



## ADMINISTRATION

Zenoss, Inc.

[www.zenoss.com](http://www.zenoss.com)

---

## Zenoss Administration

Copyright © 2010 Zenoss, Inc., 275 West St. Suite 204, Annapolis, MD 21401, U.S.A. All rights reserved.

This work is licensed under a Creative Commons Attribution Share Alike 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/>; or send a letter to Creative Commons, 171 2nd Street, Suite 300, San Francisco, California, 94105, USA.



The Zenoss logo is a registered trademark of Zenoss, Inc. Zenoss and Open Enterprise Management are trademarks of Zenoss, Inc. in the U.S. and other countries.

Flash is a registered trademark of Adobe Systems Incorporated.

Oracle, the Oracle logo, and Java are registered trademarks of the Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Linux is a registered trademark of Linus Torvalds.

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

Sybase is a registered trademark of Sybase, Inc.

Tomcat is a trademark of the Apache Software Foundation.

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

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

Part Number: 06-102010-3.0-v03

---

1. About Zenoss .....	1
1.1. High-Level View .....	1
1.1.1. Key Tenets .....	2
1.2. Architecture and Technologies .....	3
1.2.1. User Layer .....	3
1.2.2. Data Layer .....	3
1.2.3. Process Layer .....	4
1.2.4. Collection Layer .....	4
1.3. Monitoring Approach .....	4
1.3.1. File System Monitoring .....	4
1.4. Terminology .....	5
2. Using Zenoss .....	7
2.1. Interface and Navigation .....	7
2.1.1. Navigation .....	8
2.1.2. User Information Area .....	8
2.1.3. Portlets .....	8
2.1.3.1. Customizing Portlets .....	10
2.1.3.2. Adding and Duplicating Portlets .....	10
2.1.4. Network Map .....	10
2.1.4.1. Choosing the Network to Display .....	11
2.1.4.2. Viewing Device and Network Details .....	11
2.1.4.3. Loading Link Data .....	11
2.1.4.4. Filtering by Device Type .....	11
2.1.4.5. Adjusting Viewable Hops .....	12
2.1.4.6. Adjusting the Network Map .....	12
2.1.4.7. Viewing Device or Network Details .....	12
2.2. Customizing the Dashboard .....	12
2.2.1. Selecting Portlets .....	12
2.2.2. Arranging Portlets .....	13
2.2.3. Changing the Dashboard Column Layout .....	13
2.3. Search .....	13
2.4. Navigating the Event Console .....	14
2.4.1. Sorting and Filtering Events .....	15
2.4.1.1. Saving a Custom View .....	16
2.4.2. Refreshing the View .....	16
2.4.3. Viewing Event Details .....	16
2.4.4. Selecting Events .....	17
2.4.5. Managing Events .....	17
2.5. Running Commands .....	18
2.6. Creating and Using Alerts .....	18
2.6.1. Setting SMTP Settings For Alerts .....	18
2.6.2. Creating an Alerting Rule .....	19
2.6.2.1. Define and Enable the Alert .....	20
2.6.2.2. Create the Content of the Alert Message .....	21
2.6.2.3. Create a Schedule for Sending the Alert .....	21
2.6.3. Escalation of Alerting .....	22
2.6.3.1. Creating an Alerting Hierarchy .....	22
2.6.4. Adding Delay and Schedules to Alerting Rules .....	23
2.7. Creating Custom Event Views .....	24
3. Adding, Discovering and Modeling Devices .....	27
3.1. Adding Devices .....	27
3.1.1. Add a Single Device .....	27
3.1.2. Add Multiple Devices .....	28
3.2. Discovering Devices .....	29

---

3.2.1. Classifying Discovered Devices .....	30
3.2.2. Updating Device Authentication Details .....	30
3.2.3. Adding Information to a Device Record .....	31
3.3. Modeling Devices .....	31
3.3.1. Modeling Devices Using SNMP .....	31
3.3.1.1. Testing to See if a Device is Running SNMP .....	31
3.3.1.2. Configuring Windows Devices to Provide Data Through SNMP .....	32
3.3.1.3. Configuring Linux Devices to Provide Data Through SNMP .....	32
3.3.2. Modeling Devices Using SSH/COMMAND .....	32
3.3.2.1. Using Device Class to Monitor Devices Using SSH .....	33
3.3.3. Modeling Devices Using Port Scan .....	33
3.3.3.1. Using the /Server/Scan Device Class to Monitor with Port Scan .....	34
3.4. About Modeler Plugins .....	34
3.4.1. Viewing and Editing Modeler Plugins for a Device .....	34
3.4.1.1. Adding Plugins .....	34
3.4.1.2. Reordering Plugins .....	34
3.4.1.3. Deleting Plugins from a Device .....	35
3.5. Debugging the Modeling Process .....	35
4. Working with Devices .....	36
4.1. Viewing the Device List .....	36
4.1.1. Devices Hierarchy .....	36
4.1.2. Managing Multiple Devices from the Device List .....	36
4.2. Working with Devices .....	37
4.2.1. Events .....	38
4.2.2. Components .....	38
4.2.3. Software .....	39
4.2.4. Graphs .....	39
4.2.5. Administration .....	40
4.2.6. Configuration Properties .....	40
4.2.7. Modeler Plugins .....	41
4.2.8. Custom .....	41
4.2.9. Modifications .....	42
4.2.10. Monitoring Templates .....	42
4.3. Managing Devices and Device Attributes .....	42
4.3.1. Clearing Heartbeat Events .....	43
4.3.2. Pushing Configuration Changes to Zenoss .....	43
4.3.3. Locking Device Configuration .....	43
4.3.4. Renaming a Device .....	44
4.3.5. Remodeling a Device .....	44
4.3.6. Resetting the Device Manage IP Address .....	44
4.3.7. Resetting the Device Community .....	45
4.3.8. Deleting a Device .....	45
4.3.9. Dumping and Loading Devices Using XML .....	46
5. Properties and Templates .....	47
5.1. Configuration Properties .....	47
5.1.1. Configuration Properties Inheritance and Override .....	47
5.1.1.1. Viewing Properties from the User Interface .....	48
5.1.2. Configuration Property Types .....	50
5.1.3. Device Configuration Properties .....	50
5.1.4. Event Configuration Properties .....	53
5.1.5. Network Configuration Properties .....	53
5.2. Templates .....	55
5.2.1. Template Binding .....	55
5.2.1.1. Device Templates .....	55

5.2.1.2. Component Templates .....	56
5.2.1.3. Interface Templates .....	56
5.2.2. Examples .....	57
5.2.2.1. Example: Defining Templates in the Device Hierarchy .....	57
5.2.2.2. Example: Applying Templates to Multiple Areas in the Device Hierarchy .....	57
6. Core Monitoring .....	58
6.1. Availability Monitoring .....	58
6.1.1. Controlling Ping Cycle Time .....	58
6.1.2. Using the Predefined /Ping Device Class .....	58
6.1.3. Monitoring Processes .....	59
6.1.3.1. Monitoring a Process .....	59
6.1.4. Monitoring IP Services .....	60
6.1.4.1. Enabling IP Service Monitoring .....	61
6.1.4.2. Using the Predefined /Server/Scan Device Class .....	61
6.1.5. Monitoring Windows Services .....	61
6.1.5.1. Enabling Windows Service Monitoring .....	62
6.2. Performance Monitoring .....	62
6.2.1. About Performance Monitoring .....	62
6.2.2. About Monitoring Templates .....	42
6.2.2.1. Viewing Monitoring Templates .....	63
6.2.3. Template Binding .....	63
6.2.3.1. Binding Templates .....	63
6.2.4. Data Sources .....	64
6.2.4.1. Adding a Data Source .....	64
6.2.5. Data Points .....	65
6.2.6. Data Point Aliases .....	66
6.2.6.1. Alias Formula Evaluation .....	67
6.2.6.2. Adding a Data Point Alias .....	68
6.2.6.3. Reports That Use Aliases .....	68
6.2.7. Thresholds .....	68
6.2.7.1. MinMax Threshold .....	69
6.2.7.2. Adding Thresholds .....	69
6.2.8. Performance Graphs .....	71
6.2.8.1. Graph Points .....	72
6.2.8.2. Custom Graph Definition .....	74
6.2.8.3. Graph Commands .....	74
6.3. Monitoring Using ZenCommand .....	74
6.3.1. About ZenCommands .....	74
6.3.2. Example: Writing a ZenCommand (check_http example) .....	75
6.3.3. Example: Collect Data from A ZenCommand .....	76
6.3.4. Plugin Format for ZenCommands .....	76
6.3.5. Testing ZenCommands .....	77
6.4. SNMP Monitoring .....	78
6.5. Monitoring Devices Remotely Through SSH .....	78
6.5.1. Installing Plugins on the Remote Machine .....	78
6.5.1.1. Plugin Installation Technique: RPM .....	79
6.5.1.2. Plugin Installation Technique: setuptools .....	79
6.5.1.3. Testing the Plugin Installation .....	79
6.5.1.4. Troubleshooting Plugin Installation .....	79
6.5.1.5. Changing Zenoss to Monitor Devices Remotely Using SSH .....	80
6.5.1.6. Using the Predefined /Server/Cmd Device Class .....	81
6.6. Monitoring Windows Devices .....	81
6.6.1. Device Preparation for Windows Devices .....	81
6.6.2. Setting Windows Configuration Properties .....	82

6.6.3. Testing WMI on a Windows Server .....	82
6.6.4. Optional Windows Configuration .....	82
6.6.5. Modeling Services on Windows Devices .....	82
6.6.6. Collecting Windows Eventlog Events .....	83
6.6.7. Monitoring Windows Performance with SNMP Informant .....	83
6.6.8. Running winexe Commands on Windows Servers .....	83
7. Event Management .....	85
7.1. About Events .....	85
7.1.1. Basic Event Fields .....	85
7.1.1.1. device and ipAddress Fields .....	85
7.1.1.2. eventState Field .....	86
7.1.1.3. severity Field .....	86
7.1.1.4. summary and message Fields .....	86
7.1.1.5. evid .....	86
7.1.2. Other Fields .....	86
7.1.3. Details .....	88
7.1.4. De-Duplication .....	88
7.1.5. Auto-Clear Correlation .....	89
7.1.6. Event Consoles .....	90
7.1.6.1. Master Event Console .....	90
7.1.6.2. Creating Events .....	94
7.1.7. Event Sources .....	94
7.1.7.1. Generated Events .....	94
7.1.7.2. Captured Events .....	95
7.1.8. Creating Events Manually .....	95
7.1.8.1. Creating Events through the User Interface .....	95
7.1.8.2. Creating Events from the Command Line .....	96
7.1.9. Event Classes .....	96
7.1.9.1. Event Class Configuration Properties .....	96
7.1.10. Mapping and Transformation .....	97
7.1.10.1. Event Class Mappings .....	98
7.1.10.2. Event Class Transform .....	99
7.1.11. Event Life Cycle .....	100
7.1.11.1. Automatic Event Aging .....	101
7.1.11.2. Automatic Historical Event Cleanup .....	101
7.1.12. Event Commands .....	102
7.1.12.1. Creating Event Commands .....	102
7.1.13. Capturing Email Messages as Events .....	102
7.1.13.1. ZenMail .....	102
7.1.13.2. ZenPop .....	103
7.1.13.3. Translating Message Elements to the Event .....	103
7.1.14. SNMP Traps and Event Transforms .....	103
7.1.14.1. Classifying SNMP Traps .....	103
7.1.14.2. Example: Sending Test Traps .....	105
7.1.14.3. Transforming Events with Event Mappings .....	106
7.1.14.4. Event Transforms Based on Event Class .....	106
8. Production States and Maintenance Windows .....	107
8.1. About Production States and Maintenance Windows .....	107
8.2. Production States .....	107
8.2.1. Setting the Production State for Devices .....	107
8.3. Maintenance Windows .....	108
8.3.1. Maintenance Window Events .....	108
8.3.2. Creating and Using Maintenance Windows .....	109
8.3.2.1. Create a Maintenance Window for a Single Device .....	109

8.3.2.2. Create a Maintenance Window for a Group of Devices .....	109
9. Organizers and Path Navigation .....	111
9.1. About Organizers and Path Navigation .....	111
9.2. Classes .....	111
9.2.1. Viewing Device Classes .....	111
9.2.2. Adding Classes .....	112
9.2.2.1. Moving a Class .....	112
9.2.3. Setting Configuration Properties at the Class Level .....	113
9.3. Systems .....	113
9.3.1. Adding Systems .....	113
9.3.1.1. Moving a System .....	114
9.4. Groups .....	114
9.4.1. Adding Groups .....	114
9.4.1.1. Moving a Group .....	115
9.5. Locations .....	115
9.5.1. Adding Locations .....	115
9.5.1.1. Moving a Location .....	115
9.5.2. Integration with Google Maps .....	115
9.5.2.1. Obtaining an API Key .....	116
9.5.2.2. Setting an Address for a Location .....	117
9.5.2.3. Clearing the Google Maps Cache .....	117
9.5.2.4. Network Links .....	117
9.5.2.5. Google Maps Example .....	118
9.6. Inheritance .....	119
10. User Commands .....	120
10.1. About User Commands .....	120
10.2. Defining Global User Commands .....	120
10.2.1. Running Global User Commands .....	120
10.3. Defining User Commands for a Single Device .....	121
10.3.1. Running User Commands for a Single Device .....	122
10.4. Defining User Commands for All Devices in an Organizer .....	122
10.4.1. Running User Commands for Devices in an Organizer .....	122
10.5. User Command Example: Echo Command .....	122
11. Managing Users .....	124
11.1. About User Accounts .....	124
11.2. Creating User Accounts .....	124
11.3. Editing User Accounts .....	124
11.3.1. Associating Objects with Specific Users .....	125
11.3.1.1. Adding Administrators .....	126
11.4. User Groups .....	127
11.4.1. Viewing User Groups .....	127
11.4.2. Creating User Groups .....	127
11.5. Roles .....	128
11.6. Device Access Control Lists .....	128
11.6.1. About Device Access Control Lists .....	128
11.6.2. Key Elements .....	129
11.6.2.1. Permissions and Roles .....	129
11.6.2.2. Administered Objects .....	129
11.6.2.3. Users and Groups .....	129
11.6.2.4. Assigning Administered Object Access .....	129
11.6.2.5. Portlet Access Control .....	129
11.6.3. Setup and Configuration Examples .....	129
11.6.3.1. Restricted User with ZenUser Role .....	129
11.6.3.2. Restricted User with ZenManager Role .....	130

