviceB was deleted from Zenoss) you would need to manually remove them from the Graph Report.

Figure 23.3. Graph Report Edit Page



/Reports /Graph Reports /All Data Collectors

Graph Report

Name	All Data Collectors
Title	One graph from every type of collector
Number of Columns	1

Comments:

```
<div style="float: right;"><img src="img/onwhitelogo.png"></div>
<div style="font-size: 16pt;">${report/id}</div>
<div style="font-size:12pt;">${now/aDay} ${now/aMonth} ${now/day}, ${now/year}<br />
${now/AMPMMinutes}
</div>
<div style="clear: both" />
```

Add New Graph

Device:
annapolis.md.bad.comcast.net
apc1.zenoss.loc
appcorebeta.zenoss.loc
appentbeta.zenoss.loc
build.zenoss.loc
buildmaster.zenoss.loc
cent4b-64.zenoss.loc
cent4b.zenoss.loc

Component

Graph

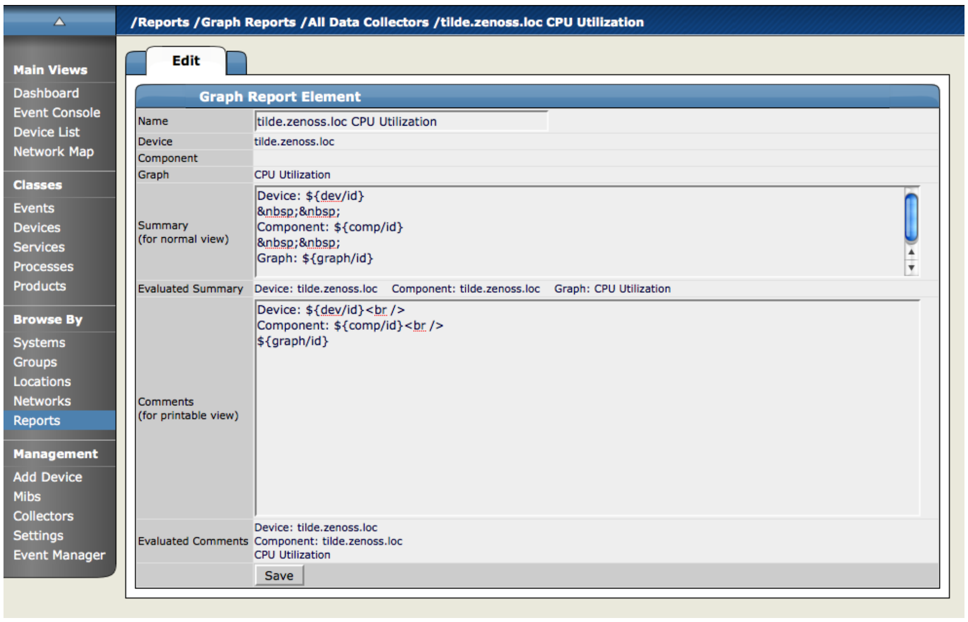
Graphs

Seq	Name	Device	Component	Graph
0	tilde.zenoss.loc CPU Utilization	tilde.zenoss.loc		CPU Utilization
1	dev.zenoss.org CPU Utilization	dev.zenoss.org		CPU Utilization
2	cent5-java.zenoss.loc JBoss Active Threads	cent5-java.zenoss.loc		JBoss Active Threads
3	win2k-test1.zenoss.loc Interrupts per Sec	win2k-test1.zenoss.loc		Interrupts per Sec
4	dev.zenoss.org Apache - Requests	dev.zenoss.org		Apache - Requests

6.3. Customizing Graph Text

The Graphs section of the Edit page lists the graphs that are included in this report. Click the name of any of the graphs takes you to an edit page where you can edit the text that appears with the graph when the report is viewed. The Summary field is displayed above the graph in the normal report view and the Comments field is displayed to the left of the graph in the printable view. Both of these fields may contain TALES expressions with these variables: dev (the device), comp (the component) and graph (the graph.)

Figure 23.4. Graph Report Element



/Reports /Graph Reports /All Data Collectors /tilde.zenoss.loc CPU Utilization

Graph Report Element

Name	tilde.zenoss.loc CPU Utilization
Device	tilde.zenoss.loc
Component	
Graph	CPU Utilization

Summary (for normal view)	Device: ${dev/id}
	
	Component: ${comp/id}
	
	Graph: ${graph/id}

Evaluated Summary	Device: tilde.zenoss.loc Component: tilde.zenoss.loc Graph: CPU Utilization
Comments (for printable view)	Device: ${dev/id}<br /> Component: ${comp/id}<br /> ${graph/id}
Evaluated Comments	Device: tilde.zenoss.loc Component: tilde.zenoss.loc CPU Utilization

6.4. Organizing Graphs

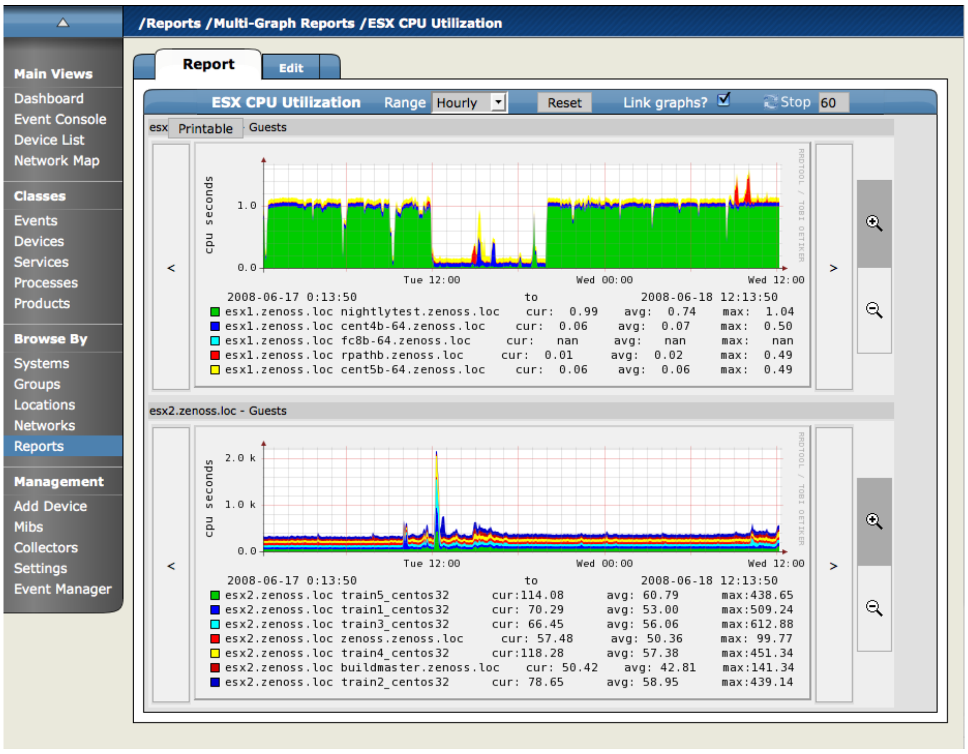
In both the normal and printed view of the report graphs are laid out from left to right in as many columns as specified on the report's edit page. For example, if a report has three columns then the first three graphs are placed on the first line of the report and the fourth graph would be the first one on the second row. By altering the sequence of the graphs on the edit page you can control where each graph appears on the report view. You change the sequence by editing the values in the Seq column beside each graph name and choosing the Re-sequence Graphs menu item.

7. MultiGraph Reports

MultiGraph Reports are a powerful mechanism for combining data from different Devices and Components into a single report. You can create a Graph Definition and have it drawn once for each in a group of Devices and Components that you define. Alternatively, you can have the data for those graphs combined into a single graph. The Graph Definitions are very similar to those used in Performance Templates except that these are defined and used exclusively within a single report. The groups of Devices and Components you assemble are called Collections. Specifying which Graph Definitions to apply to which Collections is done through Graph Group objects. Like Graph Reports, MultiGraph Reports have two different views, normal which looks similar to Device and Component graph pages and printable which is formatted more suitably for printing. The normal view is seen on the Report tab of any MultiGraph Report. At the top right of that view is a button named Printable for displaying the printable version.

MultiGraph Reports include their own Graph Definitions and thus do not use the Graph Definitions that are defined within Performance Templates. If you want to create a report that includes graphs defined on Templates then you may wish to use GraphReports rather than MultiGraph Reports.

Figure 23.5. MultiGraph Report Graphs



```
2008-06-17 0:13:50                         to                2008-06-18 12:13:50
🟩 esx1.zenoss.loc nightlytest.zenoss.loc    cur:  0.99    avg:  0.74    max:  1.04
🟦 esx1.zenoss.loc cent4b-64.zenoss.loc    cur:  0.06    avg:  0.07    max:  0.50
🟦 esx1.zenoss.loc fc8b-64.zenoss.loc    cur:   nan    avg:   nan    max:   nan
🟥 esx1.zenoss.loc rpathb.zenoss.loc    cur:  0.01    avg:  0.02    max:  0.49
🟨 esx1.zenoss.loc cent5b-64.zenoss.loc    cur:  0.06    avg:  0.06    max:  0.49
```

```
2008-06-17 0:13:50                         to                2008-06-18 12:13:50
🟩 esx2.zenoss.loc train5_centos32      cur:114.08      avg: 60.79      max:438.65
🟦 esx2.zenoss.loc train1_centos32      cur: 70.29      avg: 53.00      max:509.24
🟦 esx2.zenoss.loc train3_centos32      cur: 66.45      avg: 56.06      max:612.88
🟥 esx2.zenoss.loc zenoss.zenoss.loc      cur: 57.48      avg: 50.36      max: 99.77
🟨 esx2.zenoss.loc train4_centos32      cur:118.28      avg: 57.38      max:451.34
🟥 esx2.zenoss.loc buildmaster.zenoss.loc    cur: 50.42    avg: 42.81    max:141.34
🟦 esx2.zenoss.loc train2_centos32      cur: 78.65      avg: 58.95      max:439.14
```

7.1. Creating A New MultiGraph Report

From within the MultiGraph Reports organizer or a sub-organizer use the Add MultiGraph Report menu item to create a new report. After entering a name for the new report you will be taken to the edit page. The fields for a MultiGraph Report are:

- Name - The name of the report is displayed at the top of the report.

- Title - This is displayed at the top of the printable version of the report.

- Number of Columns - The report will display the graphs in this many columns on the report.

Once you've created the MultiGraph Report there are three steps required to get graphs showing on the report:

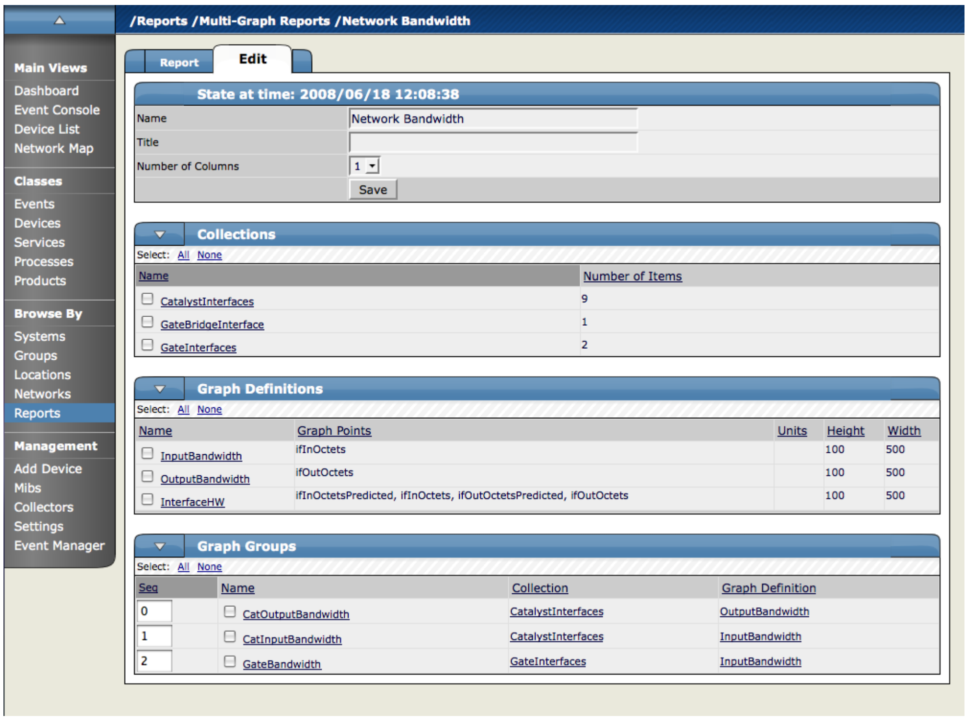
1. Create a Collection which contains the Devices and/or Components you want to graph.

2. Create a Graph Definition that describes the graph(s) you want on the report.

3. Create a Graph Group which specifies the Collection and the Graph Definition you just created. The Method setting in the Graph Group lets you choose to have the graph drawn once for each Device/Component in the Collection or you can have the data from all the Devices/Components combined into a single graph.

These are just the minimal steps to getting a functional MultiGraph Report. You can create any number of Collections, Graph Definitions and Graph Groups in a single report. See the sections that follow for details on creating Collections, Graph Definitions and Graph Groups.

Figure 23.6. MultiGraph Report Edit Page



/Reports /Multi-Graph Reports /Network Bandwidth

State at time: 2008/06/18 12:08:38

Name	Network Bandwidth
Title	
Number of Columns	1

Collections

Name	Number of Items
CatalystInterfaces	9
GateBridgeInterface	1
GateInterfaces	2

Graph Definitions

Name	Graph Points	Units	Height	Width
InputBandwidth	ifInOctets		100	500
OutputBandwidth	ifOutOctets		100	500
InterfaceHW	ifInOctetsPredicted, ifInOctets, ifOutOctetsPredicted, ifOutOctets		100	500

Graph Groups

Seq	Name	Collection	Graph Definition
0	CatOutputBandwidth	CatalystInterfaces	OutputBandwidth
1	CatInputBandwidth	CatalystInterfaces	InputBandwidth
2	GateBandwidth	GateInterfaces	InputBandwidth

7.2. Collections

A Collection is a group of Devices and/or Components. A MultiGraph Report must have at least one Collection. Collections are listed in the Collections table on the report's Edit page. You can create a new Collection with the Add Collection menu item on that table then specifying a name in the dialog that appears.

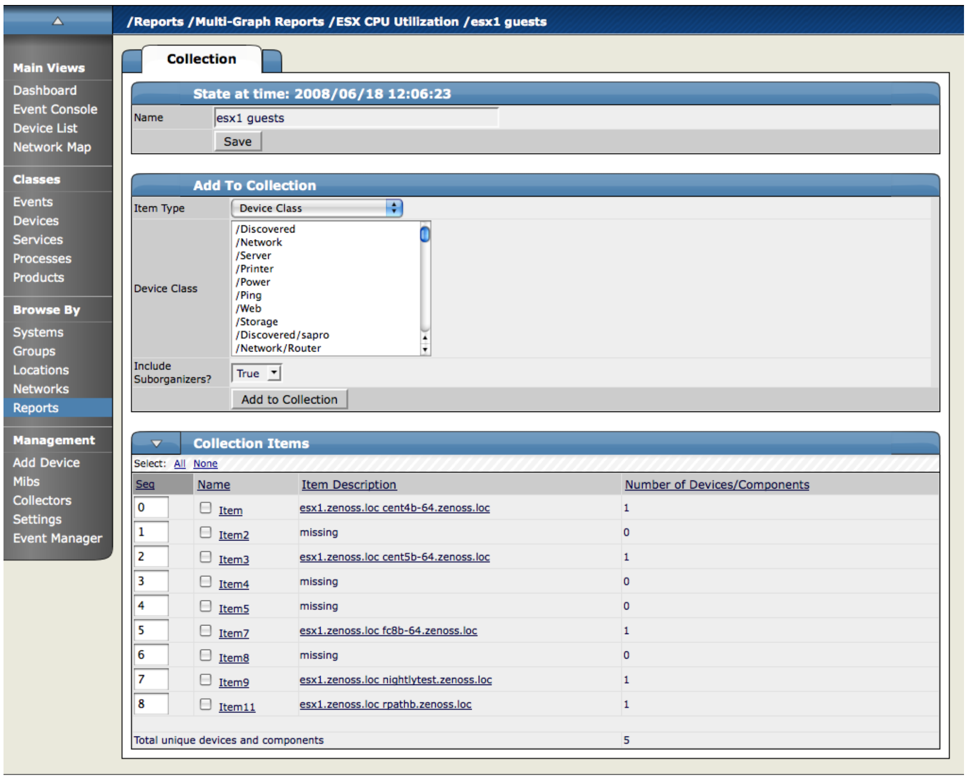
A Collection consists of one or more Collection Items. A Collection Item is a list of Device Classes, Systems, Groups, Locations or specific Devices and Components that should be included in this Collection. You can create as many Collection Items of the various types as you wish within a single Collection. The controls for creating Collection Items are in the Add To Collection table of the Collection edit page. The Item Type menu lets you select one of the following:

- Device Class/System/Group/Location - Selecting one of these options reveals a list of all organizers of that type. You can select one or more of the organizers to include in the Collection. By selecting True for the Include Suborganizers field the Collection will also include all organizers recursively beneath the ones you selected. These Collection Items are dynamic - when devices are added or removed from these organizers they will appear or disappear from the report. Clicking the Add to Collection button creates a new Collection Item for each of the selected organizers.

- Specific Device/Component - Selecting this type reveals a list of all Devices in Zenoss. You can use the Filter field to narrow this list by entering a full or partial Device name. Selecting one or more Devices will display a list of Component names that apply to one or more of the selected Devices. If you do not select any Components and click the Add to Collection button then a new Collection Item is created for each selected Device.

Once you have specified an Item Type and made your selection click the Add to Collection button to create the new Collection Item. It will be added to the list of Collection Items at the end of the page. Collection items can be deleted or reordered within this list. Order of the Collection Items determines the order that the graphs are drawn in or the order that data is drawn on a combined graph. See the section on Graph Groups for more details.

Figure 23.7. MultiGraph Report Collection



/Reports /Multi-Graph Reports /ESX CPU Utilization /esx1 guests

State at time: 2008/06/18 12:06:23

Name	esx1 guests

Add To Collection

Item Type	Device Class
Device Class	/Discovered /Network /Server /Printer /Power /Ping /Web /Storage /Discovered/sapro /Network/Router
Include Suborganizers?	True

Select: All None

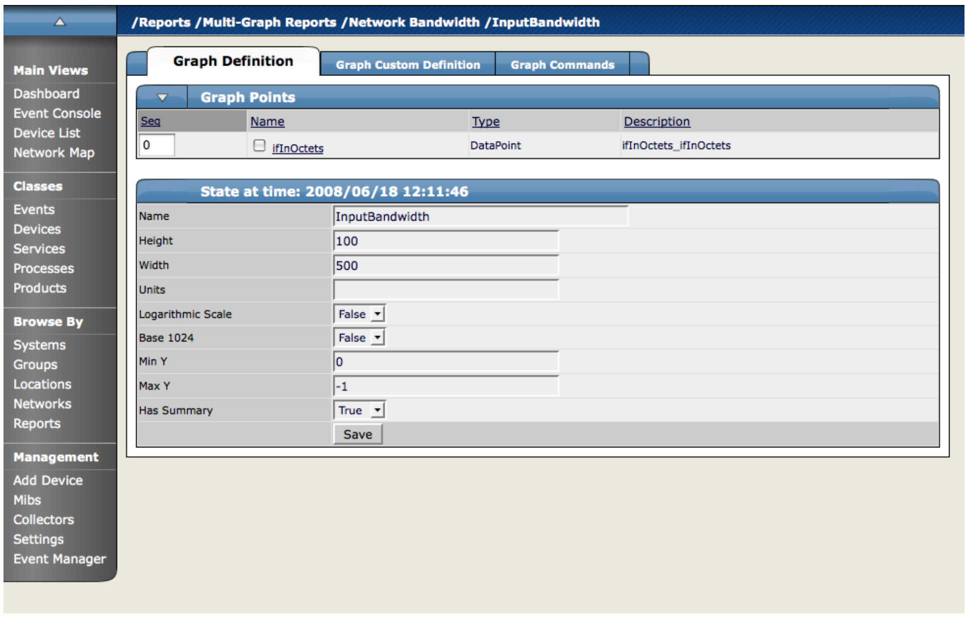
Seq	Name	Item Description	Number of Devices/Components
0	Item	esx1.zenoss.loc cent4b-64.zenoss.loc	1
1	Item2	missing	0
2	Item3	esx1.zenoss.loc cent5b-64.zenoss.loc	1
3	Item4	missing	0
4	Item5	missing	0
5	Item7	esx1.zenoss.loc fc8b-64.zenoss.loc	1
6	Item8	missing	0
7	Item9	esx1.zenoss.loc nightlytest.zenoss.loc	1
8	Item11	esx1.zenoss.loc rpathb.zenoss.loc	1
Total unique devices and components			5

7.3. Graph Definitions

Graph Definitions in the context of MultiGraph Reports are very similar to those in Performance Templates. Settings on the Graph Definition itself define some basic parameters, then you add Graph Points to specify which data should be drawn. See the section on Graph Definitions in the Performance Chapter for details on creating Graph Definitions.

The most significant differences between Graph Definitions in the two contexts is how DataPoint Graph Points and Threshold Graph Points are added. When adding a DataPoint Graph Point to a Graph Definition within a Performance Template you can select from a list of DataPoints that are defined on that Template. But within the context of a MultiGraph Report there are no GraphPoint definitions to list. So instead of listing the available DataPoints, the DataPoint Graph Point dialog has a text field where you enter the name of the DataPoint. To make things easier the input has an auto-complete feature which knows the names of every DataPoint defined in Zenoss. This same situation is true with Threshold Graph Points.

Figure 23.8. MultiGraph Report Graph Definition



/Reports /Multi-Graph Reports /Network Bandwidth /InputBandwidth

Graph Points

Seq	Name	Type	Description
0	ifInOctets	DataPoint	ifInOctets_ifInOctets

State at time: 2008/06/18 12:11:46

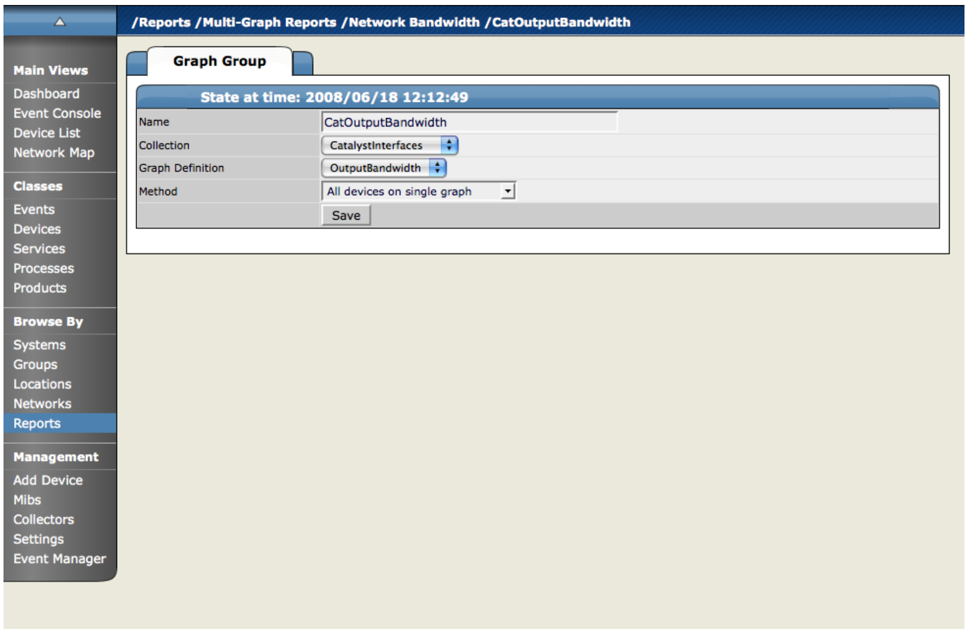
Name	InputBandwidth
Height	100
Width	500
Units	
Logarithmic Scale	False
Base 1024	False
Min Y	0
Max Y	-1
Has Summary	True

7.4. Graph Groups

Graph Groups are used to combine one Graph Definition with one Collection to produce graphs for the report. You must have at least one Graph Group or your report will have no graphs. You create a new Graph Group by selecting the Add Group menu item from the Graph Groups table menu. After entering a name for the Graph Group you are presented with the Graph Group edit page. There are 4 settings on this page:

- Name - This is used to identify the Graph Group on the MultiGraph Report page. It does not appear anywhere on the actual report.
- Collection - Select one of the Collections that have been defined for this report.
- Graph Definition - Select one of the Graph Definitions that have been defined for this report.
- Method - There are two options for applying the Graph Definition you selected to the Collection that you selected: * Separate graph for each device: The Graph Definition will be used to draw one graph for each Device and Component listed in the Collection. The graphs will appear in the list in roughly the same order they are specified within the Collection. * All devices on a single graph: This draws one graph with the data from all the Devices/Components on it.

Figure 23.9. MultiGraph Report Graph Group



State at time: 2008/06/18 12:12:49

Name	CatOutputBandwidth
Collection	CatalystInterfaces
Graph Definition	OutputBandwidth
Method	All devices on single graph

7.5. Graph Order

Graph Groups are drawn in the order they are listed on the MultiGraph Report edit page. You can change the order of the Graph Groups by editing the sequence number next to each then using the Re-Sequence Items menu item from that table's menu. If a Graph Group results in multiple graphs, the graphs are drawn in the order that the Collection Items are listed within the corresponding Collection. If a Collection Item specifies a Device organizer then the order of the Devices drawn from that Collection Item is indeterminate.

As with Graph Reports, if you have specified multiple columns for a report then the graphs are drawn left to right in that number of columns using as many rows as necessary.

8. Creating Custom Reports

There are a few ways to create reports of your own. You can create some reports through the Zenoss user Interface or using the Zope management Interface (ZMI).

8.1. Creating Custom Reports Using the ZMI

Zenoss Reports are written in python and templates are available through the Zope Management Interface (ZMI). To access the ZMI for a particular page, add a /manage to whichever report you are looking at. The ability to create individual reports is largely dependent on your ability to create python strings to get and display the information. More details on this will be forthcoming as we add more new reports and create more.

8.2. Create A Custom Device Report Example

Here are the steps for creating a Custom Device Report that will show device name, network address, and device serial number.

1. From the left navigation menu, select Reports.

2. From the Report Organizers list, click the Device Reports link.

   This is where you will be adding a new report.

3. From the bottom of the page enter a name for this custom report in the Add text box.

4. Click the Add button.

   The report you just created will appear in the Report list.

5. Click the name of the new report from the list.

   The Report detail page appears.

6. Click the Edit tab to define the report parameters.

7. Fill out the fields for this report as follows.

   - Name- The name you want to give to the report.
   - Title - The way the report is named in the display. Can be different from the name.
   - Path -The path in the hierarchy where the report will be stored.
   - Query - The actual query string for If you want to limit the report to just those devices that have a serial number, you can set the Query value to:

     here.hw.serialNumber != ""
   - Sort Column - What column you want to use to sort the report.
   - Sort Sense - The sense that the system uses to sort (For example asc is ascii)
   - Columns These columns are the actual data that be retrieved and displayed in the report.

     For example:

     getId – would get the name of any devices

     getManageIp – would get the IP addresses of the devices

     getHWSerialNumber – will grab serial numbers for device.
   - Column Names - You can add column names to make the column headers more descriptive.

     For the example columns above you could use the column names:

     Device

     Address

     Serial #

     NOTE: The information that appears in the fields and how you actually get this information is found in the admin guide in the Tales Expressions appendix in the Device schema section.

8. Click the Save button.

9. Now click the Report tab at the top of the page.

   The new device report appears showing the devices that meet the criteria.

9. Using Reports to Help Troubleshoot Zenoss Daemons

This section will show you how to find and view certain reports that will aid you in troubleshooting Zenoss daemons.

From the Zenoss web interface, navigate to Reports. Follow the path to the various reports listed below to see the reports. The troubleshooting items on the right give you clues as to what to expect to find in the various reports.

Troubleshooting Items	Report Name	Where It Lives
zenmodeler issues	Model Collection Age	/Reports/Device Reports
Any internal Zenoss issues	All Heartbeats	/Reports/Event Reports/
zendisc errors, adding devices	New Devices	/Reports/Device Reports
Any alerting rule issues, will show all rules in the system	Notification Schedules	/Reports/User Reports
Good summary of snmp status across the system including non-monitored	SNMP Status Issues	/Reports/Device Reports
Which devices are monitored and whether ping is turned on or off	Ping Status Issues	/Reports/Device Reports

Chapter 24. Advanced Zenoss Reports (Commercial)

1. Additional Reports

ZenEnterpriseReports adds three new reports to the standard core reports. The three additional reports are:

- Maintenance Windows Report - showing all maintenance windows within the Zenoss instance. This report is useful for tracking system up and down times.
- Organizer Availability Report - Gives you the availability percentage of all network organizers in the system. This report can be filtered by Organizer, Event Class, Component and Date.
- Systems Availability report that shows you systems availability in time and percentage and is filterable by date.

Chapter 25. General Zenoss Administration

1. Working with Zenoss from the Command Line

When working with Zenoss from the command line, you always want to be logged in as the user “zenoss”. If you log in as root you must become zenoss by issuing the command:

su - zenoss

2. Minimal Zenoss - ZEO and Zope

The Zenoss user interface can run without any other Zenoss processes running. In this mode, only ZEO and Zope are running.

This mode is useful for troubleshooting many Zenoss issues.

1. Stop all Zenoss processes.

   ```
   $ zenoss stop
   ```

2. Start the Zope object database.

   ```
   $ zeoctl start
   ```

3. Start Zope

   ```
   $ zopectl start
   ```

4. Go to the web UI and confirm that you can access the Dashboard.

5. Start the other Zenoss daemons.

3. Checking the Version of Zenoss

You can use the Version tab to check the version of Zenoss and all of the associated components. To access this information from the Navigation menu in the Dashboard, click Settings. Now click the Versions tab. The Versions tab appears.

This tab shows the version of the following Zenoss components:

- Zenoss
- Zope
- Database
- Twisted
- OS
- Python
- RRD

- SNMP

  This page also gives the Zope uptime.

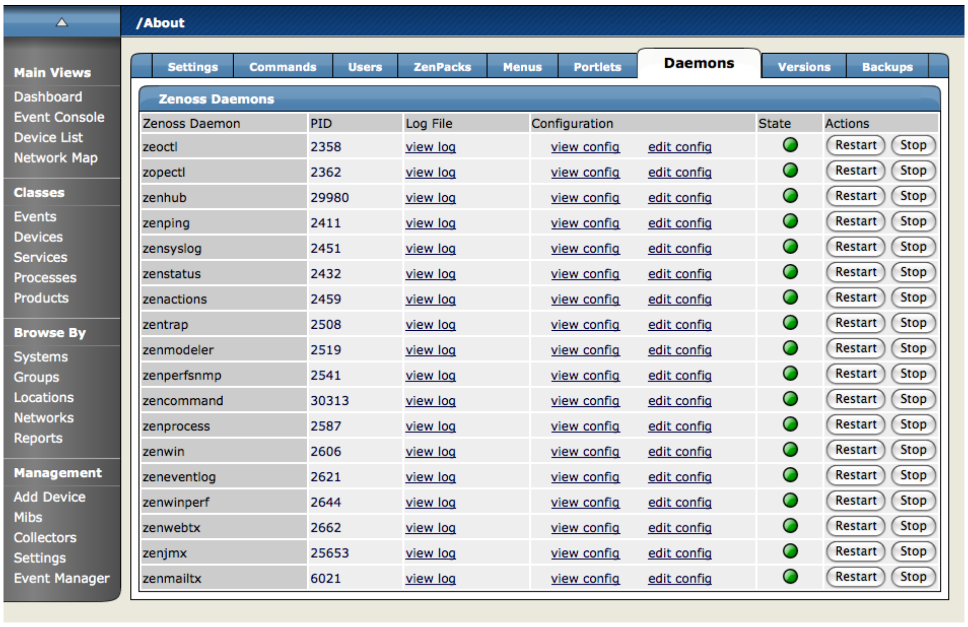
4. Checking for Zenoss Updates

Use the Settings page Versions tab to check for updates to the Zenoss software. You can have Zenoss to check daily or click the Check Zenoss Version Now button to check for an update. You can also see the last time a check for updates was made.

5. Starting and Stopping the Zenoss Daemons

You can use the Zenoss management console to start and stop the Zenoss daemons as well as check the status of the daemons. To access this information from the Navigation menu in the Dashboard, click Settings tab and then click the Daemons tab.

Figure 25.1. Settings Page - Daemons Tab



Zenoss Daemons

Zenoss Daemon	PID	Log File	Configuration		State	Actions	
zeoctl	2358	view log	view config	edit config	●	Restart	Stop
zopectl	2362	view log	view config	edit config	●	Restart	Stop
zenhub	29980	view log	view config	edit config	●	Restart	Stop
zenping	2411	view log	view config	edit config	●	Restart	Stop
zensyslog	2451	view log	view config	edit config	●	Restart	Stop
zenstatus	2432	view log	view config	edit config	●	Restart	Stop
zenactions	2459	view log	view config	edit config	●	Restart	Stop
zentrap	2508	view log	view config	edit config	●	Restart	Stop
zenmodeler	2519	view log	view config	edit config	●	Restart	Stop
zenperfsnmp	2541	view log	view config	edit config	●	Restart	Stop
zencommand	30313	view log	view config	edit config	●	Restart	Stop
zenprocess	2587	view log	view config	edit config	●	Restart	Stop
zenwin	2606	view log	view config	edit config	●	Restart	Stop
zeneventlog	2621	view log	view config	edit config	●	Restart	Stop
zenwinperf	2644	view log	view config	edit config	●	Restart	Stop
zenwebtx	2662	view log	view config	edit config	●	Restart	Stop
zenjmx	25653	view log	view config	edit config	●	Restart	Stop
zenmailtx	6021	view log	view config	edit config	●	Restart	Stop

This tab is made up of a list of all of the Zenoss daemons, their process IDs (PIDs), a status indicator and then an Action area where you can restart a process or stop it. If a daemon is stopped, the Restart button changes to start.

6. Zenoss Daemon Commands and Options

All Zenoss daemons share some similar commands. These commands can be run from the command line for the device where Zenoss is installed. Each daemon has the following commands:

- run – start daemon but don’t put it into the background. This is good for debugging.
- start – start as a daemon running in the background, detached from the shell.
- stop – stop the daemon.
- restart – stop/start the daemon. Sometimes the start command will run before the daemon has terminated. If this happens just re-run the command.
- status – check the status of a daemon will print out the current process number if running.

- help – display a list of all options for the daemon.

6.1. Configuring Zenoss Daemons

Any daemon can be configured by adding key/value pairs to a file named $ZENHOME/etc/DAEMONNAME.conf. Valid keys are the long option of any command line option. These can be listed by using the daemons “help” command.

6.2. General Options for All Daemons

Command	Description
--version	show program's version number and exit
-h, --help	show this help message and exit
-vLOGSEVERITY,--logseverity=LOGSEVERITY	Logging severity threshold
--logpath=LOGPATH	override default logging path
-CCONFIGFILE,-- configfile=CONFIGFILE	config file
--uid=UID	user to become when running default:zenoss
-c, --cycle	Cycle continuously on cycleInterval from zope
-D, --daemon	Become a Unix daemon
--host=HOST	hostname of zeo server
--port=PORT	port of zeo server
-RDATAROOT, --dataroot=DATAROOT	root object for data load (i.e. /zport/dmd)
--cachesize=CACHESIZE	in memory cachesize default: 1000
--pcachename=PCACHENAME	persistent cache file name default:None
--pcachedir=PCACHEDIR	persistent cache file directory

6.3. zenhub Options

Command	Description
-x XMLRPCPORT, --xport=XMLRPCPORT	Port to listen to for XML RPC calls
--pbport=PBPORT	pb port
--passwd=PASSWORDFILE	Location the password file is stored
--workers=NUMBER	Allows zenhub to use subordinate processes to handle collector requests. Use to reduce the load on zenhub without resorting to a multiple-hub configuration. The value of NUMBER must be an integer greater than zero.

6.4. zenmodeler Options

Command	Description
--debug	don't fork threads for processing
--parallel=PARALLEL	number of devices to collect from in parallel
--cycletime=CYCLETIME	run collection every x minutes
--ignore=IGNOREPLUGINS	Comma separated list of collection maps to ignore
--collect=COLLECTPLUGINS	Comma separated list of collection maps to use
-pPATH, --path=PATH	start path for collection ie /NetworkDevices

Command	Description
-dDEVICE, --device=DEVICE	fully qualified device name ie: www.zenoss.com
-aCOLLAGE, --collage=COLLAGE	do not collect from devices whose collect date is within this many minutes
--writetries=WRITETRIES	number of times to try to write if a read conflict is found
-F, --force	force collection of config data (even without change to the device)
--portscantimeout=PORTSCANTIMEOUT	time to wait for connection failures when port scanning
-uUSERNAME, --user=USERNAME	Login username
-PPASSWORD, --password=PASSWORD	Login password
-tLOGINTRIES, --loginTries=LOGINTRIES	number of times to try login
-LLOGINTIMEOUT, --loginTimeout=LOGIN-TIMEOUT	timeout login expect statements
-TCOMMANDTIMEOUT, --commandTimeout=COM-MANDTIMEOUT	timeout when issuing a command
-KKEYPATH, --keyPath=KEYPATH	Path to use when looking for keys
-sSEARCHPATH, --searchPath=SEARCHPATH	Path to use when looking for commands
-eEXISTENCETEST, --existenceTest=EXISTEN-CETEST	how to check for command
-rPROMPTTIMEOUT, --promptTimeout=PROMPT-TIMEOUT	timeout when discovering prompt
-xLOGINREGEX, --loginRegex=LOGINREGEX	regex that will find the login prompt
-wPASSWORDREGEX, --passwordRegex=PASS-WORDREGEX	regex that will find the password prompt
--enable	enter enable mode on a cisco device
--termlen	enter send terminal length 0 on a cisco device
--enablePause=ENABLEPAUSE	time to wait before sending enable command
--enableRegex=ENABLEREGEX	regex that will find the enable password prompt

NOTE: --ignore and --collect are mutually exclusive

6.5. zenperfsnmp Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSER-NAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.6. zenperfxmlrpc Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server

Command	Description
-uZOPEUSERNAME, --zopeusername=ZOPEUSER-NAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.7. zenprocess Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSER-NAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.8. zenping Options

Command	Description
--configpath=CONFIGPATH	path to our monitor config ie: /Monitors/StatusMonit-ors/localhost
--name=NAME	name to use when looking up our record in the dmd de-faults to our fully qualified domain name as returned by getfqdn
--test	Run in test mode: doesn't really ping, but reads the list of IP Addresses that are up from /tmp/testping

6.9. zensyslog Options

Command	Description
--statcycle=STATCYCLE	Number of seconds between the writing of stats
--dmdpath=DMDPATH	zope path to our dmd /zport/dmd
--parsehost	try to parse the hostname part of a syslog HEADER
--stats	print stats to log every 2 seconds
--logorig	log the original message
--debug	debug mode no threads
--minpriority=MINPRIORITY	Minimum priority that syslog will accept
--heartbeat=HEARTBEAT	Number of seconds between heartbeats
--syslogport=SYSLOGPORT	Port number to use for syslog events

6.10. zenstatus Options

Command	Description
--configpath=CONFIGPATH	path to our monitor config ie: /Devices/Server

Command	Description
--parallel=PARALLEL	number of devices to collect at one time
--cycletime=CYCLETIME	check events every cycletime seconds

6.11. zenactions Options

Command	Description
--cycletime=CYCLETIME	check events every cycletime seconds
--fromaddr=FROMADDR	address from which email is sent
--zopeurl=ZOPEURL	http path to the root of the zope server

6.12. zentrap Options

Command	Description
--statcycle=STATCYCLE	Number of seconds between the writing of stats
-tTRAPPORT, --trapport=TRAPPORT	Port number for listening for traps

6.13. zencommand Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSER-NAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration
--parallel=PARALLEL	number of devices to collect at one time

7. Troubleshooting Zenoss Daemons

When Zenoss is having an issue, it will write stack traces into its log files. These log files are important for deciphering exactly what went wrong with the application. This section demonstrates how to generate a stack trace. This will generate a stack trace where you can see errors in the log.

1. While logged in as the zenoss user, stop all zenoss processes.

2. Shut Down MySQL.

   ```
   $ sudo service mysqld stop
   ```

3. Start the Zope Object Database.

   ```
   $ zeoctl start
   ```

4. Run zenperfsnmp in the foreground in debug mode.

   ```
   $ zenperfsnmp run
   ```

5. Notice there are stack traces saying that the daemon can’t connect to zenoss.

6. Start Zope.

   ```
   $ zopectl start
   ```

7. Go to dashboard and see the error message “Lost connection to Zenoss”. This error on the dashboard can appear for many reasons related to the loss of connectivity or some back end failure of the system.

8. Look for a more useful error in the zope log file $ZENHOME/log/event.log.

9. Restart mysqld.

   ```
   $ sudo service mysqld start
   ```

10. Restart Zenoss.

    ```
    $ zenoss start
    ```

11. Look at the end of all zenoss log files for any errors.

    ```
    $ tail $ZENHOME/log/*.log | less
    ```

8. Automatic Startup in Linux Environments

Zenoss can be controlled entirely by the bin/zenoss script. To provide automatic startup in Linux environments, you must link Zenoss to the /etc/rc.d/init.d directory

```
$ ln -s $ZENHOME/bin/zenoss /etc/rc.d/init.d
```

It is important to either add $ZENHOME to the root environment or to the zenoss script itself.

9. Using Zenoss with a Remote MySQL Instance

If you are interested in using a remote MySQL database for Zenoss, set it up the same way you would set it up if you were using a native database. Then log into the management console.

1. First change the database address for the Backend Type for the mysql database.

   From the Zenoss Dashboard, in the Navigational menu, select Event Manager.

   The ZenEventManager page appears.

2. Change the Hostname field to the address of the MySQL database you want to use.

3. Restart Zenoss.

   The MySQL database you entered is now the default database.

10. Loading and Registering MIBs with Zenoss

The MIB loader is called zenmib. To load MIBs, first copy them to $ZENHOME/share/mibs/site. Then run zenmib using the command:

```
zenmib run mibfile
```

The MIB is now loaaded into Zenoss. You can see the MIBs you have loaded by using the left navigation menu and clicking the Mibs link.

10.1. Resolving MIB Dependencies

You loaded aaa.mib, and when you attempt to load bbb.mib, you get this error:

```
INFO:zen.zenmib:Unable to find a file providing the MIB AAA-MIB
```

To resolve the dependency, load them like this

```
zenmib run aaa.mib bbb.mib
```

Chapter 26. Backup, Recovery and Maintenance

1. Backup and Restore

Zenoss provides tools to backup the configuration and data from a Zenoss install and restore that information later. This can be useful in taking periodic snapshots of your install for backup purposes, moving your data from one Zenoss installation to another or restoring your setup and performing a fresh install of Zenoss. These are the specific items that are included in the backup and restore:

- The entire events database in mysql.
- The Zope database, which includes all devices, users, event mappings, etc.
- The $ZENHOME/etc directory, which contains config files for the zenoss daemons.
- The $ZENHOME/perf directory, which contains performance data,

  The sections below describe in detail the backup and restore scripts and the options for controlling their behavior. Typical use of zenbackup looks like:

  ```
  > zenbackup --save-mysql-access --file=BACKUPFILEPATH
  ```

  Typical use of zenrestore looks like:

  ```
  > zenrestore --file=BACKUPFILEPATH
  ```

  Suggestions for a satisfying backup/restore experience:

  - If you have the available disk space, tar and zip $ZENHOME before starting any backup or restore operation. This gives you a chance to recover in case something goes awry.
  - Make sure Zenoss, including all daemons, is stopped before performing a restore.
  - Using these tools to go from a newer version of Zenoss to an older version could be bad news and should really be avoided.
  - If you use these tools to go from an older version of Zenoss to a newer version you should run zenmigrate after the restore.
  - If restoring to a different zenoss installation than the backup is initially from, make sure file paths in the $ZENHOME/etc/*.conf files are appropriate for the new environment after you restore.

1.1. Backup Details

The script for backup is $ZENHOME/bin/zenbackup. If zenoss is running then you can run zenbackup without any arguments and a backup file will be placed in $ZENHOME/backups. zenbackup --help will give a full list of the available options. Some of the more interesting options are:

--dbname

This is the name of the mysql database zenoss uses to hold event data. By default this is "zenoss" but this can be specified when zenoss is installed. This value can be seen by looking at the database field on the Event Manager page in zenoss. If you don't specify --dbname then zenbackup will attempt to retrieve this information from zeo unless you specify --dont-fetch-args.

--dbuser, --dbpassword

These are the mysql username/password used to access the events database. If you don't specify --dbuser or --dbpassword then zenbackup will attempt to retrieve this information from zeo unless you specify --dont-fetch-args.

--dont-fetch-args

This instructs zenbackup not to attempt to get values for dbname, dbuser and dbpassword from zeo.

--file=FILE

Use --file to specify a location for the backup file. By default it will be named zenoss_<DATE>.tgz and placed in $ZENHOME/backups.

--stdout

This flag tells zenbackup to send the backup information to stdout instead of to a file. Incompatible with --verbose.

--save-mysql-access

This instructs zenbackup to save dbname, dbuser and dbpassword as part of the backup file so that zenrestore will have this information during a restore operation. Use this with caution as it means your backup files will contain a mysql username and password.