11.6.3.3. Adding Device Organizers .....	130
11.6.3.4. Restricted User Organizer Management .....	130
11.6.3.5. Viewing Events .....	130
11.6.4. Detailed Restricted Screen Functionality .....	130
11.6.4.1. Dashboard .....	130
11.6.4.2. Device List .....	130
11.6.4.3. Device Organizers .....	131
11.6.4.4. Reporting .....	131
12. Reporting .....	132
12.1. About Reporting .....	132
12.1.1. Organizing Reports .....	132
12.2. Basic Reports .....	133
12.2.1. Device Reports .....	133
12.2.2. Event Reports .....	134
12.2.3. Performance Reports .....	134
12.2.4. User Reports .....	136
12.3. Graph Reports .....	136
12.3.1. Creating a Graph Report .....	137
12.3.1.1. Adding Graphs .....	137
12.3.2. Customizing Graph Text .....	139
12.3.3. Printing Graphs .....	139
12.3.4. Organizing Graphs .....	139
12.4. Multi-Graph Reports .....	140
12.4.1. Creating A Multi-Graph Report .....	140
12.4.1.1. Adding Collections .....	141
12.4.1.2. Adding Graph Definitions .....	142
12.4.1.3. Adding Graph Groups .....	143
12.5. Creating Custom Device Reports .....	144
12.6. Exporting Reports .....	145
12.6.1. Advanced Task: Add An Export Button to a Report .....	145
12.7. Scheduling Reports .....	146
12.7.1. ReportMail Command Line Arguments .....	147
12.8. Using Reports to Troubleshoot System Daemons .....	147
12.9. Advanced Reports .....	148
13. ZenPacks .....	149
13.1. About ZenPacks .....	149
13.1.1. Provided ZenPacks .....	149
13.2. Installing ZenPacks .....	149
13.2.1. Installing from the Command Line .....	149
13.2.2. Installing from the User Interface .....	150
13.2.3. Installing All Core ZenPacks from RPM .....	150
13.2.4. Viewing Loaded ZenPacks .....	151
13.3. Creating ZenPacks .....	151
13.3.1. Why Create a ZenPack? .....	152
13.3.2. Create a ZenPack .....	152
13.3.3. Add a Database Object to a ZenPack .....	152
13.3.4. View Database Objects in a ZenPack .....	153
13.3.5. Remove a Database Object from a ZenPack .....	153
13.3.6. Adding Other Items to ZenPacks .....	153
13.4. Packaging and Distributing ZenPacks .....	153
13.5. Removing ZenPacks .....	154
13.6. Where to Find More Information .....	154
14. General Administration and Settings .....	155
14.1. Email and Pager Alerts .....	155

14.1.1. Editing SMTP and SNPP Information .....	155
14.2. Event Manager Settings .....	156
14.2.1. Editing Event Manager Settings .....	156
14.2.2. Changing Event Database Connection Information .....	157
14.2.3. Changing Event Manager Cache Settings .....	157
14.2.4. Changing Event Manager Maintenance Settings .....	157
14.3. Setting Portlet Permissions .....	158
14.3.1. User Role to ACL Mapping .....	158
14.3.2. Setting Permissions .....	158
14.3.3. Troubleshooting: Users Cannot See All Portlets .....	159
14.4. Backup and Recovery .....	159
14.4.1. Backup (zenbackup) .....	160
14.4.1.1. Backup Options .....	160
14.4.1.2. Create a Backup .....	161
14.4.1.3. Delete a Backup .....	161
14.4.1.4. Remote Backups .....	161
14.4.2. Restore (zenrestore) .....	161
14.4.2.1. Before You Restore (for Versions Earlier Than 2.4.5) .....	162
14.4.2.2. Restore Options .....	162
14.5. Working with the Job Manager .....	162
14.5.1. Viewing Jobs .....	162
14.5.2. Running the zenjobs Daemon .....	163
14.6. Maintenance and Performance Tuning .....	163
14.6.1. Packing the ZEO Database .....	163
14.6.2. Log Rotate Script .....	164
14.6.2.1. Zenoss 2.4.x .....	164
14.6.2.2. Zenoss 2.3.3 and Earlier .....	164
A. Daemon Commands and Options .....	165
A.1. Automated Modeling Daemons .....	165
A.2. Availability Monitoring Daemons .....	165
A.3. Event Collection Daemons .....	165
A.4. Performance Monitoring Daemons .....	166
A.5. Automated Response Daemons .....	166
B. SNMP Device Preparation .....	167
B.1. Net-SNMP .....	167
B.2. SNMP V3 Support .....	167
B.3. Community Information .....	168
B.4. System Contact Information .....	168
B.5. Extra Information .....	168
C. Using an Existing MySQL Server to Store Events .....	169
C.1. About .....	169
C.2. Procedure .....	169
D. Syslog Device Preparation .....	170
D.1. Forwarding Syslog Messages from UNIX/Linux Devices .....	170
D.2. Forwarding Syslog Messages from a Cisco IOS Router .....	170
D.2.1. Other Cisco Syslog Configurations .....	170
D.3. Forwarding Syslog Messages from a Cisco CatOS Switch .....	171
D.4. Forwarding Syslog Messages using Syslog-ng .....	171
E. TALES Expressions .....	172
E.1. About Tales Expressions .....	172
E.1.1. Examples .....	172
E.2. TALES Device Attributes .....	172
E.3. Tales Event Attributes .....	173
Glossary .....	175

# Chapter 1. About Zenoss

Zenoss is today's premier, open source IT management solution. Through a single, Web-based console, it enables you to manage the status and health of your infrastructure.

The power of Zenoss starts with its in-depth Inventory and IT Configuration Database. It creates this database by discovering *managed resources* -- servers, networks, and other devices -- in your IT environment. The resulting configuration model provides a complete inventory of your servers, network devices, and software applications, down to the level of *resource components* (interfaces, services and processes, and installed software).

Once Zenoss discovers the IT infrastructure, it automatically begins monitoring the performance of each device. It also provides events and fault management features that tie into the configuration database. These features help drive operational efficiency and productivity by automating many of the notification, alerts, escalation, and remediation tasks you perform each day.

## 1.1. High-Level View

Using agent-less technology, Zenoss monitors your entire IT infrastructure stack, including network, servers, HVAC and power, and even applications. At its highest level, the system comprises these major areas:

- Discovery and configuration
- Performance and availability
- Fault and event management
- Alerting and remediation
- Reporting

Zenoss unifies these areas into a single system with a modern, interactive Web user interface.

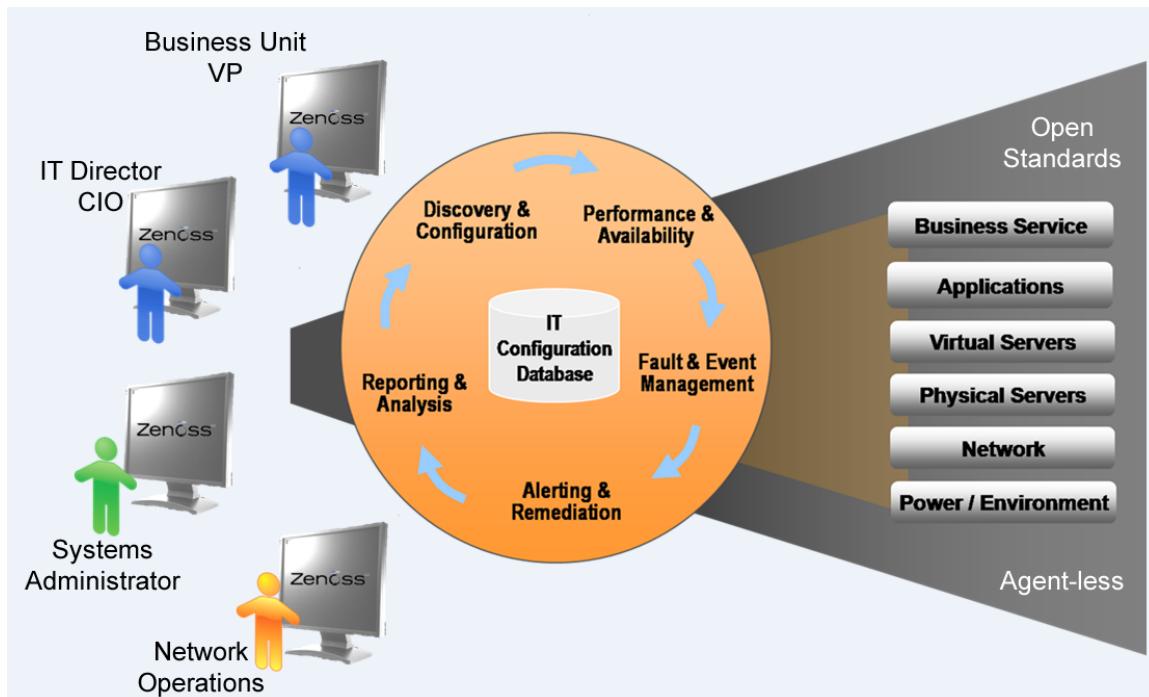


Figure 1.1. High-Level View

### 1.1.1. Key Tenets

Zenoss was designed with these important ideas at its core:

- **Modeling**

The system's model enables it to understand the environment in which it operates. Through sophisticated and detailed analysis, Zenoss determines how to monitor and manage complex IT environments. The core of the standard model describes basic information about each device's operating system and hardware. The model is object-based, and is easily extended through object inheritance.

- **Discovery**

With a sophisticated model, manual input and maintenance of data is challenging. To address this challenge, Zenoss uses *discovery* to populate the model. During discovery, the system accesses each monitored device in your infrastructure and interrogates it in detail, acquiring information about its components, network integration, and dependencies.

- **Normalization**

Because Zenoss collects information from different platforms and through different protocols, the amount and format of available information varies. For example, file system information gathered from a Linux server differs from similar information gathered from a Windows server. Zenoss standardizes the data gathered so that you can perform valid comparisons of metrics gathered by different methods and for different systems.

- **Agentless Data Collection**

To gather information, Zenoss relies on agent-less data collection. By communicating with a device through one of several protocols (including SNMP, SSH, Telnet, and WMI), it minimizes the impact on monitored systems.

- **Full IT Infrastructure**

Unlike other tools, the system's inclusive approach unifies all areas of the IT infrastructure--network, servers, and applications--to eliminate your need to access multiple tools.

- **Configuration Inheritance**

Zenoss extends the concept of inheritance in object-oriented languages to configuration. All core configuration parameters (*configuration properties*) and monitoring directions (*monitoring templates*) use inheritance to describe how a device should be monitored. Inheritance allows you to describe, at a high level, how devices should be monitored. It also supports ongoing refinements to the configuration. (For detailed information on inheritance and templates, refer to the chapter titled "Properties and Templates.")

- **Cross-Platform Monitoring**

Zenoss monitors the performance and availability of heterogeneous operating systems (including Windows, Linux, and Unix), SNMP-enabled network devices (such as Cisco), and a variety of software applications (such as WebLogic and VMware).

- **Scale**

You can deploy the system on a single server to manage hundreds of devices. The Enterprise version allows you to manage large, distributed systems by using horizontal scaling of its collectors.

- **Extensibility**

The system's extension mechanism, ZenPacks, allow for rapid addition and modification to customize your environment.

## 1.2. Architecture and Technologies

The following diagram illustrates the system architecture.