--no-eventsdb

Do not include the mysql events database as part of the backup.

-v, --verbose

Print progress messages. Incompatible with --stdout.

1.2. Backups Tab

Zenoss provides a simple web GUI for creating and managing backups. Navigate to Settings->Backups to view the Backups page. The Create New Backup section allows you to create a backup through the GUI. The options available are a subset of those available with the zenbackup command line tool (see above for details on the options.) Below that is the Backups section which lists all backup files in $ZENHOME/backups. You can delete one or more backup files by selecting them using the checkboxes and selecting the Delete Backup... menu item. Backup files can become large as your databases grow, so you may want to limit the number of backups you keep if drive space becomes an issue.

1.3. Remote Backups

Keeping backups on your zenoss server should help you recover if one of your databases becomes corrupt, your configuration becomes problematic, etc. It will probably not help you recover from a situation where your disk fails or any other failure that affects the system as a whole. For this reason it is advisable to keep at least one recent backup file on a different server, ideally at a different physical location.

1.4. Restore Details

The script for restoring zenoss from a backup is $ZENHOME/zenrestore. Make sure that zenoss is stopped before performing a restore. If you used the --save-mysql-access option when you created the backup file then you only need to specify --file when you run zenrestore. Otherwise you need to specify dbname, dbuser and dbpassword also.

--file

This is a backup file created with zenbackup You must specify either --file or --dir.

--dir

The path to an unzipped backup file. You must specify either --file or --dir.

--dbname

This is the name of the mysql database zenoss uses to hold event data. This database must exist before zenrestore is run. If there are any zenoss tables in the database they will be dropped by zenrestore before it restores the backed up tables and data. If you use a different dbname than was in use when the backup was created then after the restore you'll need to set the database name on the Event Manager page.

--dbuser, --dbpassword

These are the mysql username/password used to access the events database. If you don't specify --dbuser or --db-password then zenrestore will attempt to use values stored in the backup file if --save-mysql-access was used in creating it.

--no-eventsdb

Do not restore the mysql events database. If the backup file does not contain mysql events data then zenrestore will not modify your events database even if you do not specify --no-eventsdb.

-v, --verbose

Print progress messages.

1.5. Periodic Backups

Backing up your Zenoss data and configuration is a very good idea. $ZENHOME/bin/zenbackup can create backups that include your zeo database (devices, etc), RRD files (performance data), MySQL tables (events) and your Zenoss configuration files. See section **** for information on using zenbackup and zenrestore.

1.5.1. Pack ZEO Database

The Zeo database needs to be packed periodically to reclaim space. To do this you should set up a cron job that runs the following command weekly:

```
$ZENHOME/bin/zeopack.py -p 8100
```

1.5.2. Log Rotate Script

If your system uses logrotate to manage files put the following in /etc/logrotate.d/zenoss to manage Zenoss' log files:

```
/usr/local/zenoss/log/*.log {
weekly
rotate 2
copytruncate
}
```

1.5.3. Backing up the MySQL Event Backend

MySQL should be backed up following the MySQL manual.

Chapter 27. ZenPacks

1. Introduction to ZenPacks

A ZenPack is a package that adds new functionality to the Zenoss. A ZenPack may add Action Rules, Event Classes, Event Commands, User Commands, Service Classes, Data Sources, Graphs, Performance Templates, Reports, Model Extensions or Product Definitions. A ZenPack may also add new daemons and new UI features such as menus.

ZenPacks are a mechanism for extending and modifying Zenoss. This can be as simple as adding new Device Classes or Performance Templates or as complex as extending the data model and providing new collection daemons. ZenPacks can be distributed for installation on other Zenoss systems. Simple ZenPacks can be created completely within the Zenoss user interface while more complex ZenPacks require development of scripts or daemons in Python or another programming language.

For example, say you have developed a Performance Template for a new piece of hardware. You've created Data Sources for the OID's you think are worth monitoring, Thresholds to make sure some of these values stay within reasonable limits and several Graph Definitions to show this data graphically. Perhaps you've also created a new Device Class for this hardware. You can create a ZenPack to easily distribute your Performance Template and Device Class to other Zenoss administrators. This ZenPack can be entirely created from within the Zenoss user interface.

For another example, say you want to monitor a new piece of software running on one of your servers. You would like to monitor several performance metrics of this software, but they are available only via a programmatic API provided with the software. You could develop a new collector daemon to gather data via this API and provide it back to Zenoss. You might also create a new type of Data Source to provide configuration data for the new collector. Obviously this effort would require development skills and intimate knowledge of Zenoss not necessary for the previous example, but this functionality can still be distributed as a ZenPack.

1.1. Installing ZenPacks

ZenPacks are usually distributed as .egg files. Zenoss also supports .zip files, though support for this format will be removed in a future version of Zenoss. Either ZenPacks of either type can be installed from the command line on the Zenoss server or via the Zenoss user interface.

1.1.1. Installing via the Command Line

The following ZenPack command can be used from the command line to install ZenPack files. After installing or updating ZenPacks you need to restart Zope and ZenHub:

```
zenpack --install <filename>
zopectl restart
zenhub restart
```

If you have the source code for the ZenPack you can install directly from that rather than from a .egg or .zip file. The command is the same, you just specify the directory containing the source code. This copies the source code into either $ZENHOME/ZenPacks (for newer egg ZenPacks) or $ZENHOME/Products (for older style ZenPacks.)

```
zenpack --install <directoryname>
zopectl restart
zenhub restart
```

If you are developing a ZenPack you usually will want to maintain your source code outside of $ZENHOME/ZenPacks or $ZENHOME/Products. This is advisable for two reasons. First, if you issue a zenpack --remove command it will delete your code from either of those two locations and you would lose your files unless you had them backed up elsewhere. Second, if you are maintaining your source code in a version control system it is frequently more convenient to have the files reside elsewhere on the filesystem. Using the --link option you can install the ZenPack but have Zenoss use your code from its current location. Instead of installing your code in $ZENHOME/Zen-

Packs or $ZENHOME/Products Zenoss will create a link in one of those locations that points to your source code directory.

```
zenpack --link --install <directoryname>
zopectl restart
zenhub restart
```

1.1.2. Installing via the User Interface

You can upload and install a ZenPack .egg or .zip file via the user interface. From the Settings->ZenPacks page choose the Install Zenpack... menu item. In the dialog that follows use the Browse button to select the .zip file from your local computer. When you click the OK button the file is uploaded to the Zenoss server and installed.

1.1.3. Installing All Core ZenPacks via RPM

The Zenoss Core ZenPacks, along with third party ZenPacks, are available for download individually from http://www.zenoss.com/community/projects/zenpacks/. Also on that page is a link to download an RPM that includes the most popular Core ZenPacks. To install via the Core ZenPacks RPM follow these steps:

1. Download the appropriate file from http://www.zenoss.com/community/projects/zenpacks/all-core-zenpacks
2. Make sure zeo is running (as zenoss user):

   ```
   zeoctl start
   ```

3. Install the rpm (as root user):

   ```
   rpm -ihv <rpm file>
   ```

4. Restart Zope and ZenHub:

   ```
   zopectl restart
   zenhub restart
   ```

1.2. Creating a ZenPack

When logged into Zenoss as an Administrator click on the Setting link and then on the ZenPacks tab. Select the "Create a ZenPack..." menu item. You will get a dialog asking for a name for your new ZenPack. The name must be of the form ZenPacks.<organization>.<identifier>, where organization is a name that identifies you or your organization and identifier is a string that represents the intent of your ZenPack. For example, ZenPacks.zenoss.HttpMonitor was created by zenoss to help monitor HTTP sites. Once you have entered a name click the save button. This creates both the ZenPack object in the database as well as a new directory in the filesystem $ZENHOME/ZenPacks/<your zenpack id>.

Many types of objects can be added to a ZenPack via the user interface. Some examples are:

- Device Classes
- Event Classes
- Event Mappings
- User Commands
- Event Commands
- Service Classes
- Device Organizers

- Performance Templates

Devices are the conspicuous omission from this list. Any individual Device is usually specific to a particular site and therefore not likely to be useful to other Zenoss users.

To add one of these database objects to a ZenPack navigate to that object and use the "Add to ZenPack..." menu item. Zenoss will present a dialog which lists all installed ZenPacks. Select the ZenPack to which you want to add this object and click the Add button. To view the objects that are part of a ZenPack navigate to the Settings page then the ZenPacks tab. Click on the name of the ZenPack and you will see a page that lists both the files and the objects that are part of this ZenPack. You can remove objects from the ZenPack by selecting the checkboxes next to them and using the "Delete from ZenPack..." menu item.

ZenPacks can contain items that are not zeo database items, such as new daemons, Data Source types, skins, etc. These are added to a ZenPack by placing them in the appropriate subdirectory within the ZenPack's directory. See the Core ZenPacks at http://www.zenoss.com/community/projects/zenpacks/ for examples of how to incorporate such items into your ZenPack. Further information regarding ZenPack development is available in the Zenoss Developers Guide.

Discussion regarding development of ZenPacks takes place on the zenoss-dev mailing list and forums: http://community.zenoss.com/forums/viewforum.php?f=3

1.2.1. Packaging and Distributing Your ZenPack

ZenPacks are usually distributed as .egg files. To create the installable .egg file for a ZenPack use the "Export ZenPack..." menu item in the page menu when viewing a ZenPack. The dialog that follows has two options. The first option simply exports the ZenPack to a file named <ZenPackId>.egg in the $ZENHOME/exports directory on the Zenoss server. The second option does the same but then downloads the exported file to your browser. Other Zenoss administrators can install this exported .egg file as described in the Installing ZenPacks section.

For information on how to make your ZenPack available on the zenoss.com site see http://www.zenoss.com/community/projects/zenpacks/how-to-contribute-a-zenpack

1.3. Removing a ZenPack

Warning: Removing a ZenPack can have unexpected consequences. For example, removing a ZenPack that installed a Device Class will remove both the Device Class and all Devices within that class. Also, before removing a ZenPack you should delete any Data Source of a type provided by the ZenPack. You should always perform a backup of your Zenoss data before removing a ZenPack. See Section 21.1 Backup and Restore for information on how to backup your Zenoss data.

2. Zenoss Core ZenPacks

2.1. ZenJMX ZenPack

2.1.1. About ZenJMX

ZenJMX is a ZenPack that allows Zenoss to communicate with remote JMX agents. The ZenJMX ZenPack introduces a data source of type 'JMX'. A 'JMX' data source allows you to define which JMX attributes should be monitored by Zenoss, as well as which operations you wish Zenoss to invoke. The ZenPack also introduces the zenjmx daemon, which is used to perform the actual retrieval of data from a JMX agent.

2.1.2. JMX Background

Java Management Extensions (JMX) are used throughout the Java Virtual Machine to provide performance and management information to clients. Using a combination of JConsole (Sun Microsystems' 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, Sun introduced the Remote API for Java Management Extensions. This enhancement defines an RMI wrapper around a JMX agent and allows for independent client development. ZenJMX 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.

2.1.3. ZenJMX Capabilities

ZenJMX 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 ZenJMX allows you to define the connection, authentication, and retrieval information you wish to use in order 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 some JMX agents in a hardened and authenticated fashion. SSL-wrapped connections are supported by the underlying JMX Remote subsystem built into the JDK but they 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 wish to achieve: single-value attribute, complex-value attribute, or operation invocation. Each is described in detail in following sections. To specify the type of retrieval you'll need to either specify an attribute name (and multiple or a single data point) or you'll need to provide operation information.

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

Tread carefully when selecting the 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). Make sure you understand the semantics of the attribute name and choose the proper Zenoss data point type when you set up your data point.

2.1.4. 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: counter)

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 the "Interrogating an JMX Agent via JConsole" section for additional information on how to inspect the MBeans deployed in an JMX Agent.

2.1.5. 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 with the following information:

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

2.1.6. Operation 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()".

ZenJMX has the ability to invoke operations, but there are some subtleties in how ZenJMX sends parameters to the JMX Agent and interprets the response.

2.1.6.1. Operation Calls: Scenario #1-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)

2.1.6.2. Operation Calls: Scenario #2-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. 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.

ZenJMX 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 ZenJMX follows:

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

2.1.6.3. Operation Calls: Scenario #3-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)

2.1.6.4. Operation Calls: Scenario #4-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[].

ZenJMX 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()
- Paramater Types: java.lang.Integer
- Parameter Values: 1234
- Data Points:
  - balance (type: gauge)

Here's 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()
- Paramater Types: java.lang.String
- Parameter Values: sbb552349999
- Data Points:
  - balance (type: gauge)

2.1.6.5. Scenario #5: Multiple parameters in polymorphic operations

The above 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 then the situation where a single parameter is passed to a polymorphic operation, except that the length of the Class[] and Object[] is > 1.

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 2 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.

2.1.7. ZenJMX Behavior

The ZenJMX 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.

When the "zenjmx" daemon is started it communicates with ZenHub and retrieves a list of devices that possess "JMX" data sources. It also spawns a Java process. ZenJMX 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 and returns the results to ZenJMX. Any collection or configuration errors are sent as events to Zenoss and will appear in the event console.

Lastly, ZenJMX heartbeats after each collect to ZenHub to let Zenoss know that ZenJMX is still alive and well.

2.1.8. Running the ZenJMX Daemon

The zenjmx daemon can be started by running "${ZENHOME}/bin/zenjmx start". Output from zenjmx is logged to ${ZENHOME}/log/zenjmx.log".

You can also run zenjmx in the foreground by running "${ZENHOME}/bin/zenjmx run". Additional parameters (such as --cycle or --cycleTime) can be provided after the "run" or "start" command. This is consistent with how other Zenoss daemons behave.

Most users will want to start the ZenJMX daemon in the background using the "${ZENHOME}/bin/zenjmx start" command and immediately follow that up with "tail -f ${ZENHOME}/log/zenjmx.log" to see what ZenJMX is doing.

2.1.9. Defining Custom JMX Data Sources

Custom JMX Data Sources allow system administrators to monitor any attribute or operation result accessible via a JMX call. ZenJMX 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 ZenJMX.

Start off by creating a new Performance template at the /Device level. The performance templates associated with a device are accessible via the More->Templates link when looking at a device's page. Click the second down arrow and select "Add Template". Provide a descriptive name for the template, such as "JVM Values". After the template is created click on it. You should now be looking at the "JVM Values" performance template.

Click the down arrow next to Data Sources and select "Add Data Source". Provide a descriptive name of the JMX value you wish to retrieve. In our case we are interested in memory information so set the ID to "Heap Memory". Set the Type to JMX.

Enter the JMX Management Port (not necessarily the same as the listen port for your server) and Object Name. The Object Name is also referred to as the MBean name. Enter "java.lang:type=Memory" as the Object Name. If your JMX Agent requires authentication provide the user name and password.

Enter "HeapMemoryUsage" as the Attribute Name. Then add gauge data 425 points named "committed", "max", and "used". Click Save.

Lastly, add graphs that reference these new data points.

Please review "Interrogating an JMX Agent via JConsole" to learn how to determine the object name, attribute name, and data points that might be interesting in your application.

2.1.10. Enabling Remote JMX Access

Each application server has a slightly different process for enabling remote JMX Access. It's best to consult with your application server for specific instructions. We've included instructions for a few commonly used configurations below.

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 the Remote Method Invocation (RMI) protocol 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 (via RMI). ZenJMX can only be used with remote servers via RMI and cannot work with local-only serialized MBeans. This is not a significant limitation because ZenJMX can establish RMI connections to localhost just as easily as it can establish RMI connections to remote hosts.

2.1.10.1. Remote JMX Access for the standard JVM

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/your/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.

2.1.10.2. Remote JMX Access for Tomcat

The same JAVA_OPTS approach can be used to enable remote access to Tomcat MBeans. Set the JAVA_OPTS variable as illustrated above and then execute the "./catalina.sh start" command in ${TOMCAT_HOME}/bin.

Note that Tomcat 6.0.14's catalina.sh does not process the "stop" command properly when the JAVA_OPTS variable is set. We recommend using 2 separate shells when troubleshooting JMX problems in Tomcat: one for starting Tomcat (with the JAVA_OPTS variable set) and a different one for stopping Tomcat (where the JAVA_OPTS variable is not set).

2.1.10.3. Remote JMX Access for JBoss

JBoss uses the JAVA_OPTS approach for enabling remote access to beans. However, it requires some additional properties. To set up your JAVA_OPTS for use in JBoss see the following code segment:

```
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"
JAVA_OPTS="${JAVA_OPTS} -Djboss.platform.mbeanserver"
JAVA_OPTS="${JAVA_OPTS} -Djavax.management.builder.initial=org.jboss.system.server.jmx.MBeanServerBuilder
export JAVA_OPTS
```

When you start JBoss via the run.sh you must also pass the "-b 0.0.0.0" argument:

```
cd ${JBOSS_HOME}/bin
./run.sh -b 0.0.0.0
```

JMX actually uses 2 separate ports for MBean access: one is used for initial connection handling and authentication, and the other is used for RMI access. During the handshake between a JMX Client and the JMX Agent the agent tells the client the IP address and port number for the RMI registry. By default JBoss sets the IP address to 127.0.0.1. This works when the JMX client and the JMX agent reside on the same device, but it won't work in a distributed environment.

By passing the "-b 0.0.0.0" argument you instruct JBoss to bind to all available network ports, and this results in the JMX Agent's handshaking logic using a network reachable address when informing clients of the RMI registry hostname and port.

2.1.10.4. Remote JMX Access for WebLogic

JSR-160 standardized remote access to JMX Agents, and allows for any client to connect to an JMX Agent using classes packaged with the JDK. WebLogic versions prior to 9.0 required clients to use a WebLogic JMX client library that used a proprietary protocol to interact with the JMX Agent. You'll need to make sure you're running WebLogic 9.0 or higher in order to monitor it via ZenJMX.

If you're new to WebLogic and have not set up a domain and server you'll need to run the startWLS.sh script located in ${BEA_HOME}/wlserver_10.0/server/bin. If you don't have the Terminal I/O package installed you can set the JAVA_OPTIONS variable to the following value:

```
JAVA_OPTIONS="-Dweblogic.management.allowPasswordEcho=true"
export JAVA_OPTIONS
```

Provide a user name and password to start WebLogic. Note that WebLogic requires a password that is at least 8 characters long. Wait for WebLogic to generate a configuration and start up. Shut down WebLogic and restart it with remote JMX access enabled.

To enable remote JMX access set the following variable:

```
JAVA_OPTIONS="-Dcom.sun.management.jmxremote.port=12347"
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.authenticate=false"
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.ssl=false"
export JAVA_OPTIONS
```

Then re-run the ./startWLS.sh script. JConsole can then communicate with the server on port 12347.

2.1.11. Interrogating an JMX Agent via JConsole

JConsole is a tool built into the JDK that allows system administrators to interrogate 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.

Figure 27.1. JMX Heap Graph

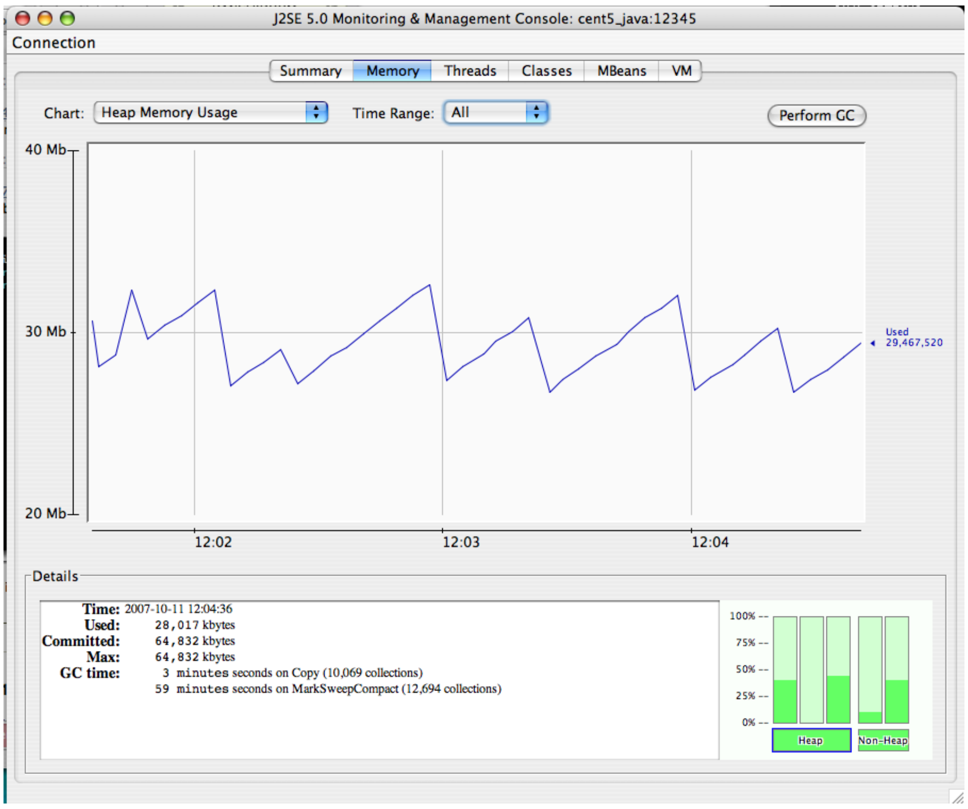


Chart: Heap Memory Usage
Time Range: All

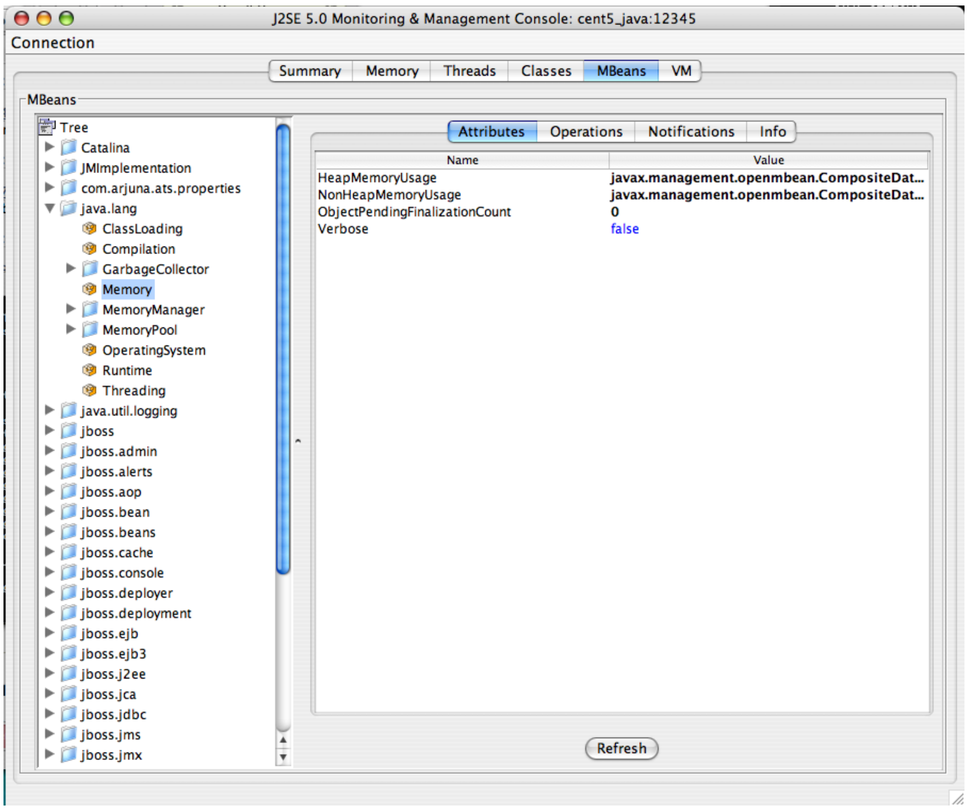
```
      Time: 2007-10-11 12:04:36
      Used:   28,017 kbytes
 Committed:   64,832 kbytes
       Max:   64,832 kbytes
   GC time:    3 minutes seconds on Copy (10,069 collections)
              59 minutes seconds on MarkSweepCompact (12,694 collections)
```

The MBean tab along the top of JConsole provides an interactive method for examining MBean values. After clicking on the MBean 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.