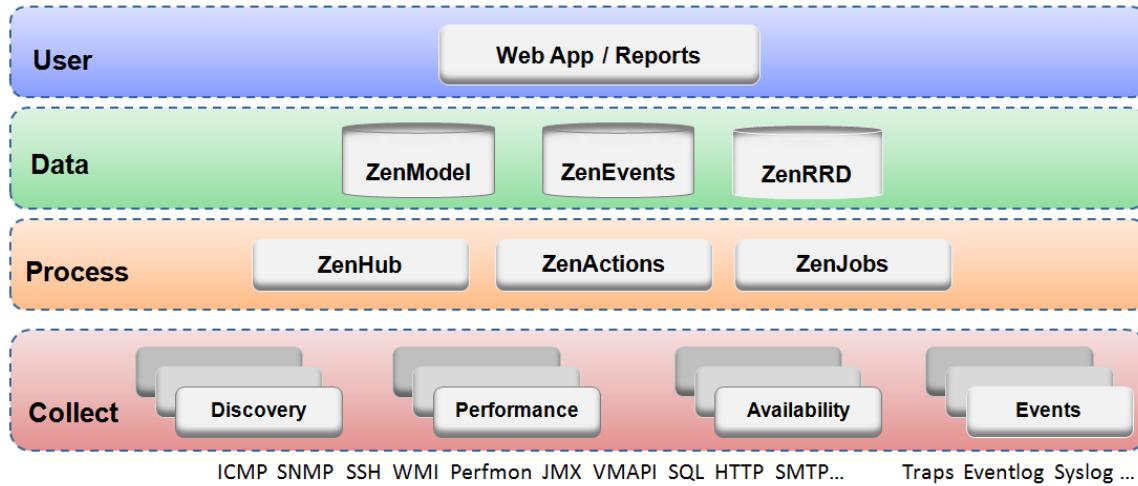


Figure 1.2. Architecture

Zenoss is a tiered system with four major parts:

- User layer
- Data layer
- Processing layer
- Collection layer

### 1.2.1. User Layer

Built around the Zope Web application environment, the user layer is manifested as a Web portal. It uses several JavaScript libraries, Mochi Kit, YUI, and extJS to provide a rich application experience.

Through the user interface, you access and manage key components and features. From here, you can:

- Watch the status of your enterprise, using the Dashboard
- Work with devices, networks, and systems
- Monitor and respond to events
- Manage users
- Create and run reports

The user layer interacts with the data layer and translates the information for display in the user interface.

### 1.2.2. Data Layer

Configuration and collection information is stored in the data layer, in three separate databases:

- **ZenRRD** - Utilizing RRDtool, stores time-series performance data. Because RRD files are stored locally to each collector, no bottlenecks result from writing to a single database as new collectors are added.
- **ZenModel** - Serves as the core configuration model, which comprises devices, their components, groups, and locations. It holds device data in the ZEO back-end object database.

- **ZenEvents** - Stores event data in a MySQL database.

### 1.2.3. Process Layer

The process layer manages communications between the collection and data layers. It also runs back-end, periodic jobs, as well as jobs initiated by the user (ZenActions and ZenJobs). The process layer utilizes Twisted PB (a bi-directional RPC system) for communications.

### 1.2.4. Collection Layer

The collection layer comprises services that collect and feed data to the data layer. These services are provided by numerous daemons that perform modeling, monitoring, and event management functions.

The modeling system uses SNMP, SSH, and WMI to collect information from remote machines. The raw information is fed into a plugin system (*modeling plugins*) that normalizes the data into a format that matches the core model.

Monitoring daemons track the availability and performance of the IT infrastructure. Using multiple protocols, they store performance information locally in RRD files, thus allowing the collectors to be spread out among many collector machines. Status and availability information, such as ping failures and threshold breaches, are returned through ZenHub to the event system.

For more information about system daemons, see the appendix titled "Daemon Commands and Options."

## 1.3. Monitoring Approach

Zenoss uses a model-driven approach to monitoring, combining discovery and the model to enable automatic monitoring. This strategy reduces system maintenance overhead and ensures that new devices and applications are monitored as they come online.

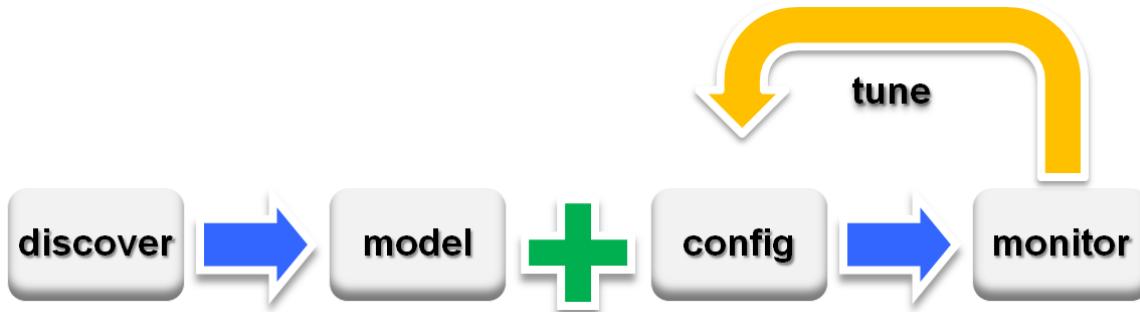


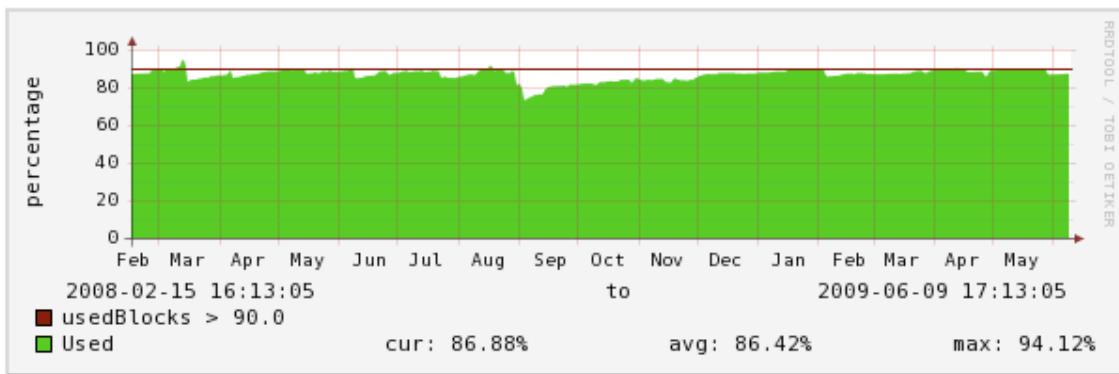
Figure 1.3. Workflow: Model-Driven Monitoring

As shown in the previous illustration, model-driven monitoring begins with discovery, which populates the model. It continues as the configuration defined in the model is automatically applied and monitoring begins. As the system runs, the configuration is further fine-tuned.

The model-driven monitoring approach is demonstrated by the following file system monitoring scenario.

### 1.3.1. File System Monitoring

By default, the system is configured with a file system threshold of 90% utilization. Each time it discovers a file system, this threshold is automatically applied to the file system, and monitoring begins.



*Figure 1.4. Monitored File System (Threshold Exceeded)*

This illustration shows the result of a system being monitored, using the default configuration. The graph shows that the threshold of 90% has been exceeded numerous times. Because the data in the model is normalized, thresholds will apply regardless of the collection mechanism (SNMP, SSH, and WMI).

The chapter titled "Properties and Templates" provides more information about modifying the monitoring configuration.

## 1.4. Terminology

You should understand product-specific use of the following terms when working with the system.

### Glossary

alert	Email or page sent as a result of an event.
data point	Data returned from a data source. In many cases, there is only one data point for a data source (such as in SNMP); but there may also be many data points for a data source (such as when a command results in the output of several variables).
data source	Method used to collect monitoring information. Example data sources include SNMP OIDs, SSH commands, and perfmon paths.
device	Primary monitoring object in the system. Generally, a device is the combination of hardware and an operating system.
device class	Special type of organizer used to manage how the system models and monitors devices (through configuration properties and monitoring templates).
device component	Object contained by a device. Components include interfaces, OS processes, file systems, CPUs, and hard drives.
discovery	Process by which Zenoss gathers detailed information about devices in the infrastructure. Results of discovery are used to populate the model.
event	Manifestation of important occurrence within the system. Events are generated internally (such as when a threshold is exceeded) or externally (such as through a syslog message or SNMP trap).
event class	Categorization system used to organize event rules.

event rules	Controls how events are manipulated as they enter the system (for example, changing the severity of an event). Configuration properties configure event rules.
graph	Displays one or more data points, thresholds, or both.
managed resource	Servers, networks, virtual machines, and other devices in the IT environment.
model	Representation of the IT infrastructure. The model tells the system "what is out there" and how to monitor it.
monitoring template	Description of what to monitor on a device or device component. Monitoring templates comprise four main elements: data sources, data points, thresholds, and graphs.
organizer	Hierarchical system used to describe locations and groups. Zenoss also includes special organizers, which are classes that control system configuration.
resource component	Interfaces, services and processes, and installed software in the IT environment.
threshold	Defines a value beyond which a data point should not go. When a threshold is reached, the system generates an event. Typically, threshold events use the /Perf event class.

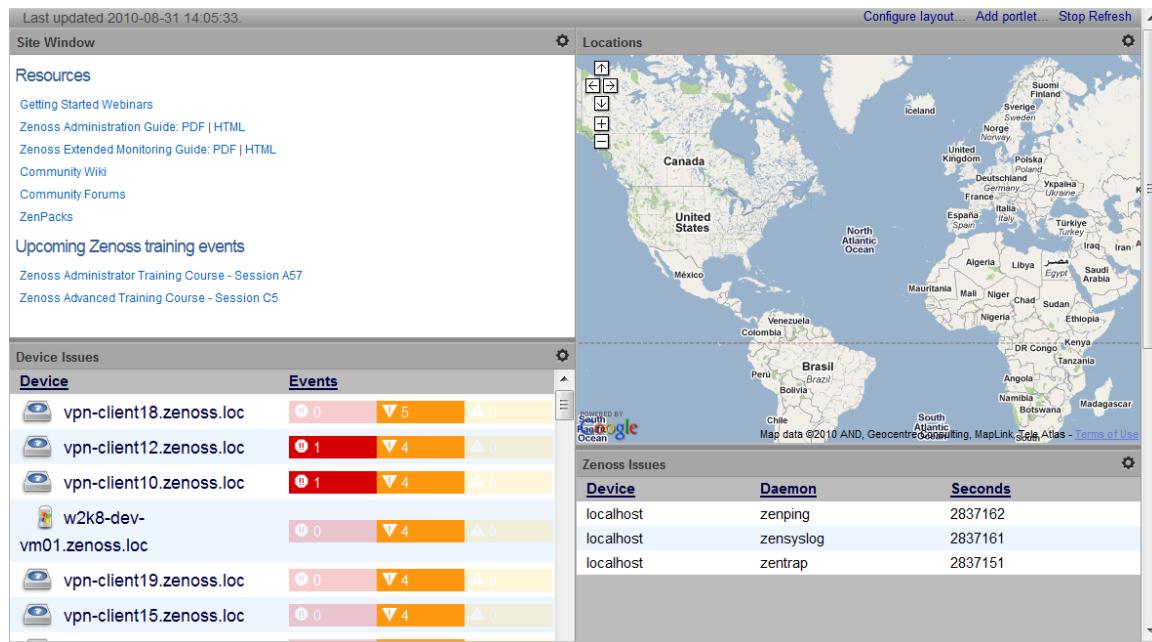
## Chapter 2. Using Zenoss

Read the following sections to learn more about working in the interface, and to learn how to:

- Customize the dashboard
  - Search for devices, events, and system objects
  - Navigate the event console
  - Run commands
  - Create and use alerts
  - Create custom event views

## 2.1. Interface and Navigation

After you install Zenoss and navigate to the interface from your Web browser, the Dashboard appears. The Dashboard provides at-a-glance information about the status of your IT infrastructure. It is the primary window into devices and events that the system enables you to monitor.



*Figure 2.1. Dashboard*

The Dashboard can show:

- System information resources and Web pages
  - Important error-level device events
  - Geographical high-level view
  - "Troubled" devices

Key Dashboard and interface areas include:

- Navigation bar
  - User information area

- Portlets
- System Network Map

## 2.1.1. Navigation

The Navigation menu lets you access major system features. In addition to the Dashboard, the menu is divided among several functional areas:

- **Events** - Guides you to the event management area, where you can monitor event status, events, history, configuration properties, and event transforms. You also can track changes made to events.
- **Infrastructure** - Offers access to network infrastructure, including, devices, networks, processes, and services.
- **Reports** - Allows you to view and define reports.
- **Advanced** - Provides access to monitoring templates, collectors, MIBs, and system settings.

## 2.1.2. User Information Area



Figure 2.2. User Information Area

The User information area offers information and selections:

- **Login ID** - The ID of the user currently logged in appears at the far left of this area. Click the ID to edit user settings, such as authentication information, roles, and groups. (You also can access user settings from the Navigation bar Advanced selection.)
- **Sign Out** - Click to log out of the system.
- **Help** - Click to access community product documentation, FAQs, and HowTos, at:

<http://community.zenoss.org/community/documentation>

## 2.1.3. Portlets

The main content of the Dashboard comprises portlets, which provide information about the system and your infrastructure. Portlets you can display on the dashboard are:

- **Device Issues** - Displays a list of devices, associated with color-coded events of error or critical severity levels. Click a device in the list to view its event log.

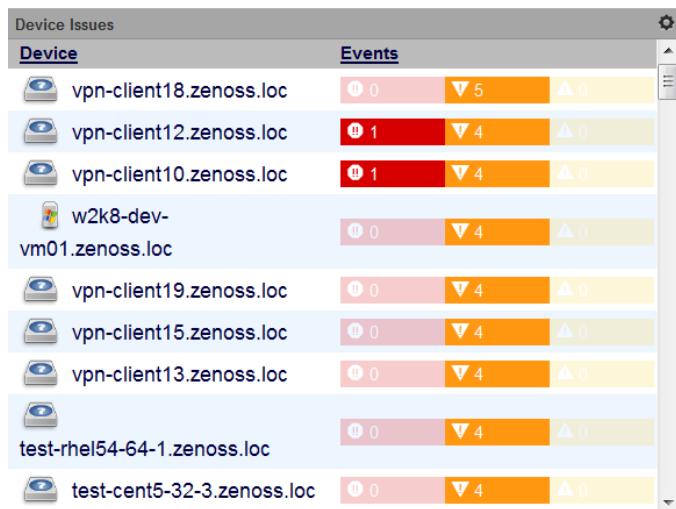


Figure 2.3. Device Issues Portlet

- **Google Maps** (device locations) - Shows configured locations and configured network connections.



Figure 2.4. Google Maps Portlet

- **Zenoss Issues** - Contains system self-monitoring information.
- **Production States** - Shows devices assigned to a particular production state.
- **Site Window** - Initially provides links to resources such as product guides, forums, and training events.

The URL for the default content is <http://www2.zenoss.com/in-app-welcome>. You can customize this portlet to display content from any URL.

- **Top Level (Root) Organizers** - Lists status for each grouping in your defined system hierarchy.



Figure 2.5. Top Level Organizers Portlet

- **Messages** - Displays system messages.

- **Object Watch List** - Allows the display of high-level status device classes, groups, systems, event classes, and locations that you select.

### 2.1.3.1. Customizing Portlets

You can customize each portlet that appears on the Dashboard. Customization options vary depending on the portlet type.

Click , which appears at the top right corner of a portlet, to view and customize display options. Click **Save Settings** to save your selections and then return to main portlet content.

The following table lists information you can customize for each Zenoss portlet.

<b>For this portlet type..</b>	<b>...you can customize:</b>
Welcome	Title, Refresh Rate, Destination URL
Device Issues	Title, Refresh Rate
Google Maps	Title, Refresh Rate, Base Location
Zenoss Issues	Title, Refresh Rate
Production States	Title, Refresh Rate, Production States (to appear on the Dashboard)
Top Level (Root Organizers)	Title, Refresh Rate, Root Organizer (to appear on the Dashboard)

### 2.1.3.2. Adding and Duplicating Portlets

To add a portlet, select Add portlet (located below the User Information area at the top right of the Dashboard). From the Add Portlet dialog, you can add a portlet or restore portlets to the default view.

Your Dashboard can display more than one of the same portlet type. You might want to display duplicate portlets, for example, to get at-a-glance information about more than one device location that appears in the Google Maps portlet.

### 2.1.4. Network Map

The Network Map represents your network's Layer 3 topology. From the map, you can quickly determine the status of each device by its highlighted color.

To access the network map, select Infrastructure, and then select Network Map.

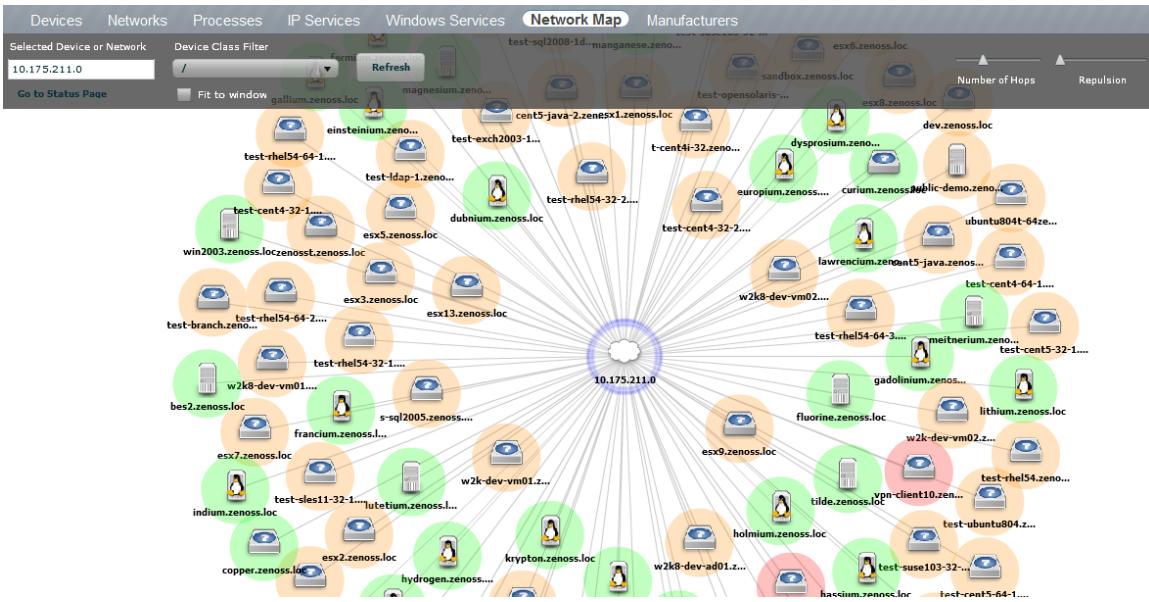


Figure 2.6. Network Map

### 2.1.4.1. Choosing the Network to Display

The network displayed is configured for each user. From user preferences, modify Network Map Start Object to indicate a network, and then click **Save**.

### 2.1.4.2. Viewing Device and Network Details

Double-click a device or network icon in the map to focus on it. Focusing on a node:

- Centers it on the map
- Shows links from the node, based on the number of hops selected

Alternatively, you can type the name or IP address of a device or network in the Selected Device or Network field, and then click **Refresh** to focus on that node.

#### Note

When you select a node, the network map displays only the links that are currently loaded into the map. It does not download and display new link data.

### 2.1.4.3. Loading Link Data

To load link data for a node:

1. Double-click the node on the map to focus on it, or enter the device name or IP address in the Selected Device or Network field.
2. Select the number of hops to download and display.
3. Click **Refresh**.

### 2.1.4.4. Filtering by Device Type

You can filter the devices that appear on the network map. To do this, select a filter from the Device Class Filter list of options. For example, to show only Linux devices on the map, select /Server/Linux from the list of options, and then click **Refresh**.

### 2.1.4.5. Adjusting Viewable Hops

You can adjust the number of hops that appear on the network map. Use the Number of Hops slider, which adjusts the number of hops from 1 to 4.

### 2.1.4.6. Adjusting the Network Map

Use the Repulsion slider to expand or contract the icons that appear on the map. Move the slider right to expand the icons, or left to contract them.

Select the Fit to Window option to bring all displayed icons into the viewable area.

### 2.1.4.7. Viewing Device or Network Details

To see detailed information about a device or network, select it in the map, and then click **Go to Status Page**.

## 2.2. Customizing the Dashboard

You can customize the Dashboard by:

- Selecting the portlets you want to monitor
- Arranging portlets
- Changing the Dashboard column layout

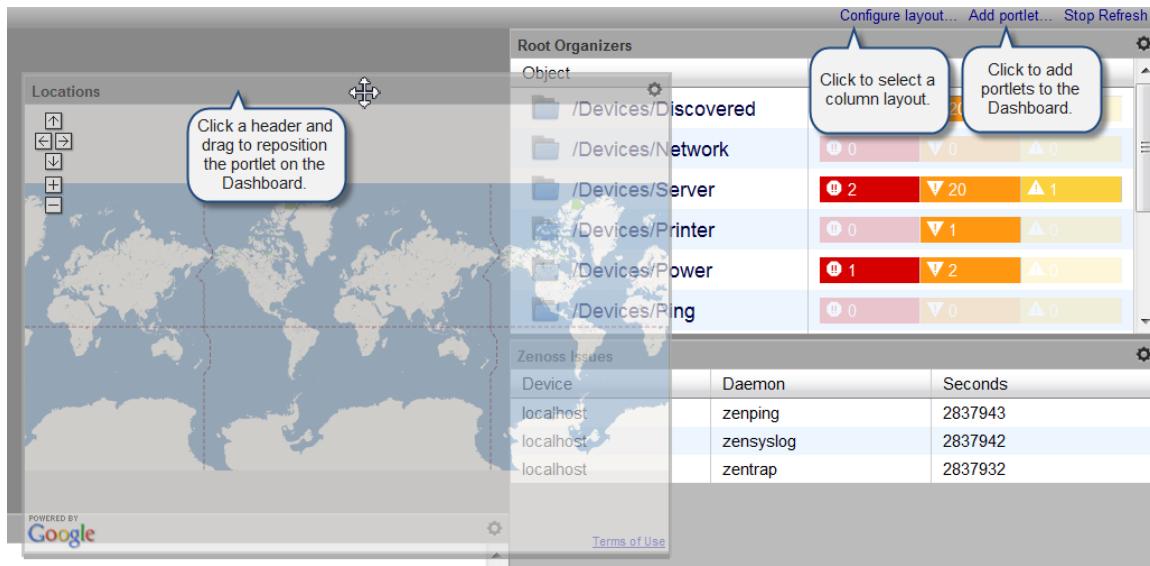


Figure 2.7. Customize Dashboard

### 2.2.1. Selecting Portlets

To add a portlet to the Dashboard:

1. Click **Add portlet** (located at the top right of the Dashboard main area).

The Add Portlet dialog appears.

2. Select a portlet.

The portlet appears at the top right of the Dashboard main area.

To remove a portlet from the Dashboard:

1. Click \* (asterisk) that appears at the top right corner of the portlet you want to remove.

The portlet expands to show its Settings area.

2. Click **Remove Portlet**.

## 2.2.2. Arranging Portlets

To arrange portlets, click the portlet header and drag the portlet to any location on the Dashboard. Other portlets rearrange depending on the location you drop it.

## 2.2.3. Changing the Dashboard Column Layout

You can change the layout of the Dashboard to one, two, or three-column displays. For two-column display, you can additionally choose a layout that offers columns of equal or varying widths.

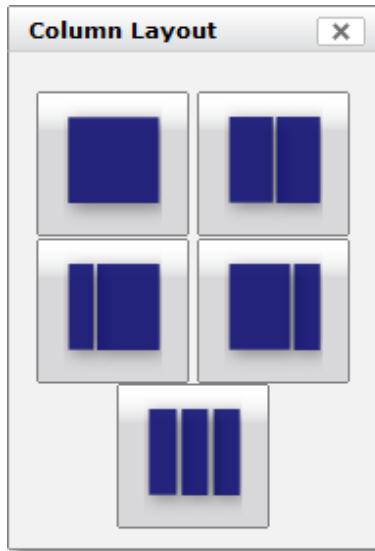


Figure 2.8. Column Layout Dialog

To change the Dashboard column layout:

1. Click Configure layout... (located at the top right of the Dashboard main area).  
The Column Layout dialog appears.
2. Click to select your preferred column layout.

### Note

After selecting a new layout, you likely will need to rearrange the portlets on the Dashboard.

## 2.3. Search

A powerful search facility, enabled by the Advanced Search Enterprise ZenPack, allows you to locate devices, other system objects, and events.

Enter part or all of a name in the search box at the top right of the interface. The system displays matches, categorized by type.

The screenshot shows a search results page with a search bar at the top containing 'test'. Below the search bar, there are two tabs: 'admin-guide' and 'SIGN'. The main area displays a list of search results categorized by type. A 'Device' category is highlighted in yellow, containing five entries:

Device	Details
test-aix53-1.zenoss.loc	(10.20)
test-aix61.zenoss.loc	(10.204)
test-cent5-64-1.zenoss.loc	(1)
test-exch2003-1d.zenoss.loc	
test-exch2007-1d.zenoss.loc	

Other categories visible include 'Host', 'Container', 'Group', 'User', and 'Event'.

Figure 2.9. Search

Click "View all search results" to display the Search Results page, which shows all results of the search. From here, you can save the search to access later. To save a search:

1. Click **Save As** (at the bottom left of the Search Results page).

The Save Search As dialog appears.

2. Enter a name for the search, and then click **Submit**.

You can access saved searches from:

- Action menu located at the bottom of the Search Results page
- Search box located at the top of the interface. (Click the arrow, and then select Manage Saved Searches.)

For information about installing the Advanced Search ZenPack, refer to *Zenoss Extended Monitoring*.

## 2.4. Navigating the Event Console

The event console is the system's central nervous system, enabling you to view and manage events. It displays the repository of all events that are detected by the system.

To access the event console, click Event Console in the Navigation menu.

## Using Zenoss

The screenshot shows the Zenoss Event Console interface. At the top, there are buttons for Event Console, History, Event Classes, and Event Manager. Below these are filters for Status, Severity, Device, and Component. A 'Configure' dropdown menu is open, showing options like 'Show severity row colors' (unchecked), 'Enable live search' (checked), 'Clear filters', 'Save this configuration...', and 'Restore defaults'. The main area displays a table of events with columns for First Seen, Last Seen, and Count. The events listed include various system logs and errors from different hosts and components.

Figure 2.10. Event Console

### 2.4.1. Sorting and Filtering Events

You can sort and filter events that appear in the event console to customize your view.

You can sort events by any column that appears in the event console. To sort events, click a column header. Clicking the header toggles between ascending and descending sort order.

Filter options appear below each column header.

This screenshot shows the Zenoss Event Console with filter options applied. The 'Configure' dropdown is open, showing 'New' selected under the 'Status' filter. Other filters shown are 'Acknowledged' and 'Suppressed'. The main table lists the same events as Figure 2.10, but the results are likely sorted by the 'First Seen' or 'Last Seen' column due to the filter applied.