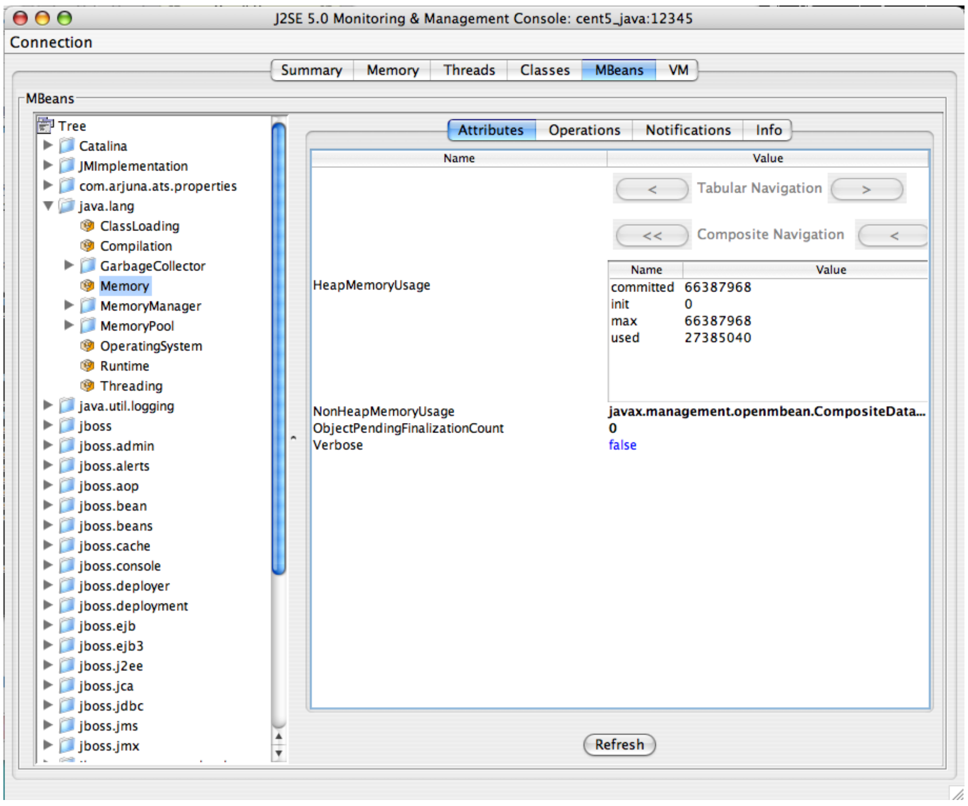
Figure 27.2. Memory MBean



Name	Value
HeapMemoryUsage	javax.management.openmbean.CompositeDat...
NonHeapMemoryUsage	javax.management.openmbean.CompositeDat...
ObjectPendingFinalizationCount	0
Verbose	false

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. 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.

Figure 27.3. Memory MBean Expanded



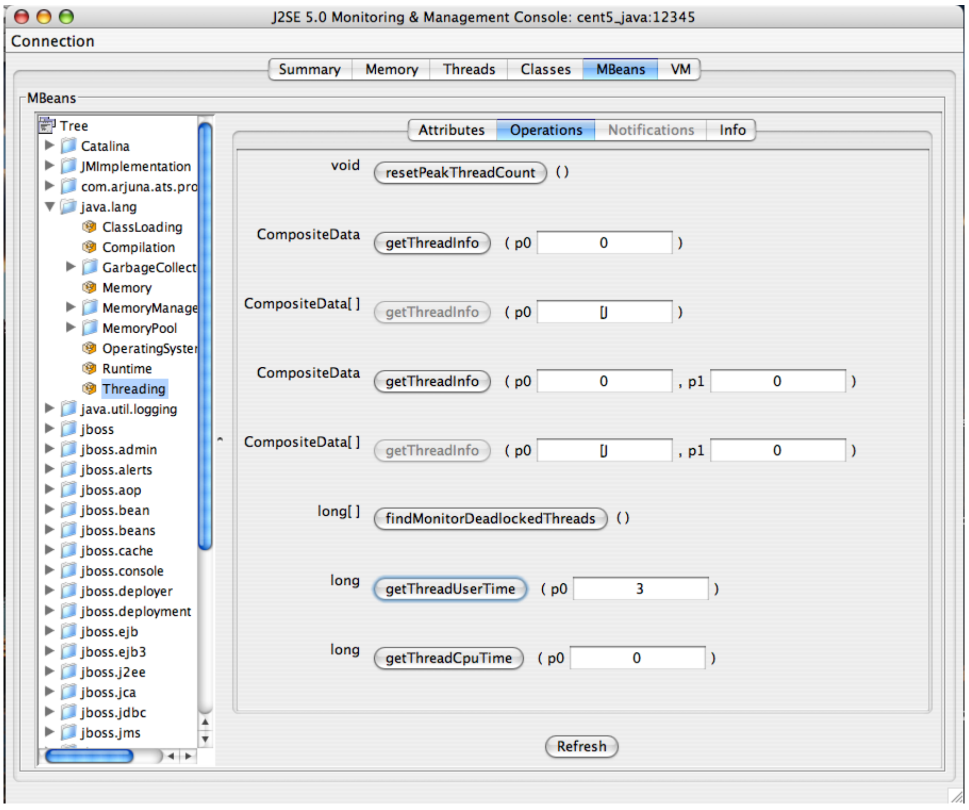
Name	Value
HeapMemoryUsage	
NonHeapMemoryUsage	javax.management.openmbean.CompositeData...
ObjectPendingFinalizationCount	0
Verbose	false

Name	Value
committed	66387968
init	0
max	66387968
used	27385040

The unique name of the MBean can be viewed by clicking on the Info tab. The first value is MBean Name and it's value in the case of Memory is: "java.lang:type=Memory". Note that there isn't a standardized way to name MBeans and application server vendors do it 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.

Figure 27.4. Operations Tab



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.

2.1.12. Installing ZenJMX

Step #1: Install the ZenJMX ZenPack

ZenJMX is installed using the "zenpack" command. As the zenoss user 6 run the following command:

```
${ZENHOME}/bin/zenpack --install /path/to/ZenJMX-1.0.0-el5-i386.zip
```

Step #2: Install Sun's JRE

You will need Java SE Version 5.0 or higher. 1.4.2 will not work, nor will gcc-java. Make sure that after you install Sun's JRE you update your PATH such that the "java" executable works. You can test this using the command "which java" - if it returns a fully qualified path to java you have successfully installed Java.

2.1.12.1. Installing ZenJMX on an appliance

ZenJMX and Sun's JRE is installed using a conary command. As root, run the following command:

```
conary update --resolve group-zenjmx
```

2.2. ApacheMonitor ZenPack

ApacheMonitor provides a method for pulling performance metrics from the Apache HTTP Server (http://httpd.apache.org/) directly into Zenoss without requiring the use of an agent. This is accomplished by utilizing the standard mod_status module that comes with version 1 and 2 of the HTTP server.

The following metrics will be collected and graphed for the Apache HTTP Server.

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

Follow these steps to setup your HTTP server so that it will allow Zenoss to access the server status.

1. On the Apache server, find your httpd.conf file. This is normally located in /etc/httpd/httpd.conf or /etc/httpd/conf/httpd.conf. Other locations are possible depending on your operating system and setup.

2. Turn the ExtendedStatus option on in the httpd.conf file. This option will typically be commented out. You can enable it by uncommenting it.

   ```
   #ExtendedStatus on
   ```

   ... becomes ...

   ```
   ExtendedStatus on
   ```

3. 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.yourdomain.com
   </Location>
   ```

4. Save the httpd.conf file with these changes then restart httpd. This can be accomplished with following command.

   ```
   apachectl restart
   ```

Once your Apache HTTP Server is configured to allow Zenoss to access the extended status, you can add Apache monitoring to the device within Zenoss by simply binding the Apache template to the device.

1. Navigate to the device in the Zenoss web interface.
2. Click the device menu, choose More then Templates.
3. Click the templates menu, choose Bind Templates.
4. Ctrl-click the Apache template from /Devices/Server to choose it.
5. Click OK.

You will now be collecting the Apache HTTP Server metrics from this device.

2.3. DellMonitor ZenPack

DellMonitor provides custom modeling of devices running the Dell OpenManage agents. It also contains hardware identification for Dell proprietary hardware. The information is collected through the SNMP interface.

The following information is modeled:

- Hardware Model 11
- Hardware Serial Number
- Operating System
- CPU Information (socket, speed, cache, voltage)
- PCI Card Information (manufacturer, model)

2.4. HPMonitor ZenPack

HPMonitor provides custom modeling of devices running the HP/Compaq Insight Management Agents. It also contains hardware identification for HP proprietary hardware. The information is collected through the SNMP interface.

The following information is modeled.

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

2.5. MySqlMonitor ZenPack

MySqlMonitor provides a method for pulling performance metrics from the MySQL database server (http://www.mysql.com/) directly into Zenoss without requiring the use of an agent. This is accomplished by utilizing the MySQL client library to connect to the database remotely.

The following metrics will be collected and graphed for MySQL server.

- Command Statistics (SELECT, INSERT, UPDATE, DELETE)
- Select Statistics (Scan, Range Check, Range Join, Full Join)
- Handler Statistics (Keyed & Unkeyed Reads, Writes, Updates, Deletes)
- Network Traffic (Received & Sent)

Follow these steps to setup your MySQL server to allow Zenoss to read performance data from the system tables.

1. Connect to the MySQL database using the MySQL client.

   ```
   mysql -u root
   ```

   ... or if there is a MySQL root password ...

   ```
   mysql -u root -p
   ```

2. Create a user for Zenoss to use. The username "zenoss" is recommended.

```
mysql> CREATE USER zenoss IDENTIFIED BY 'zenossPassword';

Query OK, 0 rows affected (0.00 sec)
```

3. Edit the zMySqlRootPassword zProperty for the device(s) within Zenoss that you intend to monitor MySQL on.

4. Bind the MySQL template to the same device(s).

Pay particular attention to the MySQL Version 5+ setting in the datasource. If you are monitoring pre-v5 installations of MySQL then be sure to set this value to False. If you are monitoring both pre v5 and v5+ installations then create two templates, one for MySQL installations earlier than v5 and another for those after.

2.6. NtpMonitor ZenPack

ZenPacks.zenoss.NtpMonitor monitors the offset between system time and a target ntp server's (Network Time Server) time.

2.6.1. Components

2.6.1.1. Event Classes

- /Events/Status/Ntp

2.6.1.2. Performance Templates

- NtpMonitor Datasource

2.6.1.3. Classes

- NtpMonitorDataSource

2.6.1.4. Datapoints

- offset Graphs: offset Skins: editNtpMonitorDataSource.pt

2.6.2. DataSource Class Options

The following Datasource options are available with this ZenPack:

- port - Port number to send ntp request (default: 123)

  NOTE: The check_ntp plugin does not seems to use this option even though its usage notes that it exists.

- warning - Response time to result in warning status (seconds)

- critical - Response time to result in critical status (seconds)

- timeout - Seconds before connection times out (default: 10)

2.6.3. Example

The NTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.NNTPMonitor EGG.

2. Navigate to the device (or Device Class) that has a NTP Server you want to monitor (add the device if necessary).

3. Choose the Templates options from the Server menu and the Bind Template option and bind the NTPServer template to the device.

   The next cycle of zencommand will collect offset data.

4. If a SSL or a custom port is being used, navigate to the NTPMonitor template and change the Use SSL and Port options as needed.

2.7. LDAPMonitor ZenPack

The ZenPacks.zenoss.LDAPMonitor monitors the response time of an LDAP server in milliseconds.

2.7.1. Components

2.7.1.1. Performance Templates

- LDAPServer

2.7.1.2. Thresholds

- SlowLDAP - 5000ms threshold for Warning event on response time; escalated after 6

- BrokenLDAP - 30000ms threshold for Critical event on response time

2.7.1.3. DataSource

- ldap using LDAPMonitor; creates time DataPoint

2.7.1.4. Graph Definitions

- LDAP Server Response Time

2.7.1.5. zProperties

- zLDAPBaseDN - The Base Distinquished Name to use for the response time check. Defaults to dc=zenoss,dc=com. Should be customized to fit the customer's LDAP server requirements.

- zLDAPBindDN - The Distinquished Name to use for binding to the LDAP server, if authentication is required. Defaults to nothing.

- zLDAPBindPassword - The password to use for binding to the LDAP server, if authentication is required. Defaults to nothing.

2.7.2. DataSource Class Options

- LDAP Server - defaults to ${dev/id}

- Base Distinguished Name - defaults to ${here/zLDAPBaseDN}

- Bind Password - defaults to ${here/zLDAPBindPassword}

- Use SSL - defaults to unchecked

- Port - defaults to 389

2.7.3. Additional Configuration

The three zProperties (zLDAPBaseDN, zLDAPBindDN and zLDAPBindPassword) should be configured to allow a valid connection to the LDAP server. The zLDAPBaseDN should be configured to issue a valid search to the LDAP server; normally this property would be set to the root organization within the server. If authentication is required on the server then zLDAPBaseDN and zLDAPBasePassword must also be configured.

2.7.4. Example

The NNTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.LDAPMonitor EGG.

2. Navigate to the device (or Device Class) that has a LDAP Server you want to monitor (add the device if necessary).

3. Choose the Templates options from the Server menu and the Bind Template option and bind the LDAPServer template to the device.

   The next cycle of zencommand will collect offset data.

4. Choose the zProperties option from the Server menu.

5. Change the zLDAPBaseDN zProperty to be the appropriate base distinguished name for your LDAP server. Typically this will be the organization's domain name, e.g. dc=foobar,dc=com. Check with your LDAP Administrator for more options.

6. Change the zLDAPBindDN and zLDAPBindPassword if authentication is required for the LDAP server. For example, cn=joe,dc=foobar,dc=com. Check with your LDAP Administrator for the format in your organization.

7. If your LDAP Servers require SSL or a custom port then navigate to the LDAP Server template, choose the ldap DataSource and then change the Use SSL and Port fields as needed.

8. Validate your configuration by running zencommand and observing that the check_ldap or check_ldaps command correctly connects to your LDAP server:

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

2.8. RPCMonitor ZenPack

ZenPacks.zenoss.RPCMonitor monitors the availability of an RPC server.

2.8.1. Components

2.8.1.1. Performance Templates

- RPCServer

2.8.1.2. DataSource

- rpc using RPCMonitor

2.8.2. DataSource Class Options

- RPC Server - defaults to ${dev/id}

- RPC Command - defaults to ${here/zRPCCommand}
- Port - defaults to 0 (to use PortMapper)

2.8.3. Additional Configuration

The zProperty zRPCCommand must be configured to indicate which RPC command should be tested for availability.

2.8.4. Example

The RPCMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.RPCMonitor EGG.
2. Navigate to the device that has a RPC Server that needs to be monitored (add the device if necessary).
3. Choose the Templates options from the Server menu and the the Bind Template option and bind the RPCServer template to the device.

   The next cycle of zencommand will collect offset data.
4. Choose zProperties from the Server menu and set the appropriate RPC command to test in the zRPCCommand zProperty, e.g. nfs, ypserv, etc.
5. If a custom port is being used, navigate to the RPCMonitor template and change the Port option as needed.
6. Validate your configuration by running zencommand and observing that the check_rpc command correctly connects to your RPC server:

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

2.9. IRCMonitor ZenPack

The ZenPacks.zenoss.IrcdMonitor monitors the number of users connected to an IRC server.

2.9.1. Components

2.9.1.1. Event Classes

- IRCD

2.9.1.2. Performance Templates

- IRCDMonitor

2.9.1.3. DataSource Classes

- IRCDMonitorDataSource

2.9.1.4. Graph Definitions

- number

2.9.1.5. Skins

- editIRCDMonitorDataSource.pt

2.9.2. DataSource Class Options

- hostname - Name or IP address of remote server to connect to.
- port - Port number to connect to remote IRC server (default: 6667)
- warning_num - Number of connections to
- critical_num - Response time to result in critical status (seconds)

2.9.3. Example

The IrcdMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.IrcdMonitor EGG.
2. Navigate to the device (or Device Class) that has a IRC Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the IRCDMonitor template to the device.

   The next cycle of zencommand will collect offset data.

2.10. JabberMonitor ZenPack

ZenPacks.zenoss.JabberMonitor monitors the response time of devices running a Jabber Server.

2.10.1. Components

2.10.1.1. Event Classes

- Jabber

2.10.1.2. Performance Templates

- JabberMonitor

2.10.1.3. DataSources

- JabberMonitorDataSource

2.10.1.4. DataPoints

- time

2.10.1.5. Graph Definitions

- time

2.10.2. DataSource Class Options

- Timeout (seconds) - Seconds before connection times out (default: 60)

- Port - port number default 5223

- Send String - string to send to the server : default <stream:stream to='${dev/id}' xmlns:stream='http://etherx.jabber.org/streams'>

- Expect String - string to expect in server response : default <stream

2.10.3. Example

The JabberMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.JabberMonitor EGG.

2. Navigate to the device (or Device Class) that has a Jabber Server you want to monitor (add the device if necessary).

3. Choose the Templates options from the Server menu and the the Bind Template option and bind the Jabber template to the device.

   The next cycle of zencommand will collect offset data.

4. If a SSL or a custom port is being used, navigate to the JabberMonitor template and change the Use SSL and Port options as needed.

2.11. DigMonitor ZenPack

DigMonitor monitors the response time of DNS lookups for devices running an Dig Server.

2.11.1. Components

2.11.1.1. Event Classes

- /Events/Status/DNS

2.11.1.2. Performance Templates

- NameServer

2.11.1.3. Datasource Classes

- DigMonitorDataSource

2.11.1.4. Datapoints

- time

2.11.1.5. Graph Definitions

- DNS Response Time

2.11.1.6. Skin Templates

- editDigMonitorDataSource

2.11.2. DataSource Class Options

- DNS Server: the nameserver against which the dig command should be run

- Port: The port on which the nameserver is listening
- Record Name: The name of the record you wish to look up
- Record Type: The DNS record type (e.g. A, MX, CNAME)

2.11.3. Example

The DigMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.DigMonitor EGG.
2. Navigate to the device (or Device Class) that has a Dig Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the DigMonitor template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the DigMonitor template and change the Use SSL and Port options as needed.

2.12. FTPMonitor ZenPack

The ZenPacks.zenoss.FTPMonitor ZenPack monitors connection response time to an FTP server.

2.12.1. Components

2.12.1.1. Event Classes

- Ftp

2.12.1.2. Performance Templates

- FtpMonitor

2.12.1.3. DataSource

- FtpMonitor

2.12.1.4. Datapoint Thresholds

- time

2.12.1.5. Graph Definitions

- time

2.12.2. Datasource Class Options

- Timeout - Seconds before connection times out (default: 60)
- Port - port to connect to FTP server (default 21)
- Send String - string to send to server

- Expect String - string to expect in server response
- Quit String - String to send server to initiate a clean close of the connection
- Refuse - Accept tcp refusals with states ok, warn, crit (default: crit)
- Mismatch - Accept expected string mismatches with states ok, warn, crit (default: warn)
- Max Bytes - Close connection once more than this number of bytes are received
- Delay - Seconds to wait between sending string and polling for response
- Certificate (minimum days for which a certificate is valid)
- Use SSL - Use SSL for the connection.
- Warning response time (seconds) - Response time to result in warning status (seconds)
- Critical response time (seconds) - Response time to result in critical status (seconds) What metrics (datapoints) does it collect? response time of the ftp connection

2.12.3. Example

The FTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.FTPMonitor EGG.
2. Navigate to the device (or Device Class) that has a FTP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the the Bind Template option and bind the Jabber template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the FTPMonitor template and change the Use SSL and Port options as needed.

2.13. NNTPMonitor ZenPack

The ZenPacks.zenoss.NNTPMonitor monitors the response time of an NNTP server in milliseconds.

2.13.1. Components

2.13.1.1. Performance Templates

- NNTPServer

2.13.1.2. DataSource

- nntp using NNTPMonitor; creates time

2.13.1.3. DataPoint Thresholds

- SlowNNTP - 5000ms threshold for Warning event on response time; escalated after 6 BrokenNNTP - 30000ms threshold for Critical event on response time

2.13.1.4. Graph Definitions

- NNTP Server Response Time

2.13.2. DataSource Class Options

- NNTP Server - defaults to ${dev/id}
- Port - defaults to 119

2.13.3. Example

The NNTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.NNTPMonitor EGG.
2. Navigate to the device (or Device Class) that has a NNTP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the NNTPServer template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the NNTPMonitor template and change the Use SSL and Port options as needed.
5. Validate your configuration by running zencommand and observing that the check_nntp or check_nntps command correctly connects to your LDAP server:

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

Chapter 28. Zenoss Global Dashboard (ZenGlobe) (Commercial)

1. About Zenoss Global Dashboard (ZenGlobe)

The Zenoss Global Dashboard is a Zenoss Enterprise feature manager of managers for displaying in one location the data from several Zenoss instances. Each Zenoss instance monitors a number of devices on various networks and then the ZGB combines all of that data and makes for easy viewing of the data. You can view the data based on collector, network, location and report on the device in combination of the groupings you create. This enhances the scalability of Zenoss as well as extending the performance and reporting capabilities.

ZenGlobe is a standalone web server.

2. Additional Requirements

The Zenoss Global Dashboard requires at least Python 2.3.

3. Installation

1. Download the latest version of the Zenoss Global Dashboard.

2. Extract the tarball and change to the created directory using the following commands:

   ```
   tar xzf zenglobe-1.0.tar.gz
   ```

   ```
   cd ZenGlobe
   ```

3. Prior to starting up the first time, ZenGlobe needs to know the port it should bind to and the Zenoss instance it should use for authentication. Run:

   ```
   sudo ./zenglobe configure
   ```

4. Enter the port to which you want ZenGlobe to bind. Make sure you have nothing else listening at that port.

When asked, enter the hostname of a running Zenoss instance that you want ZenGlobe to use for authentication. You can change this setting later, but in order to log in to ZenGlobe the first time you will need to use the username and password of a user from this Zenoss instance. Anyone with a login to this Zenoss instance will be able to view the ZenGlobe dashboard, but only the “admin” user will be able to edit settings.

1. Start ZenGlobe using the command:

   ```
   sudo ./zenglobe start
   ```

2. Check to make sure that ZenGlobe has started by accessing from your browser:

   ```
   http://[ZenGlobe machine hostname]:[port]/
   ```

   The ZenGlobe login screen will appear. If you do not get the ZenGlobe login screen retrace your steps and try again.

3.1. Setting Up Users on Remote Zenoss Instances

For security reasons, in order to gather data from remote Zenoss instances, ZenGlobe must have a login to those instances. By default, this is set to be zenglobe:zenglobe; however, it is a good idea to reconfigure ZenGlobe to use a more secure username and password combination.

You will also need to create a user on each Zenoss instance to be monitored using the same username and password.

1. In a browser, navigate to the Zenoss instance you wish to monitor, click “Settings” in the left nav pane, then select the “Users” tab..
2. From the table menu, select “Add New User.” Enter the username that you want ZenGlobe to use to log in to all Zenoss instances (e.g., “zenglobe”). You may leave the “Email” field blank. Click “OK.”
3. Repeat these steps for each Zenoss instance to be monitored, then configure ZenGlobe to log in as the remote users you have just created.

4. Setup and Configuration

1. Log in to ZenGlobe as the admin user.

   (Only the admin user can modify ZenGlobe options.)

2. Click the "Configure..." link in the top bar. The configuration box will slide down.

   The options in this Configuration box are as follows:

   - Zenoss Servers - The list of hostnames of the Zenoss instances ZenGlobe will monitor.
   - Remote Login - The username and password ZenGlobe will use to access the remote Zenoss instances. By default, it is set to zenglobe:zenglobe. Follow the instructions in 4.1.1 to set up matching users on each Zenoss instance to be monitored.
   - URL Template - The template ZenGlobe will use to build the URL by which it accesses monitored Zenoss instances. If you run your Zenoss instances on a different port, or serve them behind Apache with rewritten URLs, you will need to update this value to reflect that change.

3. You may also reset the port and authentication server using the same command line option you used when initially configuring ZenGlobe:

   ```
   sudo ./zenglobe configure
   ```

5. Using the Zenoss Global Dashboard

The global dashboard collects event and heartbeat data from the monitored Zenoss servers and aggregates them into a single view.

- Device Issues - A list of all devices with serious events. The "Server" column displays the Zenoss server that monitors that device.
- Zenoss Sub-Systems - A list of monitored Zenoss instances. An event rainbow is displayed for each instance, showing a summary of active events.
- Zenoss Issues - A list of heartbeat issues from monitored Zenoss instances. Refer to the Zenoss admin guide for instructions on how to handle these.

6. Viewing a Zenoss Instance from the Zenoss Global Dashboard

The drop-down list on the extreme left of the top bar can be used to view monitored Zenoss instances from within ZenGlobe. Simply select the hostname of an instance from the list to load it below. For security's sake, you will need to log in to the remote instance. You may return to the ZenGlobe dashboard at any time by selecting it from the same drop-down list.

7. Logging out of ZenGlobe

To logout of the Zenoss Global Dashboard, just click "Logout" in the top bar to end your ZenGlobe session.

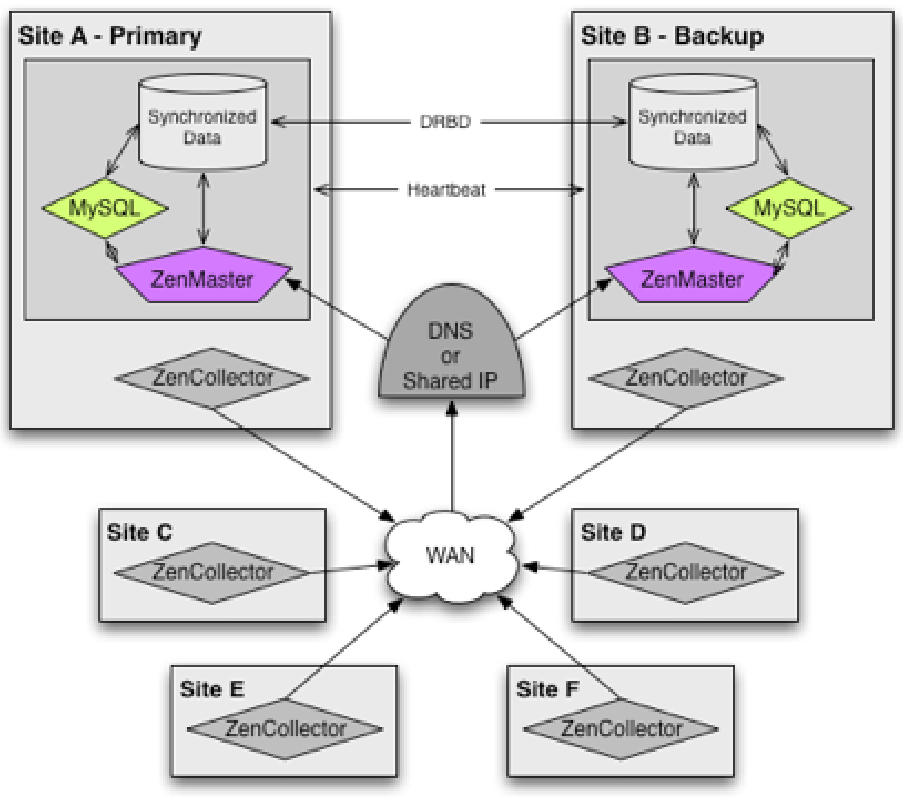
Chapter 29. Zenoss High Availability

1. Overview

This chapter will describe the method for setting up two Zenoss master servers that operate in an active/passive configuration to provide a single, highly-available master. In addition, a collector will be running on each master to monitor the local network. The solution relies on DRBD (Distributed Replicated Block Device), which is a distributed storage system for the Linux platform. It consists of a kernel module, several userspace management applications and some shell scripts. DRBD bears similarities to RAID 1, except that it runs over a network.

The full deployment can conceptualized according to the following diagram. The number of non-master sites is not limited. The collectors can be configured to choose the current active master either by a dynamically router shared IP (BGP, etc.) or through a DNS lookup of a record that must be changed manually when the master site is failed over.

Figure 29.1. Zenoss High Availability



2. Conventions

The following conventions will be used to reference different elements required during the installation and configuration.

- zenoss-master: Hostname of the current Zenoss master server or IP.
- zenoss-siteA: Hostname of the Zenoss server in site A.
- 10.0.1.20: IP address of the Zenoss server in site A.

- zenoss-siteB: Hostname of the Zenoss server in site B.
- 10.0.2.20: IP address of the Zenoss server in site B.
- cluster: Highly available combination of zenoss-siteA and zenoss-siteB.
- /dev/sdb1: Block-level device that will be replicated between site A and B.
- /r0: File system mount point for the /dev/sdb1 block device.

3. Prerequisites

Prior to beginning your installation and configuration you should make sure that the following prerequisites are met.

- Architecture of both master systems is identical (i686 or x86_64)
- Operating system of both master systems is identical (RHEL5 or CentOS5)
- A spare partition that has been sized large enough to hold the MySQL and Zope database exists in an unpartitioned and unmounted state on both masters.
- A standard installation of Zenoss Enterprise has already been performed on both of the masters. This should be a clean installation that has not been further configured or used.

4. Installation of Packages

The following RPM packages must be installed to support the disk replicate and failover mechanism. These are available as part of the standard repositories for RHEL5 and CentOS5.

- drbd82
- kmod-drbd82
- heartbeat

5. Stopping Services

Stop the MySQL and Zenoss services with the following commands on zenoss-siteA and zenoss-siteB.