Figure 2.11. Event Console Filter Options

You can filter the events that appear in the list in several ways, depending on the field type. Date fields (such as First Seen and Last Seen) allow you to enter a value or use a date selection tool to limit the list. For other fields, such as Device, Component, and Event Class, enter a match value to limit the list.

The Count field allows you to filter the list when compared to a value:

- $n$  - Displays events with counts greater than or equal to that value.

- <*n* - Displays events with counts less than that value.
- <= *n* - Displays events with counts less than or equal to that value.
- = *n* - Displays events with counts equal to that value.

To clear filters, select **Configure > Clear filters**.

#### 2.4.1.1. Saving a Custom View

You can save your custom event console view by bookmarking it for quick access later. To do this:

1. Select **Configure > Save this configuration**.

A dialog containing a link to the current view appears.

2. Click and drag the link to the bookmarks link on your browser's menu bar.

The system adds a link titled "Event Console" to your bookmarks list.

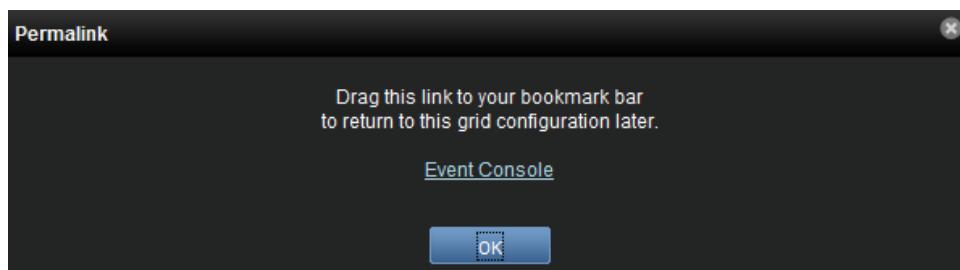


Figure 2.12. Saving a Custom View (Bookmark)

**Tip:** You may want to re-title the bookmark, particularly if you choose to save more than one event console view.

#### 2.4.2. Refreshing the View

You can refresh the list of events manually or specify that they refresh automatically. To manually refresh the view, click **Refresh**. You can manually refresh at any time, even if you have an automatic refresh increment specified.

To configure automatic refresh, select one of the time increments from the Refresh list. By default, automatic refresh is enabled and set to refresh each minute.

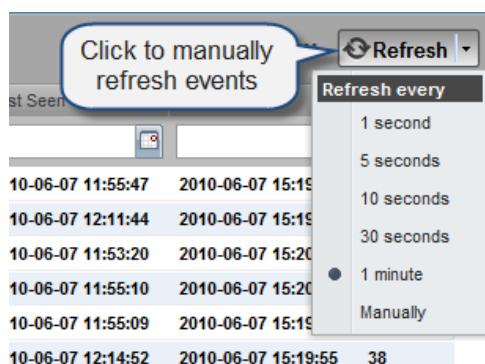


Figure 2.13. Automatic Refresh Selections

#### 2.4.3. Viewing Event Details

You can view details for any event in the system. To view details, double-click an event row.

Tip: Do not double-click on or near the device name, component, or device class in the row. Doing this displays details about that entity, rather than information about the event.

The Event Detail area appears.

localhost zenwin heartbeat failure

**Device:** zenosst.zenoss.loc [?](#)

**Component:** zenwin [?](#)

**Event Class:** /Status/Heartbeat [?](#)

**Status:** 0

**Start Time:** 2009/09/08 10:22:53.000

**Stop Time:** 2009/09/08 14:02:35.000

**Count:** 219

Show more details... [Click for more event details.](#)

LOG [ADD](#)

Figure 2.14. Event Detail

To see more information about the event, click Show more details.

You can use the Log area to add specific information about the event. Enter details, and then click **Add**.

#### 2.4.4. Selecting Events

To select one or more events in the list, you can:

- Click a row to select a single event
- Ctrl-Click rows to select multiple events, or Shift-Click to select a range of events
- Click Select to select all, none, new, acknowledged, or suppressed events

#### 2.4.5. Managing Events

You can manage events from the event console. After selecting an event, you can:

- Acknowledge the event
- Close the event (move it to history)
- Map the event, associating it with a specific event class
- Return the event to New status (revoke its Acknowledged status)

You also can add an event from the event console.

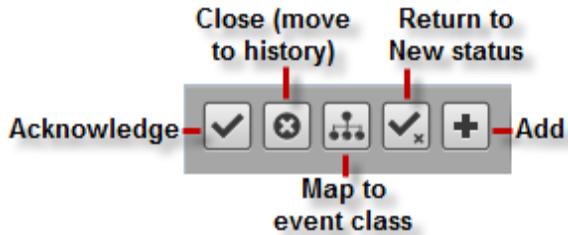


Figure 2.15. Event Management Options

## 2.5. Running Commands

Zenoss allows commands to be run through the Web-based user interface. You can run commands on a single device or on a group of devices.

The system includes several built-in commands, such as ping and traceroute.

To run commands from the user interface:

1. Select one or more devices from the Devices list.
2. Select a command from the Commands list of options.

The system runs the command. Command output appears on the screen.

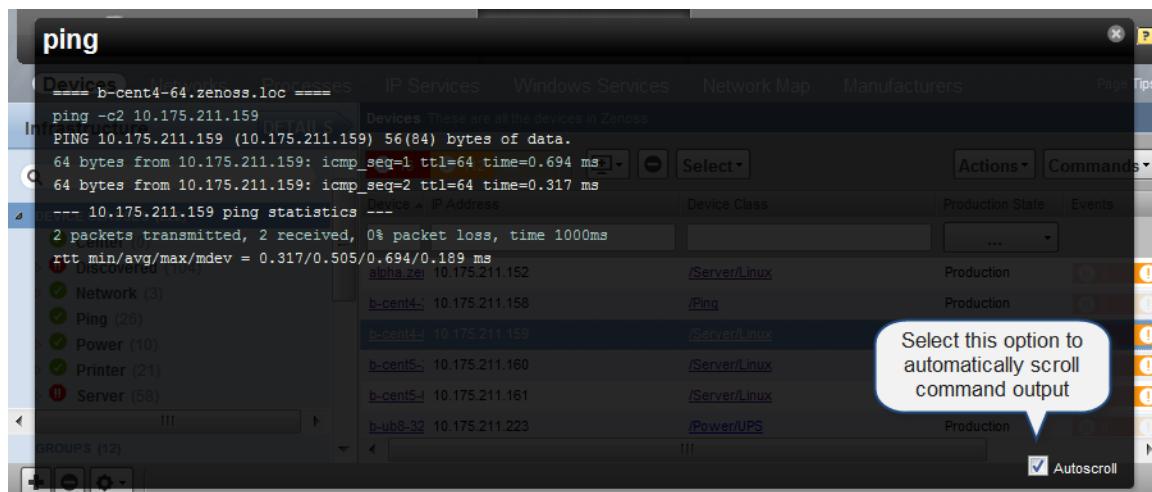


Figure 2.16. Command Output

You can resize the command output window. You also can stop automatic scrolling by de-selecting the Autoscroll option at the bottom right corner of the output window.

## 2.6. Creating and Using Alerts

You can implement *alerts* to send email or pages based on events received. Implemented by the `zenactions` daemon, the system continuously evaluates each user's paging rules against the event database. Each user has his own set of alerting rules.

Read the following sections to learn about:

- Setting SMTP settings for alerts
- Creating alerting rules
- Escalating alerts
- Scheduling alerts

### 2.6.1. Setting SMTP Settings For Alerts

To use email and pager alerts, the system must point to an SMTP relay with the proper settings.

1. From the navigation bar, select Advanced.

The Settings page appears.

The screenshot shows the Zenoss Settings page with the following configuration:

- SMTP Host:** localhost
- SMTP Port (usually 25):** 25
- SMTP Username (blank for none):** (empty)
- SMTP Password (blank for none):** (empty)
- From Address for Emails:** (empty)
- Use TLS?**:
- Page Command:** \$ZENHOME/bin/zensnpp
- 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

Figure 2.17. SMTP Settings

- To set up the mail servers, you must configure the SMTP Host, the SMTP Port, SNPP Host, and the SNPP Port.

Now you can create and use alerting rules for the system.

## 2.6.2. Creating an Alerting Rule

Alerting rules are created for each user. You can add more recipients for rules, but upon creation, the alerting rules are tied to a user account.

To create an alerting rule:

- Select Advanced from the menu bar.

The Settings page appears.

- Select Users in the left panel to display the users page.
- In the Users area, the name of the user for which you want to create an alerting rule.
- In the left panel, select Alerting Rules.
- Select Add Alerting Rule from (Action Menu).
- From the Alerting rule table menu, select Add Alerting Rule.

The Add Alerting Rule dialog appears.



Figure 2.18. Add Alerting Rule

7. Enter a name for the alerting rules, and then click **OK**.
8. Click the name of the alert.

The Alert Details page appears.

Figure 2.19. Edit Alert Details

### 2.6.2.1. Define and Enable the Alert

Set the attributes from the Alert Details page:

- **Delay (secs)** - Sets the number of seconds to wait before sending the alert. If an event clears before delay time, no alert is sent.
- **Enabled** - To enable the alert, set this value to True. Setting this to True disables all alert windows, and is the same as a 24x7 alerting window.

To alert only during certain periods specified in the alerting windows, set Enabled to False.

To ensure that an alerting rule will not send alerts, ensure that Enabled is set to False, and that all alerting rule windows are not enabled.

- **Repeat Time** - Sets the time for repeating the alert. Sends the alert every *n* seconds until the event is acknowledged.
- **Action** - Selects whether to send email or a page.

If the action is defined as email, then the event is emailed. If the action is set to page, then you must define and test the value of the Page Command field (Advanced > Settings). Many wireless phone systems have SMTP-to-SMS gateways, so in some cases you can use email to send pages.

By default, email alerts are sent to the email address for the user. Pager alerts go to the specified pager address. You can override this by filling in the Address (optional) field.

- **Where** - This area lets you set the thresholds for the alert. The default rule contains the thresholds for an event occurrence in which:

- Event State = New
- Severity >= Error
- Production State = Production

You can change these thresholds by changing one or more selections.

- **Add filter** - Select one or more filters for the alert. The page re-displays so you can select values for the filter. To remove a filter, click the (-) minus button.

Click **Save** to save selections for the alert.

### 2.6.2.2. Create the Content of the Alert Message

1. In the left panel, select Message to customize the message that is sent to the specified address.

The Message page appears.