```
service zenoss stop
service mysqld stop
```

6. Setting up DRBD for File System Replication

We will now use those extra partitions we setup to replicate the data that needs to be shared.

1. Setup /etc/drbd.conf with the following contents on zenoss-siteA and zenoss-siteB.

   ```
   global {
     usage-count yes;
   }
   ```

```
common {
  protocol C;
}
resource r0 {
  syncer {
    rate 500K;
  }
  on zenoss-siteA {
    device    /dev/drbd0;
    disk      /dev/sdb1;
    address   10.0.1.20:7788;
    meta-disk internal;
  }
  on zenoss-siteB {
    device    /dev/drbd0;
    disk      /dev/sdb1;
    address   10.0.2.20:7788;
    meta-disk internal;
  }
}
```

2. Create the DRBD meta-data on zenoss-siteA and zenoss-siteB.

   ```
   drbdadm create-md r0
   ```

3. Start the drbd service on zenoss-siteA and zenoss-siteB.

   ```
   service drbd start
   ```

4. Seed the replicated partition on zenoss-siteA only.

   ```
   drbdadm -- --overwrite-data-of-peer primary r0
   ```

5. Format the replicated disk on zenoss-siteA only.

   ```
   mkfs.ext3 /dev/drbd0
   ```

7. Relocating Shared Data to the Replicated Disk

There is very little data we need replicated, but whatever we do want to be replicated should be moved/linked to the /r0 directory that will be replicated by DRBD.

1. Mount the replicate disk on zenoss-siteA.

   ```
   mkdir /r0
   mount /dev/drbd0 /r0
   mkdir /r0/zenoss
   chown zenoss:zenoss /r0/zenoss
   ```

2. Create initial MySQL data on zenoss-siteA.

   ```
   service mysqld start
   service mysqld stop
   ```

3. Move MySQL data to /r0 on zenoss-siteA.

   ```
   mv /var/lib/mysql /r0/mysql
   chown –R mysql:mysql /r0/mysql
   ```

4. Configure MySQL to find its data on /r0 on zenoss-siteA and zenoss-siteB.

```
ln –s /r0/mysql /var/lib/
```

5. Configure Zenoss to find its data on /r0 on zenoss-siteA and zenoss-siteB.

   In /opt/zenoss/etc/zeo.conf find "$INSTANCE/var/Data.fs" and change it to "/r0/zenoss/Data.fs"

6. Create initial Zenoss data on zenoss-siteA.

```
service zenoss start
service zenoss stop
```

8. Setting up Heartbeat for Resource Migration

Heartbeat is part of the Linux HA project and is responsible for performing the real work of a active/passive high-availability setup. It will mount /r0 on whichever node is currently the master as well as stopping and starting the appropriate servers such as MySQL and Zenoss.

1. Create /etc/ha.d/ha.cf on zenoss-siteA with the following contents.

```
keepalive 2
deadtime 30
warntime 10
initdead 120
bcast eth0
node zenoss-siteA
node zenoss-siteB
crm yes
```

2. Create /etc/ha.d/authkeys on zenoss-siteA with the following contents.

```
auth 1
1 sh1 yourSecretKey
```

3. Set restrictive permissions on the authkeys file on zenoss-siteA.

```
chmod 0600 /etc/ha.d/authkeys
```

4. Copy the ha.cf and authkeys files from zenoss-siteA to zenoss-siteB.

5. Start the heartbeat service on zenoss-siteA and zenoss-siteB.

```
service heartbeat start
```

9. Setting up Heartbeat Resources

Now that we have a replicated disk and a cluster setup we need to tell heartbeat which resources we want it to control. This is all done on zenoss-siteA exclusively. The heartbeat service will replicate the changes automatically.

1. Setup constraints to make sure our replicated disk is always in place before the services that require it are started. This is done by creating a file called constraints.xml (location is irrelevant) with the following contents.

```
<constraints>
  <!-- /r0 can't be mounted until drbd is promoted -->
  <rsc_order id="r0_after_drbd"
             from="rg_zenoss" action="start"
             to="ms_zenoss" to_action="promote"
             type="after"/>
  <!-- /r0 and drbd always have to be colocated -->
  <rsc_colocation id="r0_on_drbd" to="ms_zenoss"
                  to_role="master" from="rg_zenoss"
                  score="INFINITY"/>
</constraints>
```

2. Load this file into heartbeat which will replicate it across the cluster.

```
cibadmin –U –x constraints.xml
```

3. Create a resources.xml file with the following content to setup all resources required for a Zenoss cluster.

```
<resources>
  <master_slave id="ms_zenoss">
    <meta_attributes>
      <attributes>
        <nvpair name="notify" value="yes"/>
      </attributes>
    </meta_attributes>

    <primitive class="ocf" type="drbd" provider="heartbeat"
               id="drbd_shared">
      <instance_attributes>
        <attributes>
          <nvpair name="drbd_resource" value="r0"/>
        </attributes>
      </instance_attributes>

      <operations>
        <op name="monitor" interval="29s" timeout="10s"
            role="Master"/>
        <op name="monitor" interval="30s" timeout="10s"
            role="Slave"/>
      </operations>
    </primitive>
  </master_slave>

  <group id="rg_zenoss">

    <!-- Mount /r0 -->
    <primitive class="ocf" type="Filesystem"
               provider="heartbeat" id="fs_shared">

      <instance_attributes>
        <attributes>
          <nvpair name="device" value="/dev/drbd0"/>
          <nvpair name="directory" value="/r0"/>
          <nvpair name="type" value="ext3"/>
        </attributes>
      </instance_attributes>
    </primitive>

    <!-- Take 192.168.1.229 IP -->
    <primitive class="ocf" type="IPaddr2"
               provider="heartbeat" id="ip_shared">
      <instance_attributes>
        <attributes>
          <nvpair name="ip" value="192.168.1.229"/>
          <nvpair name="nic" value="eth0"/>
        </attributes>
      </instance_attributes>
    </primitive>

    <!-- MySQL Service -->
    <primitive class="lsb" type="mysqld"
               provider="heartbeat" id="mysqld"/>
    <!-- Zenoss Service -->
    <primitive class="lsb" type="zenoss"
               provider="heartbeat" id="zenoss">
      <operations>
        <op id="zenoss_start" name="start" timeout="1m"/>
        <op id="zenoss_stop" name="stop" timeout="1m"/>
        <op id="zenoss_status" name="monitor"
            timeout="1m"/>
      </operations>
    </primitive>

  </group>
</resources>
```

4. Load this file into the cluster.

```
cibadmin –U –x resources.xml
```

10. Troubleshooting

The following are some commands that will greatly help you understand, operate and troubleshoot your cluster.

- crm_mon will show you the current state of all cluster resources and nodes.
- cat /proc/drbd will show you the current state of your replicated disk.
- crm_resource –r rg_zenoss –M –H <node-hostname> will migrate the rg_zenoss resource group to the specified node.
  - You should always run crm_resource –r rg_zenoss –U to unmigrate a resource afterwards or the resource will never be able to fail back to the node you migrated away from.

11. References

Much more in-depth information on Linux clustering can be found at the following sites.

- Linux HA (http://www.linux-ha.org/)
- DRBD Users Guide (http://www.drbd.org/users-guide/ch-configure.html)
- CentOS DRBD Howto (http://wiki.centos.org/HowTos/Ha-Drbd)

Chapter 30. Distributed Collector

1. What is Distributed Collector?

Distributed Collector allows you to deploy additional performance collection and event monitoring daemons to the Zenoss server or other servers. This allows you to:

- Distribute processor, disk, and network load between multiple servers
- Collect performance and events from networks that cannot be reached by the Zenoss server
- Configure more than one set of monitoring settings and assign devices to the configuration that makes most sense

When you first install Distributed Collector, Zenoss is configured with one hub and one collector. A collector is a set of collection daemons, on the Zenoss server or another server, that share a common configuration. That configuration contains values such as number of seconds between SNMP collection cycles, default discovery networks, and maximum number of zenprocess parallel jobs.

Each collector has its own copy of each of the Zenoss collection daemons. For example, Zenoss initially contains collection daemons named zenperfsnmp, zenprocess, and zenping. If you create a new collector named My2ndCollector, then the system creates new daemons named My2ndCollector_zenperfsnmp, My2ndCollector_zenprocess, and My2ndCollector_zenping. This collector could run on the main Zenoss server or on a remote server.

You cannot delete the initial hub and collector setup by Distributed Collector (both named localhost).

In addition to collectors, Distributed Collector allows you to set up new hubs. A hub represents an instance of the zenhub daemon, which is the daemon through which all collector daemons communicate with the object database and event database. All collectors must belong to exactly one hub; however, a hub may have many collectors associated with it. All hubs (and indirectly all collectors) refer to the same object and event databases.

1.1. Restrictions and Requirements

Servers hosting remote hubs or collectors must be the same operating system and hardware architecture as the Zenoss server. For example, if the Zenoss server if running RedHat Enterprise Linux v5 on Intel 32-bit hardware, then hubs and collectors can be deployed only to other RHEL 5 32-bit servers. If your Zenoss server is a Zenoss virtual appliance or hardware appliance, then the remote server must also be a Zenoss appliance.

For each distributed collector you add, you must have an additional 500 MB of system memory.

For additional platform-specific information, refer to Platform Notes.

1.2. Navigating Existing Collectors and Hubs

When you log in as the Zenoss admin user, the Navigation pane displays a link titled Collectors in the Management area. Click this link to go to the Collectors page, which lists existing hubs and collectors in hierarchical form. Hubs are listed at the top level; collectors are nested below the hub to which they belong.

From this page, you can:

- Add a hub
- Delete a hub (which also deletes its associated collectors)
- View and edit hub settings

The Daemons tab lists the copy of the zenhub daemon that belongs to the collector. Links adjacent to the daemon name allow you to view its log, and view and edit its configuration. Use the buttons to the right of the daemon name to stop, start, and restart the daemon.

2. Typical Usage Scenarios for Distributed Monitoring

Here are a few typical setup scenarios for utilizing multiple hubs and collectors:

- ZeoDB - local hub - local collector
- ZeoDB - local hub - multiple local collectors
- ZeoDB - local hub - remote collector
- ZeoDB - local hub - multiple remote collectors
- ZeoDB - remote hub - remote collector
- ZeoDB - multiple remote hubs - multiple remote collectors
- ZeoDB - local hub - local collector(s) - remote hub(s) - remote collectors

Each of these scenarios can address the following big-picture business cases to help make your monitoring more effective, efficient and balanced.

2.1. Scalability on the Local Device

You can use distributed monitoring to have multiple local collectors that will utilize multiple CPUs. Since each collector is one thread, the use multiple collectors allows the device to use its own load balancing to properly optimize multiple CPUs. If you were to only have one hub and one collector locally, the load would be concentrated on a small portion of the potential power.

2.2. Scalability Over Multiple Remote Devices

Scaling on multiple remote devices basically enhances the same idea of single threads as with a single local device, but spreads the collection and configuration burden across multiple machines to maximize efficiency.

2.3. Collectors and or Hubs Set Up through a Firewall

If you want to have collectors and hubs on different sides of a firewall, use the ZeoDB hostname and port numbers to manage the flow of data through your firewall.

2.4. Collection Over Multiple Overlapping IP Spaces

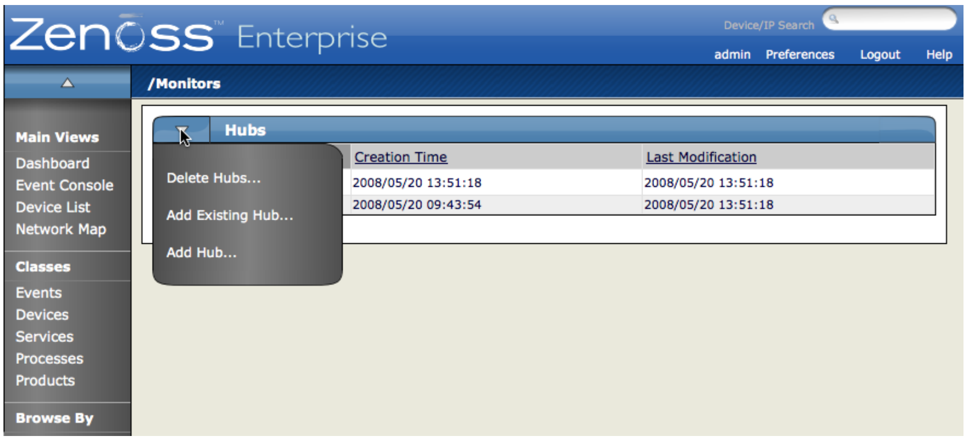
Distributed monitoring is also an essential component for enabling multiple realm IP support available to Zenoss Service Provider customers.

3. Deploying Hubs

Hubs are used to manage configuration data and pass it to the collectors. Hubs also take data from the collectors and pass it to the ZeoDB. More hubs can be a more efficient way to manage larger deployments, as they help distribute the computing resources when configuration changes are made. They further remove the potential for configuration changes to be a bottleneck to gathering and processing data.

To add a hub, from the main Collectors page, select Add Hub... from the table menu.

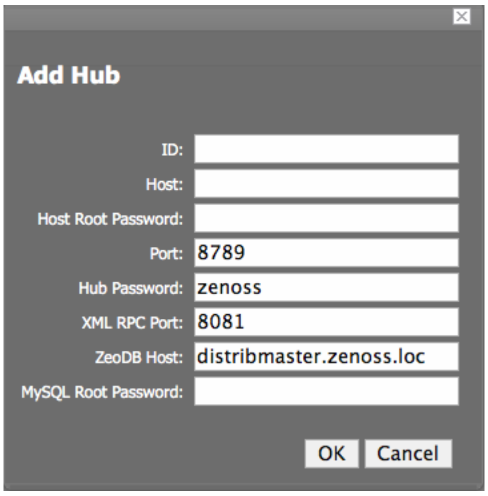
Figure 30.1. Hubs Table menu



Creation Time	Last Modification
2008/05/20 13:51:18	2008/05/20 13:51:18
2008/05/20 09:43:54	2008/05/20 13:51:18

The Add Hub dialog appears:

Figure 30.2. Add New Hub dialog



Add Hub

ID:	
Host:	
Host Root Password:	
Port:	8789
Hub Password:	zenoss
XML RPC Port:	8081
ZeoDB Host:	distribmaster.zenoss.loc
MySQL Root Password:	

Use the following fields to define the hub:

- ID - Enter a name for the new hub. The name can be any unique combination of letters, digits, and dashes.
- Host - Enter the fully qualified domain name, IP address, or resolvable hostname of the server on which the new hub will run.
- Host Root Password - Enter the root user password for the server you specified in the Host field.
- Port - Enter the port number on which the hub should listen for collectors. The default port is 8789.
- Hub Password - Enter the hub password that the collectors will use to log into this hub. The default password is "zenoss."
- XML RPC Port - Specify the port on which the hub should listen for xml-rpc requests from the collectors or other API clients.

- ZeoDB Host - Specify the server hosting the ZeoDB database (the object database). In most cases, this is the IP address or hostname of the main Zenoss server.

- MySQL Root Password - Enter the password for the root MySQL user. If the MySQL root user does not have a password, then leave this field blank.

Click OK. The system displays log output from the creation of the new hub. When fully configured (this may require several minutes), click the link at the bottom of the page to go to the overview page for the new hub.

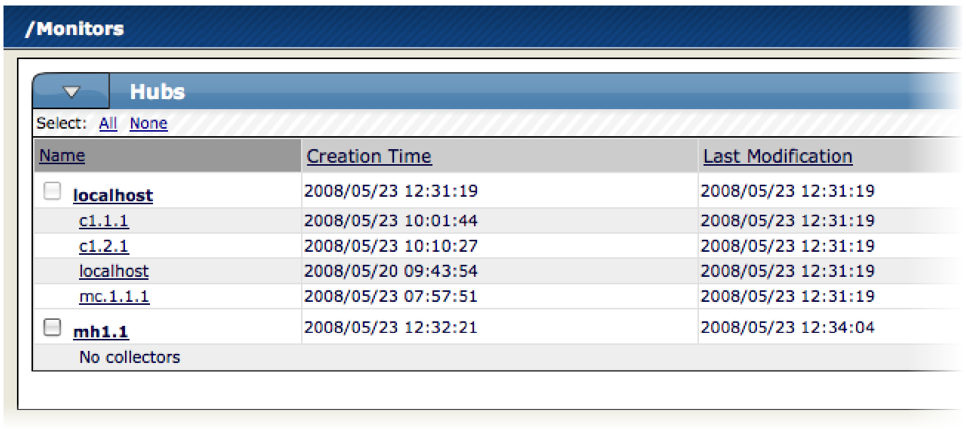
4. Deploying Collectors

To add new collector to a hub:

1. From the left navigation menu, select Collectors.

   The main Collectors page appears, showing all of the hubs and collectors.

   Figure 30.3. Main collectors page



/Monitors

Hubs

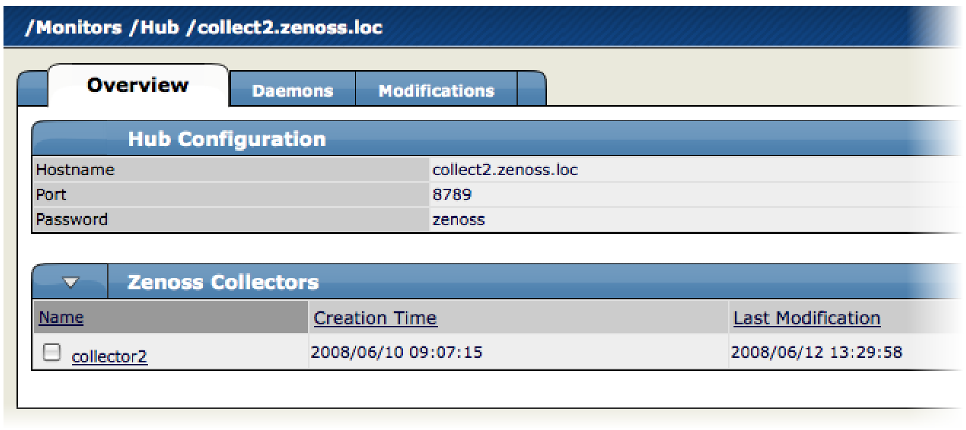
Select: All None

Name	Creation Time	Last Modification
localhost	2008/05/23 12:31:19	2008/05/23 12:31:19
c1.1.1	2008/05/23 10:01:44	2008/05/23 12:31:19
c1.2.1	2008/05/23 10:10:27	2008/05/23 12:31:19
localhost	2008/05/20 09:43:54	2008/05/23 12:31:19
mc.1.1.1	2008/05/23 07:57:51	2008/05/23 12:31:19
mh1.1	2008/05/23 12:32:21	2008/05/23 12:34:04
No collectors		

2. Click the name of the hub where you want to add the collector.

   The main page for this hub appears.

   Figure 30.4. Main hub page



/Monitors /Hub /collect2.zenoss.loc

Hub Configuration

Hostname	collect2.zenoss.loc
Port	8789
Password	zenoss

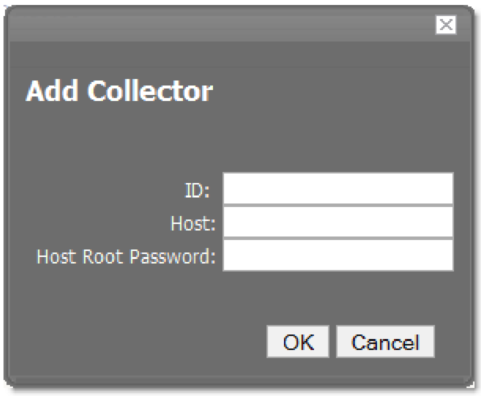
Zenoss Collectors

Name	Creation Time	Last Modification
collector2	2008/06/10 09:07:15	2008/06/12 13:29:58

3. From the Zenoss Collectors table menu, choose Add Collector...

The Add Collector dialog appears.

Figure 30.5. Add Collector Dialog



4. Add information to the dialog to define the collector.

   - ID - In the ID field, enter the name for the collector as it will be identified in Zenoss.

   - Host - Enter the name of the host for the collector. This must be a fully qualified domain name, IP address, or resolvable hostname. If you want the collector to run on the Zenoss server, then enter localhost.

   - Host Root Password - Enter the password for the root user on the Host.

5. Click OK. The system displays log output from the creation of the new collector. When fully configured (this may require several minutes), click the link at the bottom of the page to go to the overview page for the new collector.

4.1. Deleting Collectors

To delete a collector, click the name of the hub where the collector exists from the main collectors page. The Hub overview page appears. From the list of Zenoss Collectors, select the collector you want to delete. From the Zenoss Collectors table menu, select Delete Collector.

When you delete collectors using this Zenoss instance, they are not removed or "uninstalled" in anyway from the collector device. They continue to exist on the device until manually removed through the file system.

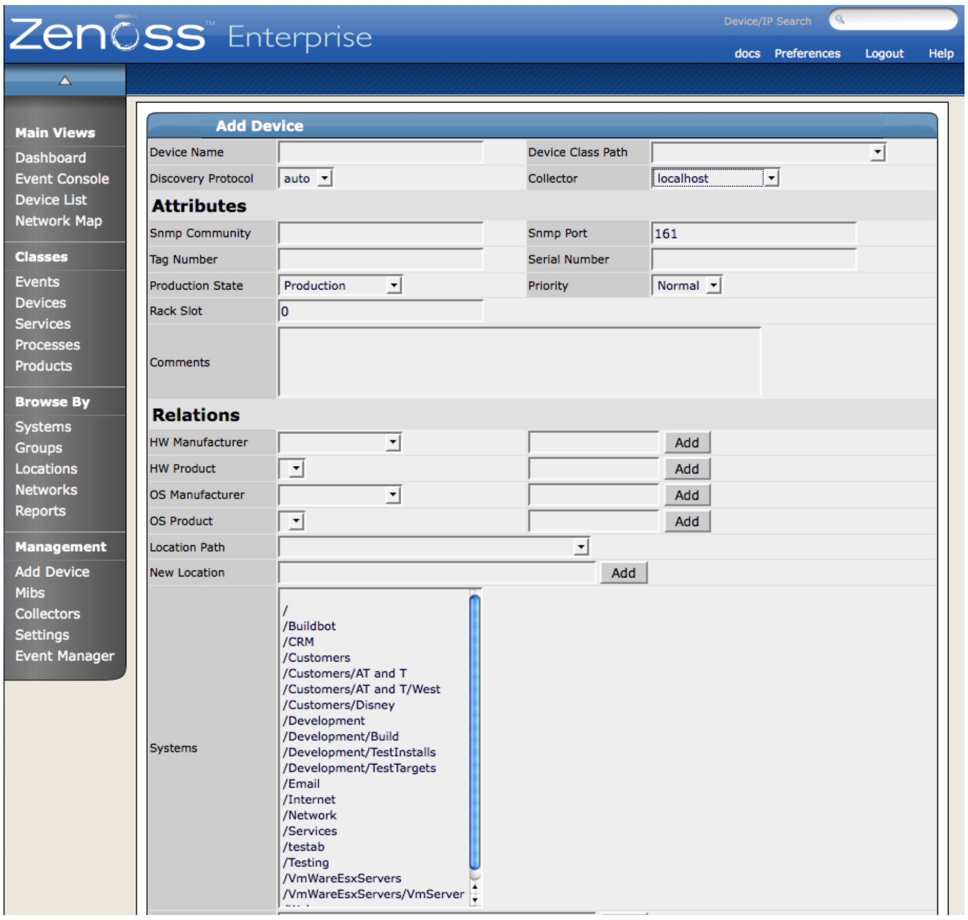
4.2. Updating a Hub or Collector

Any time you update your version of Zenoss or install additional ZenPacks, you should update any hubs or collectors. You do this by navigating to the Overview page for the hub or collector, and then choosing Update Hub or Update Collector from the page menu. This copies the most recent Zenoss code and ZenPacks to the server and restarts the daemons running there.

5. Adding Devices to Collectors

Adding devices to collectors occurs when you add the device to Zenoss. When you click Add Device from the left navigation menu you see the Add Device page.

Figure 30.6. Add Device Page



Add Device

Device Name		Device Class Path	
Discovery Protocol	auto	Collector	localhost

Attributes

Snmp Community		Snmp Port	161
Tag Number		Serial Number	
Production State	Production	Priority	Normal
Rack Slot	0		
Comments			

Relations

HW Manufacturer			Add
HW Product			Add
OS Manufacturer			Add
OS Product			Add
Location Path			
New Location		Add	

Systems

/
/Buildbot
/CRM
/Customers
/Customers/AT and T
/Customers/AT and T/West
/Customers/Disney
/Development
/Development/Build
/Development/TestInstalls
/Development/TestTargets
/Email
/Internet
/Network
/Services
/testab
/Testing
/VmWareEsxServers
/VmWareEsxServers/VmServer

In the top right of the resulting page, you can see the Collector drop-down menu. Choose the collector you want to use to collect the data for this device. The device then appears in the Device area for that collector. You can access this page by choosing the Collectors item from the left navigation menu and then choosing the collector from the resulting list. When you click on the name of the collector, the Overview page for the collector appears and then in the Devices area at the bottom of the page you can see the Device list for this collector.

5.1. Moving Devices Between Collectors

You can move devices from one collector to another using the Set Collector table menu item for any Device view. These Device views include:

- Device List
- Device Tree
- Any of the Organizers device list

The most basic way to do this (and also the most efficient for moving multiple devices from one collector to another):

1. From the left navigation menu, select Device List.

   The Device list appears.

2. Select all of the Devices you want to move to a particular collector by clicking the boxes next to all of the name(s) of the devices.

3. From the Device list table menu, select Set Collector.

   The Set Collector dialog appears.

4. Select the Collector from the drop-down list.

5. Click OK.

   The device or devices are moved to the selected collector.

5.2. Installation Notes

- Make sure the Zenoss server's hostname is a fully qualified domain name.
- Remember that collectors and hubs can be pushed only to servers with identical operating system versions and hardware architecture.
- Make sure that Event Manager > hostname has a fully qualified domain name (preferred) or at least a numeric address that can be reached by any server that has collectors or hubs deployed to it.
- Distributed Collector shuts down the iptables firewall on the Zenoss server. If you have any other firewalls on the Zenoss server, or on servers that host remote collectors or hubs, then you should disable their firewalls also.

5.3. Debugging

Hostname Configuration

The Zenoss server should have a properly configured hostname, preferably a fully qualified domain name. You can check the hostname from the shell:

root# hostname

You also can check by using the python function used by Zenoss:

root# python -c 'import socket; socket.gethostname()'

MySQL Host

Hubs on remote servers need access to the MySQL events database. This setting is the Hostname field in the Connection Information section of the Event Manager page. By default this is set to localhost, but will not work for remote hubs. Distributed Collector attempts to set this field to the FQDN of the Zenoss server when it is installed. If remote hubs appear to be having trouble connecting to MySQL or sending events, then check the value in this field to make sure it can be reached from the server the hub is on.

MySQL Access Privileges

Another aspect of remote hubs connecting to MySQL is privileges. For a hub to connect to the events database, the user specified in the User Name field in Event Settings must be granted privileges to connect to MySQL from the remote server. Distributed Collector attempts to grant these privileges any time a remote hub is created or updated. If a remote hub is logging error messages that indicate it is not allowed to connect to MySQL from the given host, then these privileges are likely not set up correctly. Granting of these privileges requires a fully qualified domain name for the remote server.

5.4. Firewall Notes

When a hub or collector is deployed DistributedCollector attempts to disable the iptables firewall on the Zenoss server. This enables the remote collector or hub to communicate with the daemons and databases running on the Zenoss server. You can re-enable iptables if you want to set up the proper rules to allow Zenoss traffic through.

Remote hubs need to communicate with the zeo database on the Zenoss server on port XXXX. Hubs also need to communicate with the MySQL server, usually on the Zenoss server (see Event Manager > Hostname), on the port specified in Event Manager > Port (usually 3306.) Collectors communicate with their hub on the port specified when the hub was created. See the ZenHub Port field on the hub's overview page.

5.5. Platform Notes

Software Appliance and Hardware Appliance

- Hubs and collectors can be deployed only to other Zenoss software or hardware appliances.
- You must stop Zenoss on an appliance before deploying a hub or collector to it.
- You must set a password for the root user on an appliance before deploying a hub or collector to it.
- Do not use conary to update appliances being used as remote collectors or hubs.

Chapter 31. Monitoring Virtual Host Machines (Commercial)

1. About Monitoring Virtual Host Monitoring

The ZenossVirtualHostMonitor Enterprise ZenPack allows you to monitor virtually hosted operating systems with Zenoss. This ZenPack refers to a Virtual Machine Host as the one running on the bare metal, and Guest for those running within the virtual hardware.

This ZenPack:

- Extends Devices to support a relationship from Host to Guest.
- Extends ZenModeler to find Guest OS's and add them to Virtual Hosts
- Provides screens and templates for collecting and displaying resources allocated to Guest OSs

2. Using Zenoss Virtual Host Monitor

To Use ZenossVirtualHostMonitor:

1. Make sure you have SNMP connectivity to your ESX 3.0 servers.
2. Make sure you have SSH keys to your Xen servers (as root).
3. Create your ESX 3.0 servers using the /Servers/Virtual Hosts/ESX device class. If you have these servers modeled already, remove them and recreate them under the ESX device class. DO NOT MOVE THEM.
4. Repeat step 3 with Xen hosts (using the Xen device class).
5. Select the Guest menu and ensure that the guest hosts were found when the devices were added.
6. Using the VMware Virtual Infrastructure Client, add Zenoss to the list of destinations for SNMP traps. See the Menus Administration-> VirtualCenter Management Server Configuration ->SNMP.

Optionally, you can install the VMware ESX MIB files in Zenoss.

Additional Notes:

- There is a handy link to the VMware web interface on each ESX server Status page.
- If the name of the Guest under ESX is the same as the name of a device being monitored directly by Zenoss, a link will be provided to take you directly from to that device from the Guest list.
- The CPU monitoring on VMware shows very little relationship to actual CPU used. We have asked VMware about the problem.

Chapter 32. Predictive Thresholding (ZenHoltWinters) (Commercial)

1. About Predictive Thresholding

The ZenHoltWinters Enterprise ZenPack adds the ability to fire threshold events when a device exceeds cyclical predicted values. Holt-Winters is the algorithm that we use for this prediction.

2. Using Predictive Thresholding

To use ZenHoltWinters, you need to add HoltWinters Thresholds against performance data. In any template, select add a new Threshold and select the HoltWintersFailurs threshold type.

Note: Zenoss relies on the existence of Holt-Winters RRAs within an RRD file. After adding Holt-Winters thresholds, the RRD files being monitored will need to be re-created so that the new configuration can occur. At the time of Zenoss Enterprise 2.1.1, there is no conversion utility. You will have to remove any existing RRD files so that new files can be created.

Important Note: Removing RRD files will remove any historical information associated with these RRD files.

3. Customizing the Thresholds

The Holt-Winters Threshold allows you to customize some aspects of the prediction engine. The 4 values to modify are:

- Rows: The number of points to use for predictive purposes.
- Alpha: A number from 0 to 1 that controls how quickly the model adapts to unexpected values.
- Beta: A number from 0 to 1 that controls how quickly the model adapts to changes in unexpected rates changes.
- Season: The number of primary data points in a season. Note that Rows must be at least as large as Season.

Chapter 33. Authentication Enterprise ZenPacks

1. Active Directory Authentication (Commercial)

1.1. About Active Directory Authentication

The ActiveDirectory is an Enterprise ZenPack allows you to authenticate users using Microsoft Active Directory.

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

1.2. Active Directory Monitored Metrics

The Active Directory ZenPack allows you to monitors the following metrics:

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

- LDAP Successful Binds
- LDAP UDP Operations/Sec
- LDAP Writes/Sec
- NTLM Authentications

2. LDAP Authentication in Zenoss (Commercial)

2.1. About LDAP Authentication

The LDAPAuthenticator Enterprise ZenPack allows Zenoss to use your existing LDAP authentication infrastructure (Active Directory, OpenLDAP and others) to provide SSO to the Zenoss web interface.

2.2. Setting Up LDAP Authentication

Follow these steps to setup the integration once you have installed the the LDAPAuthenticator ZenPack.

1. Go to:

   http://yourzenoss:8080/zport/acl_users
2. Choose your authentication plugin in the top-right and click Add.
   - Choose "ActiveDirectory Multi Plugin" for Microsoft Active Directory.
   - Choose "LDAP Multi Plugin" for any other LDAP server.
3. Complete the following form with your LDAP credentials and paths.
   - The ID can should be something like ldapAuthentication.
   - The Title can be left blank.
   - Default User Roles should be set to ZenUser.
4. Navigate back to /zport/acl_users using the breadcrumbs at the top.
5. Click on plugins in the list of objects.
6. Click on the Authentication Plugins link.
7. Move your ldapAuthentication plugin to the list of active plugins.

Chapter 34. Transaction Monitoring ZenPacks

1. Synthetic Transactions (Zenwebtx)(Commercial)

1.1. About Synthetic Transaction (Zenwebtx)

ZenWebTx is an Enterprise ZenPack that allows the user to test availability and performance of web sites. The user creates one or more tests which mimic a user's actions in a web browser. Zenoss then performs these tests periodically. Events are created in Zenoss whenever a test fails or a time threshold is exceeded. Additionally, Zenoss can record data regarding each test run, such as time taken for the test to execute, time for any portion of the test to complete and values extracted from web pages during the test.

1.2. Overview

Each test is represented by a single Data Source of type WebTx. ZenWebTx uses a scripting language called Twill to describe the steps of a test. These steps include actions such as clicking on a link or filling out form fields, assertions to check for the presence or absence of text on a page, and descriptions of what data to collect during the test. Users can write Twill commands manually or they can record a browser session which ZenWebTx then translates into Twill. These recordings can be made with a FireFox addon called TestGen4Web. A daemon named zenwebtx processes these Twill commands periodically, recording data and creating events as appropriate.

WebTx Data Sources can contain many Data Points depending on the Twill commands you specify:

- The total number of seconds required to execute the transaction. All WebTx Data Sources include this Data Point, named totalTime, by default.
- The success or failure of the transaction. All WebTx Data sources include this Data Point, named available, by default.
- The presence or absence of text patterns on any page. Additionally, numeric values that are part of these patterns can be extracted from the web page and recorded.

What's Included with ZenWebTx:

- New type of datasource, WebTx, which allows you to specify a web transaction
- New daemon, zenwebtx, that processes WebTx Data Sources
- New event classes: /Status/Web and /Perf/Web
- A sample performance template: ZenWebTx Example

1.3. Creating a WebTx Data Source

WebTx Data Sources are created in Performance Templates just like Data Sources that use SNMP or any other method to gather data within Zenoss. To create a new WebTx Data Source you use the “AddData Source...” menu item from the Data Sources table. A dialog will ask prompt you for some information. Enter the name of the new Data Source, select WebTx as the type and click the OK button. There are then two pages you use to specify the behavior of the new Data Source, the Data Source tab and the Script Tab.

1.4. Data Source Tab

After creating a new Data Source the next page you see is the Data Source tab. The fields on this page allow you to specify parameters about how and when this Data Source's web transaction will be performed as well as what data should be collected. The fields in the top part of the page are as follows:

- Name – This is the name of the Data Source that you initially specified in the Create Data Source dialog. This name along with the Data Point names (see section below on Data Points) are used in Thresholds and Graph Definitions to refer to the data collected by this Data Source.

- Source Type - This is always set to WebTx to indicate that this is a synthetic web transaction Data Source. This can not be edited.

- Enabled – Can be set to True or False to indicate whether this Data Source should be collected. Normally this is set to True, but you may wish to disable Data Sources occasionally while developing the Data Source or making changes to the web application being tested.

- Component – Any time the web transaction fails an event is generated within Zenoss. This field allows you to set the Component field of the generated event.

- Event Class – This allows you to select the Event Class of the events generated by this Data Source. Frequently this is set to /Status/Web.

- Timeout - Number of seconds zenwebtx will attempt to execute this Data Source's twill commands before it quits and generates an error event.

- Cycle Time - Number of seconds zenwebtx waits between the start of one test run and the start of the next.

- User Agent - This is the text zenwebtx will present to target web sites to identify itself.

1.5. Script Tab

The other page concerning WebTx Data Sources is the Script page, which you access via the Script tab when viewing the Data Source. This page is where you specify the details of the transaction. It also helps to debug your Twill commands when you are first setting up the Data Source.

- Initial URL – This is the url of the page where the transaction should start. This field frequently contains a TALES expression to refer to a device's id or ip address. For example, this field is often http://${dev/id} or http://${dev/manageIp}. For more details on TALES expressions see the Zenoss Admin Guide.

- TestDevice – This field is used for testing and debugging the Twill commands. You can enter the id of a device and press the Test Twill Commands button to have Zenoss execute the Twill Commands against that device. If you don't specify a device and you press the Test Twill Commands button then Zenoss will attempt to select a device for you.

- Upload Recording – This field allows the user to upload a web session recording generated by the FireFox TestGen4Web addon. If the user specifies a file here and clicks the Save button Zenoss will translate the file to Twill Commands and replace the contents of the Twill Commands field with these newly translated commands. NOTE: the current contents of the Twill Commands field is completely replaced when this is done.

- Twill Commands – These are the actual Twill commands that will be executed to produce values and events for the Data Source. Twill is described in more detail in the section below.

1.6. Creating the Twill Commands

ZenWebTx uses a language called Twill to specify the steps of a web test. Each WebTx Data Source has a field that contains the Twill commands that describe a web transaction. You can create this list of Twill commands manually or you can record a session in a browser and use that as the basis for your Data Source.

Some Twill commands specify an action, such as following a specific link on a page or entering data in a form field. Other Twill commands specify a test, such as searching for specific text on a page or making sure the title does not contain specific text. The full range of available commands are described in the accompanying ZenWebTx Twill Commands Reference.

1.7. TestGen4Web

TestGen4Web is an add-on for Firefox which allows you to record browser sessions. ZenWebTx can take these sessions and convert them to Twill as a starting point for developing ZenWebTx Data Sources. TestGen4Web can be found at https://addons.mozilla.org/firefox/addon/1385. Record a TestGen4Web session using the Record and Stop buttons on the TestGen4Web toolbar in Firefox. Once you have recorded a session use the Save button in the toolbar to save the session to a file. On the Script page of a ZenWebTx Data Source you can use the Browse button to select your saved session. When you click the Save button the TestGen4Web session is converted to Twill and placed into the Twill Commands field on that same page. Note: any previous Twill Commands will be replaced by the converted TestGen4Web commands.

1.8. Manually Creating Twill Commands

Even if you use TestGen4Web to initially create Twill commands, you will frequently want to edit these commands manually to add datapoints, additional content checks, etc. The ZenWebTx Twill Commands Reference describes in detail the commands that can be used. The Test Twill Commands button on the Script page is invaluable for testing your Twill commands as you create or edit them.

You can also execute twill commands interactively using the twill-sh program from the command line. This lets you enter commands one at a time and inspect the pages that come back. You can invoke twill-sh with:

```
> PYTHONPATH=$ZENHOME/Products/ZenWebTx/lib
$ZENHOME/Products/ZenWebTx/bin/twill-sh
```

Within twill-sh you can use the help command to list available commands or get a description of any specific command. Of particular interest are the following commands:

- showforms – lists the forms on the page and the fields within each

- showlinks – lists the links on the page

- show – list the actual source html from the page

- exit – quits twill-sh

Frequently the most convenient way to use twill-sh is to create a text file which contains your Twill commands. You can then specify that file on the command line when you invoke twill-sh. This lets you inspect the situation when your Twill commands hit a snag. You invoke twill-sh with a text file as such:

```
> PYTHONPATH=$ZENHOME/Products/ZenWebTx/lib
$ZENHOME/Products/ZenWebTx/bin/twill-sh -i myTwillCommands.txt
```

The -i option instructs twill-sh to leave you in the Twill shell rather than exiting when it is done running the commands in myTwillCommands.txt.

1.9. Event Generation

There are several situations where ZenWebTx will create events in Zenoss. These events will use the Component and Event Class specified on the Data Source tab. These situations are:

- ZenWebTx is unable to retrieve a page during the transaction
- One of the Twill commands fails, such as finding text that doesn't exist or following a link that doesn't exist
- The Timeout specified on the Data Source tab is exceeded.
- A Threshold defined for one of the Data Points in this Data Source is exceeded. Thresholds are defined within the Performance Template that contains the Data Source.

1.10. Data Points

Data produced by any Zenoss Data Source are called Data Points. ZenWeb Data Sources contain two Data Points by default:

- totalTime – this is the number of seconds taken to complete the entire transaction
- success – this is either 1 or 0 depending on whether the transaction succeeded or not

Other Data Points can be created by using the extract and printTimer Twill commands. These output data values when the Twill commands are run. You also need to create new Data Points with the same name you used in those commands to bring that data into Zenoss. The description of extract and printTimer in the ZenWebTx Twill Commands Reference contains more details.

2. Mail Transactions Monitoring (ZenMailTx) (Commercial)

2.1. About Mail Transaction Monitoring

ZenMailTX is an Enterprise ZenPack that allows you to monitor round-trip email delivery. It provides a new type of data source, MAILTX, that contains server and message information. ZenMailTX datasources are processed by a new daemon, zenmailtx.

2.2. Creating a ZenMailTX Data Source

To create your own MailTX Data Sources follow these steps:

1. Navigate to either a new or an existing Performance Template and select "New DataSource" from the Data Sources table menu.
2. Enter a name for the Data Source, select MAILTX as the type and click OK.
3. Enter the To Address and From Address.
4. Enter the SMTP server details, including the transport security (TLS or SSL) and port (default 25).
5. Enter the POP server details, including the transport security and port (default 110).

6. Enter the body of the test message.

7. Click the Save button.

You may test the Data Source settings by clicking the “Test Now” tab, entering a device against which to test the Data Source, and clicking “Test Email.”

Any of the MAILTX fields can take TAL expressions. It is often convenient to use a custom variable to define the value of the password fields. Because those fields are expected to be set via custom variables they are plain text fields, and not the typical password entry field.

2.3. Zenmailtx daemon

The zenmailtx daemon sends the test email message via the SMTP server specified, retrieves the email message from the POP server specified, and sends the time taken to send, time taken to fetch, and total time back to Zenoss. This daemon appears on the Zenoss Daemons page and can be started, stopped and restarted from there. From the command line it can be controlled via the zenoss script or by issuing commands directly to the daemon, which is symlinked at $ZENHOME/bin/zenmailtx.

3. SQL Transaction Monitoring (ZenSQLTx) (Commercial)

3.1. About SQL Transaction Monitoring

ZenSQLTx is an Enterprise ZenPack that allows the user to test the availability and performance of MySQL and MSSQL servers. It provides a new type of Data Source, SQL, which contains a set of SQL queries to be executed against a server. SQL Data Sources are processed by the zencommand daemon.

3.2. How To Create an SQL Data Source

To create your own SQL Data Sources follow these steps:

1. Navigate to either a new or an existing Performance Template and select "New DataSource" from the Data Sources table menu.

2. Enter a name for the Data Source, select SQL as the type and click OK.

3. Choose MySQL or MS SQL as the Database Type. Set the Host Name on which the database is located (accepts a TALES expression, such as “${here/id}” or “${here/getManageIp}”). Also set the port on which the database server is listening (default MySQL port is 3306, default MSSQL port is 1433), the name of the database, and a username and password that has permission to connect to the database and run the queries you specify.

4. Specify the SQL queries that this Data Source should execute. A summary of MySQL syntax may be found here: http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html. A summary of MS SQL syntax may be found in the documentation accompanying the software.

5. Click the Save button. Notice that a datapoint is created corresponding to the total query time in milliseconds.

Chapter 35. Application Monitoring Enterprise ZenPacks

1. MS SQL Monitoring (MSSQLServer)(Commercial)

1.1. About MS SQL Transaction Monitoring

The MSSQLServer Enterprise ZenPack monitors Microsoft SQL Server and its related services. The ZenPack enables users to view graphs based on MS SQL Server Performance Counters and to monitor processes related to MS SQL Server. This ZenPack is dependent on the ZenWinPerf ZenPack in its use of the WinPerf datasource type.

1.2. Installed Objects

1.2.1. Device Class

- /Devices/Server/Windows/MSSQLServer

1.2.2. Event Class

- /Events/Win/MSSQLServer

1.2.3. WinServices

- MSSQLSERVER
- MSSQLServerOLAPService
- MsDtsServer
- ReportServer
- SQLBrowser
- SQLSERVERAGENT
- SQLServer
- SQLWriter
- msftesql

1.2.4. IpServices

- ms-sql-s

1.2.5. Performance Templates

- MSSQLServer

1.2.6. Datasources

- ssamFullScansPerSec
- ssbmBufferCacheHitRatio

- ssbmFreePages
- ssdDataFileSSizeKB
- ssgsUserConnections
- sslkAverageWaitTimeMs
- sslkLockRequestsPerSec
- sslkNumberOfDeadlocksPerSec
- ssltLatchWaitsPerSec
- ssmmConnectionMemoryKB
- ssssBatchRequestsPerSec

1.2.7. Thresholds

- ssbmBufferCacheHitRatioLessThan90Percent

1.3. Using the MSSQLServer ZenPack

After installation you will have a device class MSSQLServer. Place any devices with MSSQLServer installed into this device class. An MSSQLServer template will automatically be bound to your devices. Make sure the zenwinperf daemon is running. On a device, click on the ‘More’ – ‘Push Changes’ menu item. This will update the collectors. To view the graphs, click on the ‘Perf’ tab.

2. MS Exchange Monitoring (MSExchange)(Commercial)

2.1. About MS Exchange Monitoring

The MS Exchange EnterpriseZenPack is an application monitoring ZenPack that monitors Micosoft Exchange and its related services. The ZenPack enables users to view graphs based on MS Exchange Performance Counters and to monitor processes related to MS Exchange. This ZenPack is dependent on the ZenWinPerf ZenPack and its use of the WinPerf datasource type.

2.2. Installed Objects

2.2.1. Device Classes

- /Devices/Server/Windows/MSExchange

2.2.2. Event Classes

- /Events/Win/Exchange

2.2.3. Event Class Keys

- MSExchangeDSAccess_2064
- MSExchangeDSAccess_2069
- MSExchangeIS_9582

2.2.4. Alerting Rules

- MSExchangeISWMTotal16MBFreeBlocks

2.2.5. WinServices

- IISADMIN
- IMAP4Svc
- MSExchangeES
- MSExchangeIS
- MSExchangeMGMT
- MSExchangeMTA
- MSExchangeSA
- MSExchangeSRS
- POP3Svc
- RESvc
- SMTPSVC
- W3SVC

2.2.6. IP Services

- imap4-ssl
- msexch-routing

2.2.7. Performance Templates

- MSExchangeIS is automatically bound to Device Class MSExchange

2.2.8. Datasources

- logicalDiskFreeMegabytes
- memoryPercentCommittedBytesInUse
- mseisMailboxFolderOpensPerSec

- mseisMailboxLocalDeliveryRate
- mseisMailboxMessageOpensPerSec
- mseisRPCAveragedLatency
- mseisRPCOperationsPerSec
- mseisRPCRequests
- mseisVMLargestBlockSize
- mseisVMTotalFreeBlocks
- mseisVMTotalLargeFreeBlockBytes
- mseisVMTotal16MBFreeBlocks

2.2.9. Thresholds

- logicalDiskFreeMegabytesUnder100
- logicalDiskFreeMegabytesUnder25
- memoryPercentCommittedBytesInUseOver75
- mseisVMLargestBlockSizeUnder16
- mseisVMLargestBlockSizeUnder32
- mseisVMTotal16MBFreeBlocksUnder3
- mseisVMTotalLargeFreeBlockBytesUnder32

2.3. Performance Counters and Thresholds

ZenPack will automatically set up performance monitoring templates for the following core metrics:

2.4. Reports, Views & Dashboards

- MSExchangeAvailability.pt
- Availability Summary
- Up/Down%, Uptime %, Up Since (Dials?),

- Current status matrix for key exchange services

2.5. Using the the MSExchange ZenPack

After installation you will have a device class MSExchange. Place any devices with MSExchange installed into this device class. An MSExchange template will automatically be bound to your devices. Make sure the zenwinperf daemon is running. On a device, click on the ‘More’ – ‘Push Changes’ menu item. This will update the collectors. To view the graphs, click on the ‘Perf’ tab.

3. IIS Monitoring (Commercial)

3.1. About IIS Monitoring

The IISMonitor Enterprise ZenPack collects key metrics from Microsoft IIS. The metrics are collected using Windows Perfmon and require no agent to be installed on the IIS server.

- 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

3.2. Setting up IIS Monitoring

Follow these steps to setup your IIS server to have this information collected.

1. Make sure you have the ZenPack properly installed.
2. Verify that your Zenoss Windows service account has access to the IIS server. Within Zenoss this is specified using zWinUser and zWinPassword.
3. Bind the IIS performance template to the IIS server device within Zenoss.

4. JBoss Application Server Monitoring (Commercial)

4.1. About JBoss Application Server Monitoring

JBossMonitor is an Enterprise ZenPack that allows System Administrators to monitor JBoss Application Servers. JBossMonitor uses the JMX Remote API and accesses MBeans deployed within JBoss that contain performance information about the components that are being managed.

4.2. Overview

The collected performance information includes: pool sizes for data sources (JDBC), enterprise java beans (EJBs), message queues (JMS), threads, servlets, JSPs, and classloaders. Cache information is also accessible, providing system administrators insight into the number of hits (or misses) their cache policy has produced.

The ZenPack also aggregates individual performance metrics into higher level concepts that provide a picture of the performance of the application. Cache hits and misses are combined on the same graph to provide an overall picture of cache performance. Likewise, queue metrics are combined to show the number of messages currently on the queue, being processed, and being placed on the queue. Queue subscribers and publishers are also graphed.

Each of the individual performance metrics can be trended and predicted, and thresholds can be explicitly defined. Both the predicted thresholds and explicit thresholds inform system administrators of potential future problems before they occur. Since so much of J2EE involves "managed resources", the ability to monitor pool sizes and alert administrators prior to resources being exhausted is extremely valuable and can reduce the likelihood of a fatal outage caused by resource depletion.

Most of the metrics that are collected in JBossMonitor represent combinations of individual component metrics. For example, the Thread Pool metric represents all threads in all pools. It is possible to configure JBossMonitor to perform at higher granularity and have it monitor a Thread Pool with a particular name. However, since these names are application and specific we have chosen to configure JBossMonitor to collect at a rather coarse grained level out of the box. The installer is highly encouraged to customize and configure!

One particular performance template that screams for end-user configuration involves Servlets. If a site to be monitored is revenue generating, and credit card submissions from the website are handled via a back-end servlet it may be critically important to monitor the resources made available by the JBoss container to the servlet container. If the number of free spaces in the servlet pool dwindles to 0 it could prevent your application from making a sale.

4.3. Collected Metrics for JBoss

The following list shows al of the collected metrics for JBoss servers:

- Active Threads
- JMS Message cache memory usage
- JMS Message hits/misses
- JMS Topic/Destination queue size
- Java heap memory usage
- JCA commit, rollback, and transaction count
- JCA Connection pool in-use connections and available connections
- JCA connections created/destroyed
- JCA total connections
- JGroups cluster messages sent/received
- JGroups cluster bytes sent/received
- MBean creation/removal count
- MBean messages processed count

4.4. Setting Up the JBoss ZenPack

The JBoss Application server ZenPack does not require any additional setup after unzipping. Although you can customize the data points the same way you would customize any data point in Zenoss.

5. WebLogic Application Server Monitoring (Commercial)

5.1. About WebLogic Application Server Monitoring

WebLogicMonitor is an Enterprise ZenPack that allows System Administrators to monitor a WebLogic Server. WebLogicMonitor uses the JMX Remote API and and accesses MBeans deployed within JBoss that contain performance information about the components that are being managed. This performance information includes pool sizes for data sources (JDBC), threads, connections (JCA), queues (JMS), servlets, JSPs, enterprise java beans (EJB), timer queues.

Throughput is also monitored when it is available. This metric is computed by WebLogic and is based on the number of messages moving through a queue at any given time. The throughput metric gives a good picture of the health of the messaging subsystem, which is commonly used throughout many enterprise applications. Stateless, Stateful, and Entity EJB performance metrics are monitored, as are message driven bean performance.

Security realms are also monitored for potential denial of service attacks. This includes recording of authentication failures, broken out by valid accounts, invalid accounts, and accounts that are currently locked out. Application specific realms can be monitored by customizing the built in WebLogic default realm.

All of the observed values are fed into the comprehensive Zenoss thresholding and event subsystem, allowing events to be correlated.

5.2. Weblogic Collected Metrics

5.2.1. Overall Application Server Vitals

- Number of total and active JMS connections and servers
- Overall number of JTA transactions that are rolled back or abandoned
- JTA transactions rolled back due to system, app, or resource issues
- Number of JTA rollbacks that timeout
- Active and committed JTA transaction count
- Timer exceptions, executions, and scheduled triggers
- User accounts that are locked and unlocked
- Authentication failures against locked accounts and non-existent accounts
- Total sockets opened, and the current number of open sockets
- JVM Mark/Sweep and Copy garbage collector execution counts

- Number of JVM daemon threads
- JVM Heap/Non-Heap used and committed memory

5.2.2. Entity EJB, Message Driven Bean (MDB), and Session EJB Subsystem Metrics

- Rollback and commit count on a per-EJB basis
- Bean pool accesses, cache hits, and cache misses
- Number of Beans in use, idle, and destroyed
- Number of activations and passivations

5.2.3. Data Pool (JDBC) metrics

- Leaked, Total, and Active connections
- Number of requests waiting for a connection
- Number of reconnect failures

5.2.4. Queue (JMS) Metrics

- Bytes received, currently active, and pending in the queue
- Number of queue consumers
- Number of current, pending, and receives messages

6. Tomcat Application Server Monitoring (Commercial)

6.1. About Tomcat Application Server Monitoring

TomcatMonitor is an Enterprise ZenPack that allows System Administrators to monitor the Tomcat Application Server. Tomcat is a web application container that conforms to many parts of the J2EE Specification.

The more extensive JBoss Application Server uses Tomcat as a Web Application engine to manage web applications deployed inside enterprise applications within JBoss. As a result, the TomcatMonitor ZenPack can be used to monitor Tomcat MBeans that are active within JBoss.

This ZenPack focuses on the metrics that Tomcat updates in it's internal MBean container that is accessible via the remote JMX API.

These metrics focus on attributes that relate to the servicing of web pages and primarily include thread pool size, CPU use, available file descriptors, JSP and servlet counts, and request counts.

TomcatMonitor places much emphasis on monitoring thread status because every web request is serviced in a separate thread. Each thread requires file descriptors to be maintained, and thus those are monitored as well. The amount of CPU time spent servicing each thread is also captured and reported.

TomcatMonitor also reports on the number of times JSPs and Servlets are reloaded. This metric can be useful in highly dynamic sites where JSPs or Servlets change on the fly and need to be reloaded periodically. Monitoring of this metric can lead to the identification of small "Reloading Storms" before they cause production outages.

The amount of time Tomcat spends servicing a request is also recorded. This extremely high level metric can provide insight into downstream systems that are not monitored. If all the Tomcat resources are within normal tolerances but processing time suddenly spikes it can be an indication that a back-end service (such as a database or another web service) is misbehaving.

All metrics captured by TomcatMonitor can be predicted using the built-in Zenoss trending software. Explicit thresholds can also be defined that operate on top of the existing robust event engine.

These event based notifications can be useful at multiple granularities of monitoring, all the way from the high level "Web Processing Time" metric down to the number of times a servlet is being reloaded per minute.

6.2. Collected Metrics for Tomcat

The following metrics can be collected and graphed for Tomcat:

- Tomcat cache (accesses vs hits)
- Daemon and User thread count
- Overall CPU time
- Global Request Traffic: bytes sent/received
- Global Request Traffic: request count and error count
- Global Request processing time
- JSP/Servlet reload time
- Servlet class loading and processing time
- Servlet request and error count

6.3. Tomcat Monitoring Setup

The out-of-the-box TomcatMonitor data source configuration has been defined at the macro level, but can be configured to operate on a more granular basis. For example, the Servlet Reload Count applies to all servlets in all web applications but it could be narrowed to be Servlet /submitOrder in web application "production server".

Chapter 36. Integration Enterprise ZenPacks

1. Remedy Integration in Zenoss (Commercial)

1.1. About Remedy Integration

The RemedyIntegrator Enterprise ZenPack provides a way to have Zenoss automatically open tickets in your Remedy system when specific events occur. The cases are opened by Zenoss sending a specially formatted email to the Remedy service's email address.

1.2. Setting Up Remedy Integration

Follow these steps to setup the integration once you have installed the RemedyIntegrator ZenPack.

1. Go to Settings/Users within the Zenoss web interface.

   You will find a new user named "Remedy."

2. Change the email address for this user to the email address that your Remedy service picks up email on.

3. Go to the Alerting Rules tab of the Remedy user.

4. Click on the "Open Ticket" alerting rule to edit it.

5. Click on the Message tab and modify the following attributes for your specific setup:

   - Server: remedy.yourdomain.com
   - Login: remedyUsername
   - Password: remedyPassword
   - Support Company: Your Company
   - Support Organization: YourSupport
   - Assigned Group: YourGroup

     ... and any other attributes you would like to customized ...

6. Click Save then go back to the Edit tab.

7. Set the alerting rule to Enabled and adjust the event filter to your requirements.

2. RANCID Integration (Commercial)

2.1. About RANCID Integration

RANCIDIntegrator is an Enterprise ZenPack designed to allow integration between the popular RANCID (ht-tp://www.shrubbery.net/rancid/) configuration management tool and Zenoss. The integration points between the tools can be described as following:

- Zenoss will build the router.db file for RANCID. This allows for the centralization of administration activities and reduces the duplication of effort normally required to maintain the two tools.

- Implementation of this feature is as easy as adding a cronjob to execute $ZENHOME/bin/zenrancid as often as you'd like the router.db file to be updated.

- Zenoss will automatically run RANCID's rancid-runm tool on a single device in response to a ciscoConfigManEvent SNMP trap being sent from the device to Zenoss. Cisco devices will send this trap whenever their configuration is changed. This allows for real-time capturing of router configuration changes in your CVS repository.

2.2. Setting Up RANCID Integration

To implement this feature you must configure your Cisco routers to send their SNMP traps to the Zenoss server.

Link from Cisco router device status pages to the most recent configuration stored in your CVS repository via viewvc (http://www.viewvc.org/).

To implement this feature, change the zRancidUrl to your viewvc URL, other configuration options can be set using the following zProperties:

- zRancidRoot: File system directory where RANCID is installed. It may be NFS mounted from the real RANCID server. Default is "/opt/rancid"

- zRancidUrl: Base URL to viewvc. Default is http://rancid.mydomain.com/viewvc

- zRancidGroup: RANCID group attribute. Controls what router.db file the device is written to. Can be set at the device class or device level. Default is "router" on the /Network/Router/Cisco class.

- zRancidType: RANCID type attribute. Controls what device type is written to the router.db file. Can be set at the device class or device level. Default is "cisco" on the /Network/Router/Cis

Appendix A. Net-SNMP and Zenoss

1. Net-SNMP Configuration Settings

These are the Net-SNMP configurations settings that are needed for Zenoss. Add these lines to your snmpd.conf.

1.1. Community Information

This line will map the community name "public" into a "security name":

```
# sec.name source community
```

```
com2sec notConfigUser default public
```

This line will map the security name into a group name:

```
# groupName securityModel securityName
```

```
group notConfigGroup v1 notConfigUser
```

```
group notConfigGroup v2c notConfigUser
```

This line will create a view for you to let the group have rights to:

```
# Make at least snmpwalk -v 1 localhost -c public system fast again.
```

```
# name incl/excl subtree mask(optional)
```

```
view systemview included .1
```

This line will grant the group read-only access to the systemview view.

```
# group context sec.model sec.level prefix read write
notif
access notConfigGroup "" any noauth exact systemview
none none
```

1.2. System Contact Information

It is also possible to set the sysContact and sysLocation system variables through the snmpd.conf file:

```
syslocation Unknown (edit /etc/snmp/snmpd.conf)
```

```
syscontact Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
```

```
# Added for support of bcm5820 cards. pass .1 /usr/bin/ucd5820stat
```

1.3. Extra Information

See the snmpd.conf manual page, and the output of "snmpd -H".

```
trapcommunity public
```

```
trapsink default
```

Appendix B. Event Database Dictionary

1. Event Database Dictionary

Event Database Field	Description
dedupid	events will deduplicate based on the value of this field. by default: device, component, eventClass, eventKey, severity
device	name of device
component	name of component (like eth0, httpd, etc)
eclass	eventClass (if not specified maybe added by rule process if this fails will be /Unknown)
eventKey	If a component needs further deduplication specification this field maybe used
summary	message text truncated at 150 characters
message	full message text
severity	number from 0 to 5
eventState	state of event 0 = new, 1 = acknowledged, 2 = suppressed
eventClassKey	key by which rules processing begins. Often equal to component.
eventGroup	logical group of event source (syslog, ping, nteventlog etc)
stateChange	last time event changed automatically updated
firstTime	unix timestamp when event is received.
lastTime	last time an event was received
count	number of times an event has repeated
prodState	prodState of the device context
suppid	id of event that suppressed this event
manager	fqdn of the collector from which this event came
agent	collector name from which event came (zensyslog, zentrap, etc)
DeviceClass	device class from device context
Location	device location from device context
Systems	device systems from device context separated by |
DeviceGroups	device systems from device context separated by |
ipAddress	IP from which event came
facility	syslog facility of this is syslog event
priority	syslog priority of this is syslog event
ntevid	nt event id if this is nt eventlog event.

Appendix C. TALES Expressions

1. About TALES Expressions

TALES is syntax you can use to retrieve values call methods on Zenoss objects. Several fields in Zenoss accept TALES syntax, including command templates, event mapping transforms, user commands, event commands, zProperties, zLinks and others.

Commands (those associated with devices as well as those associated with events) can use TALES expressions to incorporate data from the related devices and/or events. TALES is a syntax for specifying expressions that let you access the attributes of certain objects such as a device or an event in Zenoss. For additional documentation on TALES syntax please see the TALES section at:

http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/AppendixC.stx

Depending on the context you may have access to a device and/or an event. Below is a list of the attributes and methods you may wish to use on device and event objects. The syntax for accessing device attributes and methods is ${dev/attributename}, so for example to get the manageIp of a device you would use ${dev/manageIp}. For events, the syntax is ${evt/attributename}

A command to ping a device might look like this. The ${..} is a TALES expression to get the manageIp value for the device.

```
{{{
ping -c 10 ${dev/manageIp}
}}}
```

More Examples:

DNS forward lookup

(assumes device/id is a resolvable name)

```
{{{
host ${device/id}
}}}
```

DNS reverse lookup

```
{{{
host ${device/manageIp}
}}}
```

SNMP Walk

```
{{{
snmpwalk -v1 -c${device/zSnmpCommunity} ${here/manageIp} system
}}}
```

To use these expressions effectively you need to know which objects, attributes and methods are available to you in which contexts. Usually there is a dev and/or device which allows you access the device in a particular context. Contexts related to a particular event usually have evt and/or event defined. Some available attributes for each of these classes are listed below. List items with parenthesis after them are methods and much have the parenthesis included in the TALES expression to function correctly.

2. TALES Device Attributes

Attribute	Description
getId	The primary means of identifying a device within Zenoss

Attribute	Description
getManageIp	The IP address used to contact the device in most situations
productionState	The production status of the device: Production, Pre Production, Test, Maintenance or Decommisioned. This attribute is a numeric value, use getProductionStateString for a textual representation.
getProductionStateString	Returns a textual representation of the productionState
snmpAgent	The agent returned from snmp collection
snmpDescr	The description returned by the snmp agent
snmpOid	The oid returned by the snmp agent
snmpContact	The contact returned by the snmp agent
snmpSysName	The system name returned by the snmp agent
snmpLocation	The location returned by the snmp agent
snmpLastCollection	When snmp collection was last performed on the device. This is a DateTime object.
getSnmpLastCollectionString	Textual representation of snmpLastCollection
rackSlot	The slot name/number in the rack where this physical device is installed
comments	User entered comments regarding the device
priority	A numeric value: 0 (Trivial), 1 (Lowest), 2 (Low), 3 (Normal), 4 (High), 5 (Highest)
getPriorityString	A textual representation of the priority
getHWManufacturerName	Name of the manufacturer of this hardware
getHWProductName	Name of this physical product
getHWProductKey	Used to associate this device with a hardware product class
getOSManufacturerName	Name of the manufacturer of this device's operating system.
getOSProductName	Name of the operating system running on this device.
getOSProductKey	Used to associate the operating system with a software product class
getHWSerialNumber	Serial number for this physical device
getLocationName	Name of the Location assigned to this device
getLocationLink	Link to the Zenoss page for the assigned Location
getSystemNames	A list of names of the Systems this device is associated with
getDeviceGroupNames	A list of names of the Groups this device is associated with
getOsVersion	Version of the operating system running on this device
getLastChangeString	When the last change was made to this device
getLastPollSnmpUpTime	Uptime returned from snmp
uptimeStr	Textual representation of the snmp uptime for this device
getPingStatusString	Textual representation of the ping status of the device
getSnmpStatusString	Textual representation of the SNMP status of the device

3. TALES Event Attributes

Attribute	Description
dedupid	A key used to correlate duplicate events
evid	A unique id for the event
device	The id of the associated device, if applicable
ipAddress	The IP Address of the associated device, if applicable
component	The component of the associated device, if applicable
eventClass	The event class associated with this device
eventGroup	logical group of event source (syslog, ping, nteventlog etc)
eventKey	The eventKey is the primary criteria for mapping events into event classes
facility	syslog facility of this is syslog event
severity	One of 0 (Clear), 1 (Debug), 2 (Info), 3 (Warning), 4 (Error) or 5 (Critical)
priority	syslog priority of this is syslog event
summary	Text description of the event
stateChange	When the mysql record for this event was last modified
firstTime	The first time this event was seen
lastTime	The last time this event was seen and it's count incremented
count	Number of times this event has been seen
prodState	prodState of the device context
manager	fqdn of the collector from which this event came
agent	collector name from which event came (zensyslog, zentrap, etc)

zProperties are also available for devices and events using the same syntax as above.

Appendix D. Device Preparation

1. Net-SNMP

By default Net-SNMP does not publish the full SNMP tree. Check to see if that is currently the case on a device and configure it correctly.

1. Confirm snmpd is running:

   ```
   > snmpwalk -v1 -cpublic <your device name> system
   ```

2. Retrieve the IP table for the device with snmpwalk:

   ```
   > snmpwalk -v1 -cpublic <your device name> ip
   ```

   Typical SNMP View:

   ```
   view systemview included .1
   view systemview included .1.3.6.1.2.1.25.1
   access notConfigGroup "" any noauth exact systemview none none
   ```

2. SNMP V3 Support

Zenoss has initial support for SNMPv3 data collection.

The following new zProperties control the authentication and privacy of these requests:

- zSnmpAuthType: use either "MD5" or "SHA" signatures to authenticate SNMP requests
- zSnmpAuthPassword: the shared private key used for authentication. Must be at least 8 characters long.
- zSnmpPrivType: either "DES" or "AES" cryptographic algorithms.
- zSnmpPrivKey: the shared private key used for encrypting SNMP requests. Must be at least 8 characters long.
- zSnmpSecurityName: the Security Name (user) to use when making SNMPv3 requests.

If zSnmpPrivType and zSnmpPrivPassword are set, the message is sent with privacy and authentication. If only the zSnmpAuthType and zSnmpAuthPassword are set, the message is sent with Authentication but no Privacy. If neither the Priv or Auth values are set, the message is sent with no authentication or privacy. It is an error to set the PrivType and PrivPassword without also setting an AuthType and AuthPassword.

SNMPv3 encryption using the AES (Advanced Encryption Standard) algorithm is supported only if the host platform net-snmp library supports it.

At the time of this writing, RedHat 5 does not, Ubuntu 7.10 does not, but OpenSuSE 10.2 and the Zenoss Appliance do.

You can determine if your platform supports AES with the following test:

```
$ snmpwalk -x AES 2>&1 | head -1
```

If the response is:

```
"Invalid privacy protocol specified after -x flag: AES"
```

then your platform does *not* support AES encryption for SNMPv3.

If the response is:

```
"No hostname specified."
```

Then your platform *does* support AES.

SNMPv3 Traps are not supported by Zenoss in version 2.1.1.

3. Forwarding Syslog Messages from UNIX/Linux Devices

Zenoss has its own syslog server, zensyslog. Managed devices should point their syslog daemons to zenoss. To do this, edit the /etc/syslog.conf file and add an entry where 1.2.3.4 is the zensyslog IP.

1. Log on to the target device (as a super user).
2. Open /etc/syslog.conf file with a text editor (e.g VI).
3. Enter *.debug and press the Tab key. then enter the host name or IP address of the Zenoss server. See example below:

   ```
   *.debug @192.168.X.X
   ```

4. Save the file and exit the file editor program.
5. Restart the Syslog service using the command below:

   ```
   /etc.init.d/syslog restart
   ```

4. Forwarding Syslog Messages from a Cisco IOS Router

Here are some links to Cisco commands to turn on syslog. Typically, it’s easier to use syslog than SNMP traps from network devices. The most basic IOS command to send syslog messages is:

```
logging 1.2.3.4
```

4.1. Other Cisco Syslog Configurations

Here are some additional configurations for other Cisco devices. To set up these configurations follow the following steps using the configurations that follow below.

1. Log on to the target router.
2. Type the command enable at the prompt.
3. Once you are prompted for a password, enter the correct password.
4. Type the command config at the prompt.
5. Type the command terminal at the configuration prompt.
6. At the prompt, Set the Syslog forwarding mechanism. See example below:

   logging <IP address of the Zenoss server>

7. Exit out all the prompts to the main router prompt.

Catalyst

```
set logging server enable
set logging server 192.168.1.100
set logging level all 5
set logging server severity 6
```

Local Director

```
syslog output 20.5
no syslog console
syslog host 192.168.1.100
```

PIX Firewalls

```
logging on
logging standby
logging timestamp
logging trap notifications
logging facility 19
logging host inside 192.168.1.100
```

5. Forwarding Syslog Messages from a Cisco CatOS Switch

1. Log on to the target switch.

2. Type the command enable at the prompt.

3. Once you are prompted for a password, enter the correct password.

4. Set the Syslog forwarding mechanism. See example below:

   ```
   set logging server <IP address of the Zenoss server>
   ```

5. You can set the types of logging information that you want the switch to provide with the commands below as examples:

   ```
   set logging level mgmt 7 default
   set logging level sys 7 default
   set logging level filesys 7 default
   ```

6. Forwarding Syslog Messages using Syslog-ng

Here is an example for FreeBSD and Linux platforms.

1. Log on to the target device (as a super user)

2. Open /etc/syslog-ng/syslog-ng.conf file with a text editor (e.g VI).

3. Add source information to file. See example below:

   FreeBSD:

   ```
   source src { unix-dgram("/var/run/log"); internal ();};
   ```

   Linux: (will gather both system and kernel logs)

   ```
   source src {
   internal();
   unix-stream("/dev/log" keep-alive(yes) max-connections(100));
   pipe("/proc/kmsg");
   ```

```
udp();
};
```

4. Add destination information (in this case, the Zenoss server). See example below:

```
log { source(src); destination(zenoss); };
```

Appendix E. ZenWebTx Twill Commands Reference

1. About Twill Commands

Twill is the language used by ZenWebTx to simulate user actions in a web browser and to test pages retrieved by the simulation. Below is a list of the Twill commands available for use within ZenWebTx Data Sources. For further information on ZenWebTx please see the ZenWebTx documentation.

Some Twill commands below produce text output, especially those under the heading Display. These commands do not affect the execution of tests by ZenWebTx, but they may be useful in testing and debugging your ZenWebTx Data Sources. Their output can be seen when using the Test Twill Commands button on the Script page of ZenWebTx Data Sources.

2. Browsing

- go <url> - visit the given URL.
- back - return to the previous URL.
- reload - reload the current URL.
- follow <link name> - follow a link on the current page

3. Assertions

- code <code> - assert that the last page loaded had this HTTP status, e.g. ``code 200`` asserts that the page loaded fine.
- find <regexp> - assert that the page contains this regular expression.
- notfind <regexp> - assert that the page *does not* contain this regular expression.
- url <regexp> - assert that the current URL matches the given regexp.
- title <regexp> - assert that the title of this page matches this regular expression.

4. Display

- echo <string> - echo the string to the screen.
- redirect_output <filename> - append all twill output to the given file.
- reset_output - display all output to the screen.
- save_html [<filename>] - save the current page's HTML into a file. If no filename is given, derive the filename from the URL.
- show - show the current page's HTML.
- showlinks - show all of the links on the current page.
- showforms - show all of the forms on the current page.
- showhistory - show the browser history.

5. Forms

- submit *[<n>]* - click the n'th submit button, if given; otherwise submit via the last submission button clicked; if nothing clicked, use the first submit button on the form. See `details on form handling` for more information.
- formvalue <formnum> <fieldname> <value> - set the given field in the given form to the given value. For read-only form widgets/controls, the click may be recorded for use by submit, but the value is not changed unless the 'config' command has changed the default behavior. See 'config' and `details on form handling` for more information on the 'formvalue' command. For list widgets, you can use 'formvalue <formnum> <fieldname> +value' or 'formvalue <formnum> <fieldname> -value' to select or deselect a particular value.
- fv - abbreviation for 'formvalue'.
- formaction <formnum> <action> - change the form action URL to the given URL.
- fa - abbreviation for 'fa'.
- formclear - clear all values in the form.
- formfile <formspec> <fieldspec> <filename> [<content_type>]* - attach a file to a file upload button by filename.

6. Cookies

- save_cookies <filename> - save current cookie jar into a file.
- load_cookies <filename> - replace current cookie jar with file contents.
- clear_cookies - clear all of the current cookies.
- show_cookies - show all of the current cookies. Sometimes useful for debugging. See note under the Display heading above.

7. Debugging

- debug <what> <level> - turn on or off debugging/tracing for various functions. The first argument is either 'http' to show HTTP headers, 'equiv-refresh' to test HTTP EQUIV-REFRESH headers, or 'twill' to show twill commands. The second argument is '0' for off, '1' for on.

8. Other Commands

- tidy_ok - check to see if 'tidy' runs on this page without any errors or warnings. (`tidy` is very stringent - you've been warned!)
- exit *[<code>]* - exit with the given integer code, if specified. 'code' defaults to 0.
- run <command> - execute the given Python command.
- runfile <file1> [<file2> ...]* - execute the given files.

- agent - set the browser's "User-agent" string.
- sleep [<seconds>] - sleep the given number of seconds. Defaults to 1 second.
- reset_browser - reset the browser.
- extend_with <module> - import commands from Python module. This acts like ``from <module> import *`` does in Python, so e.g. a function ``fn`` in ``extmodule`` would be available as ``fn``. See *examples/extend_example.py* for an example.
- add_auth <realm> <uri> <user> <password> - add HTTP Basic Authentication information for the given realm/URI combination. For example, “add_auth IdyllStuff http://www.idyll.org/ titus test” tells twill that a request from the authentication realm "IdyllStuff" under http://www.idyll.org/ should be answered with username 'titus', password 'test'. If the 'with_default_realm' option is set to True, ignore 'realm'.
- config [<key> [<value>]] - show/set configuration options.
- add_extra_headers <name> <value> - add an extra HTTP header to each HTTP request.
- show_extra_headers - show the headers being added to each HTTP request.
- clear_extra_headers - clear the headers being added to each HTTP request.

9. Details on Form Handling

Both the `formvalue` (or `fv`) and `submit` commands rely on a certain amount of implicit cleverness to do their work. In odd situations, it can be annoying to determine exactly what form field `formvalue` is going to pick based on your field name, or what form & field `submit` is going to "click" on.

Here is the pseudocode for how `formvalue` and `submit` figure out what form to use (function `twill.commands.browser.get_form`)::

for each form on page:

if supplied regexp pattern matches the form name, select

if no form name, try converting to an integer N & using N-1 as

an index into the list or forms on the page (i.e. form 1 is

the first form on the page).

Here is the pseudocode for how `formvalue` and `submit` figure out what form field to use (function `twill.commands.browser.get_form_field`)::

search current form for control name with exact match to fieldname;

if single (unique) match, select.

if no match, convert fieldname into a number and use as an index, if

possible.

if no match, search current form for control name with regexp match to fieldname;

if single (unique) match, select.

if *still* no match, look for exact matches to submit-button values.

if single (unique) match, select.

Here is the pseudocode for `submit`::

if a form was _not_ previously selected by formvalue:

if there's only one form on the page, select it.

otherwise, fail.

if a field is not explicitly named:

if a submit button was "clicked" with formvalue, use it.

otherwise, use the first submit button on the form, if any.

otherwise:

find the field using the same rules as formvalue

finally, if a button has been picked, submit using it;

otherwise, submit without using a button

10. ZenWebTx Extensions to Twill

ZenWebTx adds several additional commands to the standard Twill vocabulary.

10.1. twilltiming

This is used to set timers within a set of Twill commands. If you then define a Data Point for this timer you can graph and set Thresholds on this timer value. To use twilltiming you must include the line "extend_with twilltiming" once near the top of your ZenCommands. You then use the command "startTimer myTimerName" to start a new timer and "printTimer myTimerName" to output the value. Timer values should only be output once, so for example, to output the time from the start of the script to more than one point in the script you need to use more than one timer. Example:

extend_with twilltiming

setTimer wwwZenossCom

setTimer bothPages

go http://www.zenoss.com

printTimer wwwZenossCom

setTimer communityPage

follow "Community"

printTimer communityPage

printTimer bothPages

To use these timers in Zenoss you create Data Points with the same name as the timers. In this example you could create Data Points named wwwZenossCom, communityPage and bothPages. You could then use there Data Points in Zenoss Thresholds and Graph Definitions.

10.2. twillextract

twillextract is used to extract numeric values from web pages during the transaction. To use twillextract you include the command "extend_with twillextract" once near the top of your Twill commands. You then use the command "extract <dataName> <regularExpression>" to match the given regular expression to the current page. The value 1 or 0 is assigned to dataName depending on whether the regular expression matched or not. Additionally, you can use python's regular expression substring matching syntax to extract substrings of the matched text. For example, http://www.zenoss.com contains a copyright notice near the bottom that looks like "Copyright (c) 2005-2007, Zenoss Inc." The following Twill commands use a regular expression to grab the 2nd year from that notice:

extend_with twillextract

go http://www.zenoss.com

extract copyright "(?P<firstYear>[0-9]*)-(?P<secondYear>[0-9]*) Zenoss, Inc"

(?P<name>.....) is python syntax for naming that particular part of the regular expression. The value extracted from that part of the matching text is given the name from the extract command, then a dash, then the name from the sub pattern. In this example copyright gets a value of 1 or 0 depending on whether the pattern was found on the page or not, and copyright-firstYear and copyright-secondYear get the values extracted from the matched text. To use these values in Zenoss you must create Data Points in the WebTx Data Source with the same name as those you used in the extract command. In this case you would create Data Points named copyright, copyright-firstYear and copyright-secondYear. You can then create Graph Definitions and Thresholds for these Data Points if you wish.

10.3. ignorescripts

This is used to strip javascript from visited pages before they are processed by Twill. Usually Twill ignores script tags, however it is possible for script to include strings that Twill will interpret as HTML tags. Including the command "extend_with ignorescripts" near the top of your Twill commands will cause all script tags to be stripped thereby avoiding this issue.

Appendix F. Basic Troubleshooting in Zenoss

1. How To Troubleshoot Zenoss Daemons

Help for figuring out why collectors are failing

If Zenoss behaves strangely or otherwise does not do what you expect it to do, you can enable debugging messages to help you diagnose problems.

To do so, edit the configuration file of the daemon where you want to enable debugging: Go to Settings/Daemons/edit config. Then change the line "level info" in the section eventlog to "level debug". Save, then restart the daemon (under tab "Daemons").

You should now see debug information in the daemon's logfile unter $ZENHOME/log.

Do not forget to turn off debugging when you are finished troubleshooting; the logfiles grow quickly in debug mode.

1.1. Identify the Problem

1.1.1. Check the logs

Your first step in troubleshooting a daemon is to look in the logs for errors. The Zenoss logs are located in in $ZENHOME/logs. You can view the logs via the web user interface under the About link.

1.1.2. Get more information

You can get additional logging by decreasing the verbose limit and running the daemon in foreground mode.

For example, this command will run zenperfsnmp:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10
```

See this page: http://www.zenoss.com/community/docs/howtos/zenoss-daemon-command-line-arguments/ for a list of command line options for the remaining Zenoss daemons.

You will get more (sometimes, a lot more) log messages, but zenperfsnmp will only perform a single scan of all the devices.

1.1.3. Find out what the program is _really_ doing

If a program is hanging, or not behaving as you expect, you can eavesdrop on what the program is asking the operating system to do on its behalf. This is a good way to determine if a helper program is failing, or system errors are not propagating up to the log file.

On Linux machines, the command to trace these system calls is "strace". On the appliance you can add strace to your system with:

```
$ conary update strace
```

Other posix-like operating systems have their own commands (truss on Solaris, dtrace on OS X). For example, you can verify that zenperfsnmp is really sending packets:

```
$ strace -f -e trace=sendto \
$ZENPERFSNMP/bin/zenperfsnmp run
```

Don't forget the largest and most complex Zenoss daemon: MySQL.

Check the version against the one needed by Zenoss. If you see "Lost Connection to Zenoss" in the dashboard, it is likely a MySQL connection problem.

1.2. Narrow the Problem

If you have managed to limit a problem to a single device, or simply suspect a device because it's running an odd configuration, or was recently added, most of the active collectors will allow you to scan a single device:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10 --device SomeDevice
```

If the problem is related to a long running server, you can ask that the server run in foreground mode, but continue with the normal endless cycle:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10 --cycle
```

1.3. Look for Conflicts

Are you running more than one copy of a daemon? During debugging and before version 1.1, Zenoss could lose track of background processes.

Stop zenoss and look for stray processes:

```
$ ps auxww | grep /z
```

Are you resource limited? Is the file system full? Do you have free memory? My favorite tool for this is "top" under Linux:

```
$ top
```

This program will constantly update the display with a list of the most CPU hungry programs. You can also sort the list by memory usage.

1.4. Reproduce the Problem

The ability to reproduce a problem with a consistent set of steps will help enormously. Often the only way to find the problem is to use "binary search". You reproduce the problem and take away "half" of the configuration. Slowly you can reduce the "halves" that are causing the problem until a single element remains.

1.5. Getting Help Solving the Problem

1.5.1. Search the Zenoss Forums

Hopefully someone has seen a similar problem. Their pain can save you time and energy. Also, we are always looking for patterns across users and that will help narrow down an issue.

1.5.2. Report the Problem to Zenoss

Some problems should be reported, even in the absence of detailed information because they are almost certainly bugs.

- the python interpreter crashes (a segmentation fault, for example)
- a python trace in a log file
- a daemon regularly drops heartbeats
- a daemon's size grows over time to consume all resources