State at time: 2010/04/16 17:18:08 |

**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:
```

**Save**

Figure 2.20. Alerting Rules - Messages

2. Specify the email message subject and body. You must create two messages:

- **Message** - Text to send when the thresholds for the alert are met or exceeded.
- **Clear Message** - Text to send when the event has cleared.

The fields for the subject and message areas are Python format strings.

3. Click **Save** to save the messages.

### 2.6.2.3. Create a Schedule for Sending the Alert

By default, all enabled schedules are active at all times. If you want to restrict the times for which an alerting rule is active, follow these steps:

1. Set the Enabled alert field to False.
2. From the left panel, select Schedule to set up a schedule for the alert.

The Active Periods page appears.

3. Select Add Rule Window from the Actions menu.

The Add Active Period dialog appears.

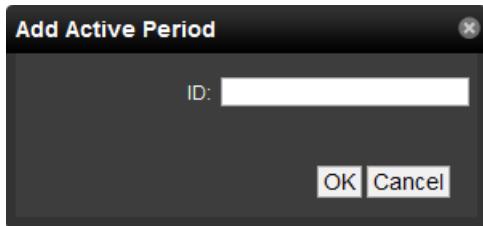


Figure 2.21. Add Active Period

4. Enter a name for the schedule in the ID field, and then click **OK**.

The Schedule you added appears in the Active Periods list.

5. Click the name of the new schedule to set its details.

The Schedule Details page appears.

State at time: 2010/04/16 17:12:12	
Name	test_add_active_period_2
Enabled	False
Start	04/16/2010
Duration	0 Days 01 Hours 00 Minutes
Repeat	Never
<b>Save</b>	

Figure 2.22. Alerting Rules - Schedule Details

6. Enter information or make selections:

- **Enabled** - Set to a value of true if you want to restrict this alert to monitor only at certain times, for certain durations.
- **Start** - Enter the date you want the alert to start, or click Select to select the date from a calendar. Select the hour and minutes you want the alert to start.
- **Duration** - Specify the length of time you want the alert to be listening, based on the start time.
- **Repeat** - Optionally select a time frame for the alert to repeat. You can choose:
  - Never
  - Daily
  - Every Weekday
  - Weekly
  - Monthly
  - First Sunday of the Month

7. Click **Save**.

### 2.6.3. Escalation of Alerting

You can create an alerting hierarchy by using some of the different management tools.

#### 2.6.3.1. Creating an Alerting Hierarchy

You can create an alert hierarchy based on event status and delays by using alerting rules.

### **Sample Scenario:**

You want to set up alerting rules so if that "Person A" (the first person in the hierarchy responding to alerts) does not acknowledge or suppress an event of a specific priority within a specific length of time (changing the event status), then "Person B" is notified by email to respond.

#### **Step 1: Create an Alerting Rule for the Default Case (Initial State)**

The default case is "when any new event of any priority occurs, alert Person A."

Create an alerting rule with a Delay value of 0 (zero) seconds.

#### **Step 2: Create an Alerting Rule for the Next Level**

For the next level in the hierarchy, the case is "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)."

Create an additional alerting rule for Person B. To do this, you can:

- Create an additional rule for the currently logged-in User account.
- Add Person B's email address to the Address field. This address overrides the User account email.

Set the value of Delay to the number of seconds you want to wait after an event has come in to the system but whose status has not changed. In this example, the wait time is one hour (3600 seconds).

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 (including those acknowledged by Person A). For this example, select New.

### **2.6.4. 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 delays will allow you to specify that if an event is not acknowledged in a certain amount of time, then the system should send email to the next person in the hierarchy. You accomplish this by filtering on event state ('New') and adding a delay.

## Example

Create an alerting rule (no delay) for a "tier 1" support representative, so that he is immediately alerted and can acknowledge the event.

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 indicate that this rule is in effect only for events that have not yet 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. Select Message in the left panel. 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. Select Schedule from the left panel 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.

The new schedule appears in the list.

7. Click the name of the new schedule and set these values:

- **Name** - Name of the new schedule.
- **Enabled** - Set to True.
- **Start** - Specify when you want the rule to start.
- **Duration** - Specify how long you want the rule to be in effect.
- **Repeat** - Specify the number of times to repeat the schedule.
- **Every** - Specify how many time periods to repeat.

8. Click **Save**.

## 2.7. Creating Custom Event Views

You can create and edit custom event views, narrowing the event list view according to filters you set and save. Custom event views are set individually for users.

To create a custom event view:

1. From the menu bar, select Advanced, and then select Settings.

2. In the left panel, select Users.
3. Click a user in the Users list.
4. In the left panel, select Event Views.

The Event Views page appears.

5. Select Add Event View from the Action menu.

The Add Event View dialog appears.



*Figure 2.23. Add Event View Dialog*

6. In the ID field, enter a name for the event view.
7. Click **OK**.

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

8. Click the new event view in the list.

The Edit Event View page appears.

*Figure 2.24. Edit Event View*

9. Add conditions for this event view:
  - **Type** - Select whether to show active events or the event history.
  - **Where** - Use this area to add filters (similar to alerting rules "where" clauses).
  - **Order by** - Specify the order of entries in the view.
  - **Result Fields** - Select the fields to display in the view. Click the X next to each field you want to remove from the view.

10. Click **Save**.

# Chapter 3. Adding, Discovering and Modeling Devices

Modeling is the process by which Zenoss:

- Populates the device database
- Collects information about the devices in the system (such as operating system type or file system capacity)

The system models devices when they are added to the database, either manually or through the discovery process.

Read this chapter for more information about:

- Adding devices
- Discovering, classifying, and authenticating devices
- Modeling devices by using SNMP, SSH/COMMAND, and Port Scan
- Working with modeler plugins
- Debugging the modeling process

## 3.1. Adding Devices

You can manually add single or multiple devices.

### 3.1.1. Add a Single Device

Follow these steps to add a single device:

1. From the navigation bar, select Infrastructure.

The Devices page appears.

2. Select Add a Single Device from  (Add Devices).

The Add a Single Device dialog appears.

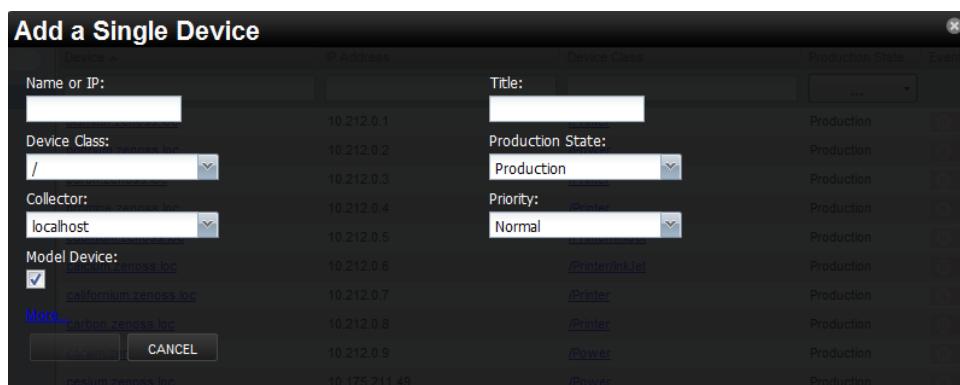


Figure 3.1. Add a Single Device

3. Enter information or make selections to add the device:

- **Name or IP** - Enter the network (DNS) name or IP address of the device.

- **Device Class** - Select a device class to which this device will belong. For example, if the new device is a Windows server, then select /Server/Windows/WMI.
  - **Collector** - By default, this is localhost. Select a collector for the device.
  - **Model** - By default, this option is selected. De-select this option if you do not want the device to be modeled when it is added.
4. Optionally, click More to display additional fields. From the expanded page, you can:
- Enter device-specific details
  - Edit SNMP settings
  - Set hardware and operating system information
  - Add device comments

#### Note

Name or IP, Device Class, and Model are the only required selections when adding a device. You may want to continue (click **Add**) without additional information or selections, as information you add may conflict with information the system discovers about the device.

An exception is if you are adding a Cisco router in a device class other than /Network. In this case, before adding the device you should set the value of the zlfDescription configuration property to True. This will give you additional information about Cisco routers. By default, this option is set to True for the /Network class.

5. Click **Add**.

#### Note

You can view the Add Device job in progress. Click **View Job Log** in the dialog that appears when you add the device.

When the job completes, the device is added in the selected device class.

### 3.1.2. Add Multiple Devices

Follow these steps to manually add multiple devices:

1. From the navigation bar, select Infrastructure.

The Devices page appears.

2. Select Add Multiple Devices from  (Add Devices).

The Add Devices panel appears.

3. For each device you want to add, enter the fully qualified domain name or IP address of a device on your network.
4. In the Details area, select a device type from the list. If your device type is not listed, then use the default selection. (You can choose a different device type after adding the device.)
5. Enter the appropriate credentials used to authenticate against the device.
6. If you want to add more than one device, click **+**. Enter one hostname or IP address on each line. Each device can have only one set of associated credentials.
7. To add the devices, click **Submit**. Zenoss models the devices in the background.

## 3.2. Discovering Devices

You can provide network or IP address range information so that the system can discover your devices.

Follow these steps to discover devices:

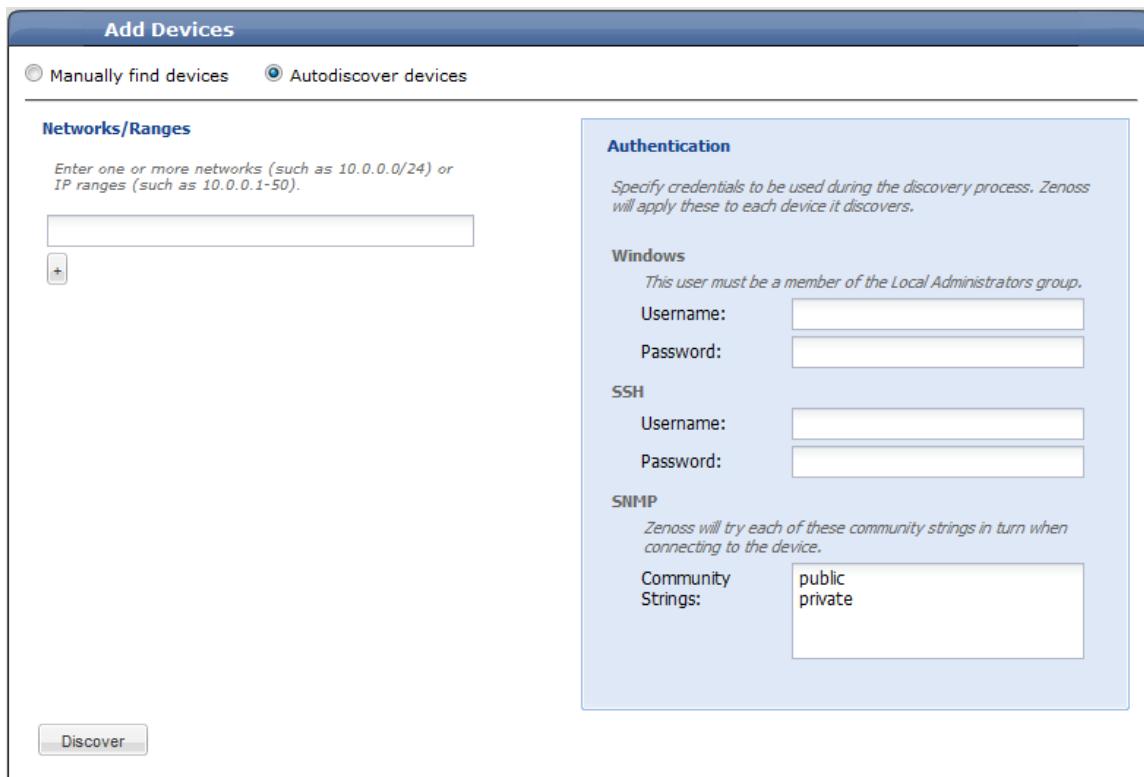
1. From the navigation bar, select Infrastructure.

The Devices page appears.

2. Select Add Multiple Devices from  (Add Devices).

The Add Devices panel appears.

3. Select the **Autodiscover devices** option.



The screenshot shows the 'Add Devices' configuration window. At the top, there are two radio buttons: 'Manually find devices' (unchecked) and 'Autodiscover devices' (checked). Below this, there's a section for 'Networks/Ranges' with a text input field and a '+' button. To the right, under 'Authentication', there are three sections: 'Windows' (Username: [input], Password: [input]), 'SSH' (Username: [input], Password: [input]), and 'SNMP' (Community Strings: [input] containing 'public' and 'private'). At the bottom left is a 'Discover' button.

Figure 3.2. Add Multiple Devices (Discover)

4. For each network or IP range in which you want Zenoss to discover devices, enter an address or range. For example, you might enter a network address in CIDR notation:

10.175.211.0/24

or a range of IP addresses:

10.175.211.1-50

5. If you want to enter multiple addresses or ranges, click +. For each network, you must enter a netmask or IP range.

6. For each network or IP range, specify the Windows, SSH, or SNMP credentials you want Zenoss to use on the devices it discovers. You can enter only one of each. Zenoss will attempt to use the same credentials on each device it discovers within the networks or IP ranges specified, but will not try to automatically classify the devices.
7. Click **Discover**.

The discovery process iterates through every IP address in the networks and IP ranges you specify, adding each device that responds to a ping request. Further, the process adds information to any device that responds to an SNMP, WMI, or SSH request.

Zenoss places discovered routers in the device path /Network/Router. Devices are placed in the /Discovered device class.

### 3.2.1. Classifying Discovered Devices

Once discovery is complete, you must move discovered devices (placed, by default, in the /Discovered class) to an appropriate device class in the hierarchy. Moving devices to their correct hierarchy location makes it possible for monitoring to begin.

Servers are organized by operating system. If the system discovers Windows devices, for example, you might choose to relocate them to /Server/Windows. Similarly, you might choose to classify discovered Linux devices in /Server/Linux (if you want to monitor and model using SNMP), or /Server/SSH/Linux (if you want to monitor and model using SSH).

To classify discovered devices:

1. Select one or more discovered devices (highlight one or more rows) in the device list.
2. Drag the selected devices to the new device class in the tree view.

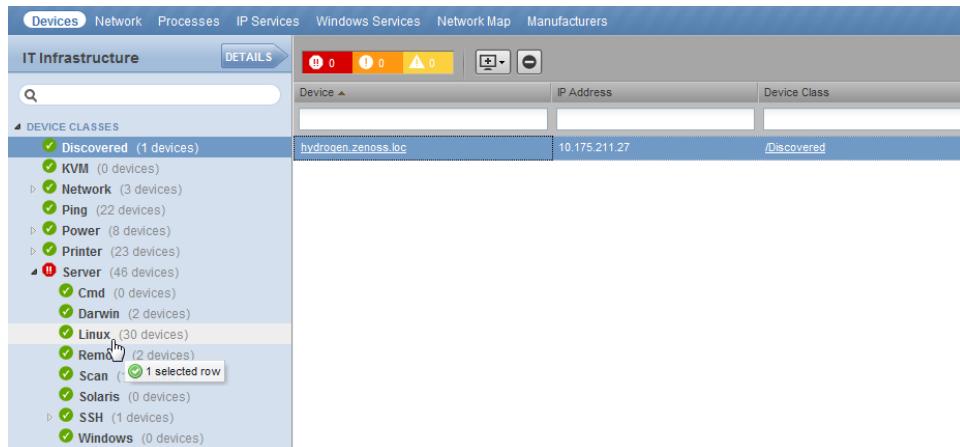


Figure 3.3. Classifying Discovered Devices

The Move Devices dialog appears.

3. Click **OK**.

The list of devices refreshes, and the devices now appear in the newly selected class.

### 3.2.2. Updating Device Authentication Details

For each device added to the database and set to its proper device class, the system may require additional or different authentication information before it can gather device information and monitor the device.

For example, for a device in the /Server/Windows class, you must supply your Windows user name and password before the system can monitor the device. To do this:

1. Click a device name in the devices list.

The Device summary page appears.

2. Select Configuration Properties from the left panel.
3. Set the user name and password values in the zWinUser and zWinPassword configuration properties.
4. Click **Save**.

Similarly, for a device in the /Server/SSH/GenericLinux class, you must supply your SSH user name and password. Set these values in the device's zCommandUsername and zCommandPassword configuration properties.

#### Tip

After making changes, you should remodel the device to ensure the authentication changes are valid.

### 3.2.3. Adding Information to a Device Record

You may want to add details about a discovered device.

To add information:

1. Click a device name in the devices list.

The Device summary page appears. Values that appear in text fields can be edited.

2. Enter or change information in one or more fields, and then click **Save** to save your changes.

## 3.3. Modeling Devices

To model devices, the system can use:

- SNMP
- SSH
- WMI
- Telnet

The modeling method you select depends on your environment, and on the types of devices you want to model and monitor.

By default the system remodels each known device every 720 minutes (12 hours).

#### Tip

You can change the frequency with which devices are remodeled. Edit the value of the Modeler Cycle Interval in the collector's configuration.

For larger deployments, modeling frequency may impact performance. In such environments, you should stop the ZenModeler daemon and run the modeling process once daily from a cron job.

### 3.3.1. Modeling Devices Using SNMP

Read this section for information about the methods Zenoss uses to model devices using SNMP.

#### 3.3.1.1. Testing to See if a Device is Running SNMP

To test whether a device is running SNMP, run this command:

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

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

### 3.3.1.2. Configuring Windows Devices to Provide Data Through SNMP

By default, Windows may not have SNMP installed. To install SNMP on your particular version of Windows, please refer to the Microsoft documentation.

After setting up and configuring the SNMP service, you must set the zSnmpCommunity string in Zenoss to match, to obtain SNMP data.

If you want processor and memory monitoring, install SNMP-Informant on the device. Go to <http://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 the SyslogAgent Windows add-on, available from:

<http://syslogserver.com/syslogagent.html>

### 3.3.1.3. Configuring Linux Devices to Provide Data Through 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 to then use SNMP to monitor Linux devices.

## 3.3.2. Modeling Devices Using SSH/COMMAND

You can gather additional information by running commands on the remote device and interpreting the results. This provides a more scalable and flexible way to gather information that may not be available through any other means.

Each built-in 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.

To model a device using command plugins, first add the device by using the protocol "none," and then choose the plugins you want to apply:

1. From the navigation bar, select Infrastructure.
2. Select Add a Single Device from  (Add Devices).

The Add a Single Device dialog appears.

3. Enter values for Name or IP and Device Class.
4. De-select the Model Device option.
5. Click **Add**.
6. After adding the device, select the device name in the devices list.

The Device summary page appears.

7. In the left panel, select Configuration Properties.
8. If necessary, set the values of the zCommandUsername and zCommandPassword configuration properties to the user name and password of the device (or set up authentication by using RSA/DSA keys.)

### Note

If using RSA keys for a device or device class, change the value of the zKeyPath configuration property to:

```
~/.ssh/id_rsa
```

9. In the left panel, select Modeler Plugins.
10. Click Add Fields for a complete list of command plugins.
11. Drag zenoss.cmd.uname from the Available list (on the right) to the Plugins list (on the left). Place it at the top of the list.

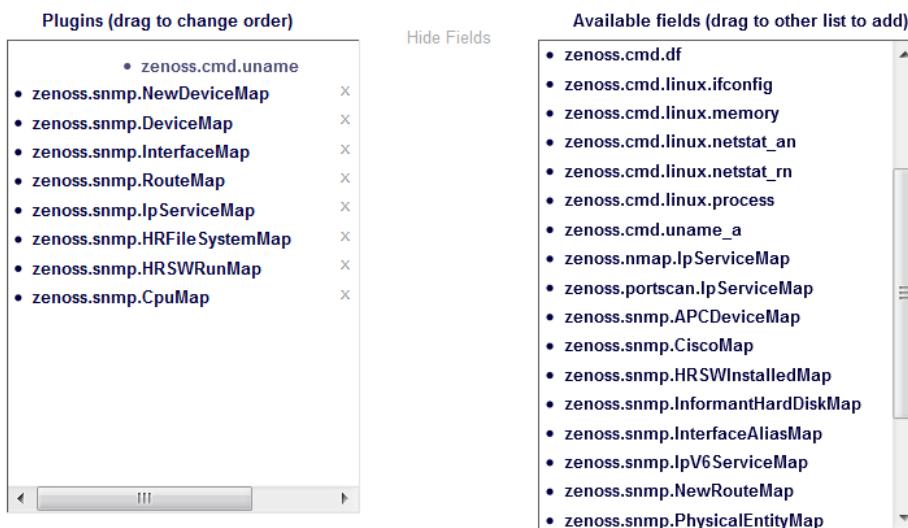


Figure 3.4. Add Plugin

12. Click the X next to plugins you want to remove, and then drag remaining plugins to the left.
13. Click **Save**.
14. Remodel the device.

### 3.3.2.1. 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 zCollectorPlugins have been modified (see the section titled "Modeling Using SSH/Command"), and the device, file system, 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 pre-configured templates and configuration properties. You also must set the zCommandUsername and zCommandPassword configuration properties to the appropriate SSH login information for each device.

### 3.3.3. Modeling Devices Using Port Scan

You can model IP services by doing a port scan, using the Nmap Security Scanner (<http://nmap.org/>). You must provide the full path to your system's nmap command.

To determine where nmap is installed, at the command line, enter:

```
which nmap
```

If your system returns a result similar to:

```
/usr/bin/which: no nmap in
(/opt/zenoss/bin:/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/opt/zenoss/bin)
```

then nmap is not installed. Install it, and then try again.

After locating the `nmap` command (including the directory beginning with `/`), enter the following as the zenoss user on the Zenoss server:

```
cd $ZENHOME/libexec  
ln -s Full_Path_to_nmap
```

To model a device using a port scan:

1. View the device properties (from the left panel, select Configuration Properties).
2. Change the zTransportPreference value to portscan.
3. Remodel the device.

### 3.3.3.1. Using the /Server/Scan Device Class to Monitor with Port Scan

The /Server/Scan device class is an example configuration for modeling devices by 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.

## 3.4. About Modeler Plugins

Zenoss uses plug-in maps to map real world information into the standard model. Input to the plug-ins can come from SNMP, SSH or Telnet. Selection of plug-ins to run against a device is done by matching the plug-in name against the zCollectorPlugins configuration property. 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 network interfaces on a device.
- **RouteMap** – Collects the network routing table from the device.
- **IpServicesMap** – Collects the IP services running on the device.
- **FileSystemMap** – Collects the list of file systems on a device.

### 3.4.1. Viewing and Editing Modeler Plugins for a Device

Plugins are controlled by regular expressions that match their names. To view a list of plugins for any device:

1. Click the device name in the devices list.

The Device summary page appears.

2. In the left panel, select Modeler Plugins.

The Modeler Plugins page appears.

#### 3.4.1.1. Adding Plugins

To add a plugin to a device:

1. Click Add Fields (located to the right of the Plugins list) to display a list of additional plugins.
2. Drag one or more plugins from the Available list (on the right) to the Plugins list.
3. Click **Save**.

#### 3.4.1.2. Reordering Plugins

Plugins run in the order in which they are listed. To re-order plugins, drag them in the Plugins list, and then click **Save**.

### 3.4.1.3. Deleting Plugins from a Device

To delete a plugin from a device, click X next to the plugin in the Plugins list.

## 3.5. Debugging the Modeling Process

You can run the modeler from the command line against a single device. This feature is useful when debugging issues with a plugin.

By passing the **--collect** command to the modeler, you can control which modeler plugins are used. For example, the following command runs only the interface plugin against the build.zenoss.loc device:

```
$ zenmodeler run -v10 --collect=IpInterface -d build.zenoss.loc
```

If the command returns any stack traces, check the community forums for assistance. Otherwise, forward these details to Support for assistance:

- Command you ran
- Stack trace or stack traces returned
- Version of your Zenoss instance
- OS version and patch level for the remote device

# Chapter 4. Working with Devices

This chapter provides information and procedures for managing devices in the system.

## 4.1. Viewing the Device List

The device list shows all devices in the system. From this view, you can search for devices and perform a range of management tasks on all devices.

To access the device list, select Infrastructure from the navigation bar.

The screenshot shows the Zenoss interface for viewing the device list. On the left, there is a tree view of device classes, groups, systems, and locations. The 'DEVICE CLASSES (27)' section is expanded, showing categories like AWS (1), Discovered (0), HTTP (0), KVM (0), Network (3), Ping (2), Power (0), Printer (0), Server (21), Storage (0), VMware (0), and Web (0). Below this, 'GROUPS (0)', 'SYSTEMS (0)', and 'LOCATIONS (0)' are listed. The main area is titled 'Devices' and contains a table with columns: Device, IP Address, Device Class, Production State, and Events. The table lists numerous devices, each with a status icon (e.g., green, yellow, red) and a small number indicating event count. The 'Production State' column shows 'Production' for most devices, while some are marked as 'Testing'. The 'Events' column shows counts ranging from 0 to over 40. The table has a header row and several data rows, each representing a different device entry.

Figure 4.1. Device List

### 4.1.1. Devices Hierarchy

Devices are organized in the tree view by:

- Device class
- Group
- System
- Location

Click the indicator next to each category name to expand it and see included devices.

### 4.1.2. Managing Multiple Devices from the Device List

You can perform some management tasks for more than one device at a time. You can:

- Move devices to a different class

- Assign devices to groups, systems, and locations
- Remove devices
- Perform actions such as assign priority and production state
- Assign monitors for collecting from selected devices
- Lock devices

## 4.2. Working with Devices

To view details for a single device, click its name in the device list. The device overview page appears.

The screenshot shows the Zenoss Device Overview page for the device `colo3560g.zenoss.loc`. The top navigation bar displays the device name and IP address (`10.175.211.2`). Below the navigation bar, there are tabs for Overview, Events, Components, Software, Graphs, Cisco Details, Administration, Configuration Properties, Modeler Plugins, Custom, Modifications, Monitoring Templates, and Device (INetwork/Cisco). The Overview tab is selected. The main content area is divided into several sections:

- Uptime:** 114d:22h:49m:24s
- First Seen:** 2010/06/07 11:57:54
- Last Change:** 2010/06/07 11:58:20
- Model Time:** 2010/06/07 11:58:21
- Locking:** Unlocked
- Memory/Swap:** 92.7MB/Unknown
- Device Name:** colo3560g.zenoss.loc
- Production State:** Production
- Priority:** Normal
- Collector:** localhost
- Systems:** None
- Groups:** None
- Location:** None
- Tag:** (empty input field)
- Serial Number:** FOC1220Z4TX
- Rack Slot:** 0
- Hardware Manufacturer:** Cisco
- Hardware Model:** .1.3.6.1.4.1.9.1.617
- OS Manufacturer:** Unknown
- OS Model:** IOS 12.2(35)SE5
- SNMP SysName:** colo3560g
- SNMP Location:** CoLo
- SNMP Contact:** it@zenoss.com
- SNMP Agent:** (empty input field)

At the bottom of the page are buttons for **SAVE** and **CANCEL**.

Figure 4.2. Device Overview

Event status is shown in the "event rainbow" at the top of the page. Other key information that appears at the top of the device overview page includes:

- Device name
- IP address used to communicate with the device
- Device status (shows the current results of a ping test)
- Production state (Pre-Production, Production, Test, Maintenance, or Decommissioned)

When you open the page, device overview information displays. This view provides classification and status information. From here, you can edit device information (indicated by text fields or edit links).

The Links area displays links between the device and other external systems. For more information about custom links, see the chapter titled "Properties and Templates."

The left panel of the device overview page allows you to access other device management views, such as:

- Events

- Components
- Software
- Graphs
- Administration
- Configuration Properties
- Modeler Plugins
- Custom
- Modifications
- Monitoring Templates

Information that appears here varies depending on device type.

## 4.2.1. Events

Detailed information about events, scoped to the device, appears in the Events view. From here, you can:

- Sort event information by a range of categories
- Classify and acknowledge events
- Filter events by severity, state, or by one of several categories

## 4.2.2. Components

The Components view provides information about the different types of device components, including:

- IPService
- WinService
- IpRouteEntry
- IplInterface
- CPU
- FileSystem

To access components information, select Components in the left panel, and then select a component type.

	IP Address	Network	MAC Address	Status	Monitored	Locking
GigabitEthernet0/1	00:21:A1:3B:DF:81			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/10	00:21:A1:3B:DF:8A			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/11	00:21:A1:3B:DF:8B			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/12	00:21:A1:3B:DF:8C			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/13	00:21:A1:3B:DF:8D			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/14	00:21:A1:3B:DF:8E			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/15	00:21:A1:3B:DF:8F			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/16	00:21:A1:3B:DF:90			Up <span style="color: green;">●</span>	true	
GigabitEthernet0/17	00:21:A1:3B:DF:91			Up <span style="color: green;">●</span>	true	

Figure 4.3. Device (Components)

The status of each device component type, as shown by the color of its indicator, is determined by the collective status of the monitored components of the same type. For example, if the IPServices status is green, then all monitored

IpServices on the device are functioning normally. If there is an event related to a monitored IpService, then the highest severity event associated with that component is displayed.

### Note

If there is an event unrelated to a known component, then the system places it in the component type Other.

From this view, you can:

- Lock components
- Turn on or off component monitoring
- Delete components

## 4.2.3. Software

The Software view lists software installed on the device. The details provided in this area depend on the method used to model the device.

Listed software links into the system's inventory of software in your IT infrastructure. You can view this inventory from the Manufacturers link on the sub-navigation menu.

To access software information, select Software in the tree view.

## 4.2.4. Graphs

The Graphs view shows performance graphs defined for the device. To access graphs, select Graphs in the left panel.

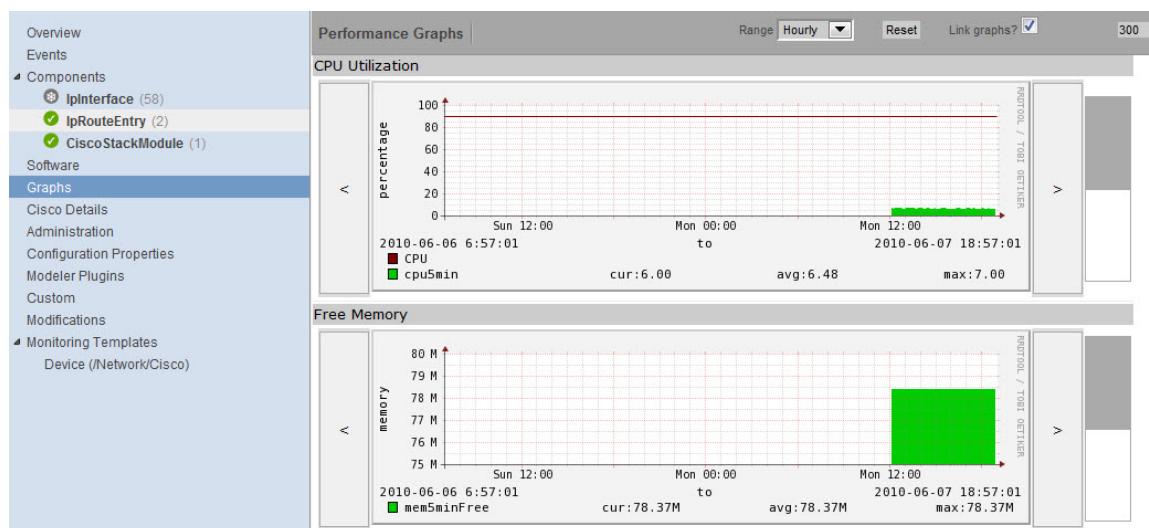


Figure 4.4. Device (Graphs)

### Tip

You can use the arrow key and magnifying glass controls on the sides of each graph to change the graph view, scrolling through or zooming in or out of a graph.

You can control these performance graph options:

- **Range** - Select the span of time displayed in the graph. You can select:
  - Hourly - Past 36 hours
  - Daily - Past ten days
  - Weekly - Past six weeks
  - Monthly - Past 15 months
  - Yearly - Past two years
- **Reset** - Click to return to the default (initial view) of the graphs.
- **Link graphs** - By default, all graphs move together. If you click the back arrow for a graph, for example, then all graphs move backward. De-select the Link graphs option to control each graph individually.
- **Stop/Start** - Toggle to turn off and on automatic refresh of the graphs. Optionally, modify the refresh value (by default, 300 seconds), and then click **Stop/Start** to begin refreshing graphs at the new interval.

For more information about performance monitoring and performance graphs, see the section titled "Performance Monitoring."

## 4.2.5. Administration

Use the Administration view to:

- Create custom user commands and run commands
- Manage maintenance windows
- Determine who holds administration capabilities for the device, and their roles

To access administration options, select Administration in the left panel.

Name	Description	Command	Modifications
DNS_forward	Name to IP address lookup	host \${device/d}	
DNS_reverse	IP address to name lookup	host \${device/manageip}	
ping	Is the device responding to ping?	ping -c2 \${device/manageip}	
snmpwalk	Display the OIDs available on a device	snmpwalk -S\${device/zSnmpVer} -c\${device/zSnmpCommunity} \${here/manageip} system	
traceroute	Show the route to the device	traceroute -q 1 -w 2 \${device/manageip}	

Name	Start	Duration	Repeat	Start State	Stop State	Enabled?	Modifications

Name	Role	Level	Email	Pager

Figure 4.5. Device (Administration)

See these topics for more information about device administration tasks:

- "Defining User Commands" in the chapter titled User Commands
- "Adding Administrators" in the chapter titled Managing Users

## 4.2.6. Configuration Properties

From the Configuration Properties view, you can configure certain configuration properties for a device. Further, you can delete local properties from a device.

To access configuration properties, select Configuration Properties in the left panel.

Property	Value	Type	Path
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/Network/Cisco
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		password	/
zCommandPath	\$ZENHOME/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/

Figure 4.6. Device (Configuration Properties)

For detailed information about working with configuration properties, see the chapter titled "Properties and Templates."

### 4.2.7. Modeler Plugins

Use the Modeler Plugins view to manage plugins that are run against a device. To access plugins, select Modeler Plugins in the left panel.

Name: zCollectorPlugins	Path: /Network/Cisco
Plugins (drag to change order)	
<ul style="list-style-type: none"> <li>• zenoss.snmp.NewDeviceMap</li> <li>• zenoss.snmp.DeviceMap</li> <li>• zenoss.snmp.CiscoMap</li> <li>• zenoss.snmp.InterfaceMap</li> <li>• zenoss.snmp.HSRPMap</li> <li>• zenoss.snmp.RouteMap</li> <li>• RTTProbes</li> <li>• SLBVirtualServers</li> <li>• zenoss.snmp.CiscoStackModuleMap</li> </ul>	
<input type="button" value="Save"/> <input type="button" value="Delete"/>	
<input type="button" value="Hide Fields"/>	
<b>Available fields (drag to other list to add)</b> <ul style="list-style-type: none"> <li>• BrocadeDeviceMap</li> <li>• BrocadeFCPorts</li> <li>• CheckPoint.DeviceMap</li> <li>• DellCPUMap</li> <li>• DellDeviceMap</li> <li>• DellPCIMap</li> <li>• FilerFileSystemMap</li> <li>• HPCPUMap</li> <li>• HPDeviceMap</li> <li>• JuniperCardMap</li> <li>• JuniperDeviceMap</li> <li>• JuniperNoCommunityRouteMap</li> <li>• JuniperNoCommunityRoutePrep</li> <li>• NetAppDeviceMap</li> <li>• NetScreenVPN</li> <li>• NortelInterfaceMap</li> </ul>	

Figure 4.7. Device (Modeler Plugins)

### 4.2.8. Custom

Use the Custom view to edit the values of custom properties applied to a device.

To access custom device properties, select Custom in the left panel.

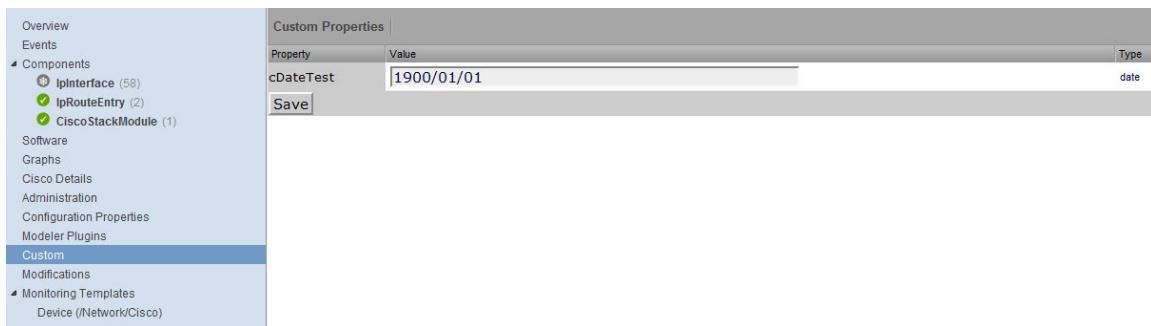


Figure 4.8. Device (Custom Properties)

You cannot define custom properties on an individual device. See the section titled "Adding Custom Properties" for more information.

## 4.2.9. Modifications

The Modifications view shows changes made to a device. The list shows when the change was made and the user who made the change. To access this information, select Modifications in the left panel.

## 4.2.10. Monitoring Templates

Monitoring templates determine how the system collects performance data for devices and device components.

To access monitoring templates, expand Monitoring Templates in the left panel, and then select Device. The page shows all of the monitoring templates that are bound by name to this device.

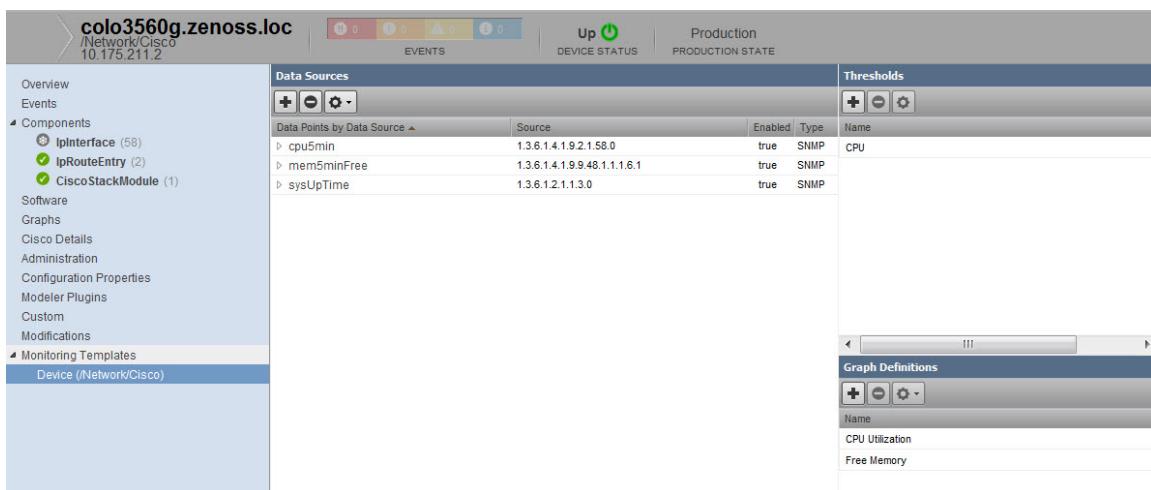


Figure 4.9. Device (Monitoring Templates)

For detailed information about monitoring templates, go to the section titled "Performance Monitoring."

## 4.3. Managing Devices and Device Attributes

Read the information and procedures in this section to learn about specific device management tasks, including:

- Clearing heartbeat events

- Pushing configuration changes to the system
- Locking device configuration
- Renaming devices
- Remodeling devices
- Setting the device IP manage address

### 4.3.1. Clearing Heartbeat Events

If you have configured a device to send a recurring event that you have mapped to a heartbeat class, you can clear stale heartbeat events.

To clear the heartbeat events associated with a device:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Clear Heartbeats from  (Action Menu).  
The Clear Heartbeats dialog appears.
3. Click **Submit** to confirm the clear action.

The system moves the heartbeat events for the device to event history and displays a confirmation message of the action.

### 4.3.2. Pushing Configuration Changes to Zenoss

When you make a configuration change, it is automatically propagated to all the remote collectors. If you think that your change has not been propagated correctly, then you can manually force a configuration "push" to the collectors.

To push configuration changes:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Push Changes from  (Action Menu).  
The Push Changes dialog appears.
3. Click **Yes**.

The system pushes changes to the collectors and displays a confirmation message of the action.

### 4.3.3. Locking Device Configuration

You can lock a device's configuration to prevent changes from being overwritten when remodeling the device. Two levels of locking are available. You can lock the configuration from deletion and updates, or solely from deletion.

#### Note

Device locking prevents changes and deletion due to remodeling. It does not prevent manual changes and deletion.

To edit lock selections for a device configuration:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Lock from  (Action Menu).

The Lock dialog appears.

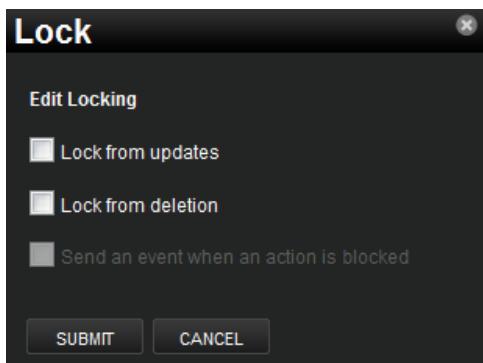


Figure 4.10. Edit (Configuration) Lock Dialog

3. To send events when actions are blocked by a lock action, select the "Send an event..." option.
4. Select the type of lock you want to implement or remove.

The lock or unlock action is implemented on the device, and the system displays a confirmation message of the action.

#### 4.3.4. Renaming a Device

Because the system uses the manage IP to monitor a device, the device name may be different than its fully qualified domain name (FQDN). The device name must always be unique.

To rename a device:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Rename Device from (Action Menu).

The Rename Device dialog appears.

3. Enter the new name for the device, and then click **Submit**.

The system renames the device and displays a confirmation message of the action.

#### 4.3.5. Remodeling a Device

Remodeling forces the system to re-collect all configuration information associated with a device. Normally, the system models devices every 720 minutes; however, if you want to remodel a device immediately, follow these steps:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Model Device from (Action Menu).

The system remodels the device. A dialog appears that shows progress of the action.

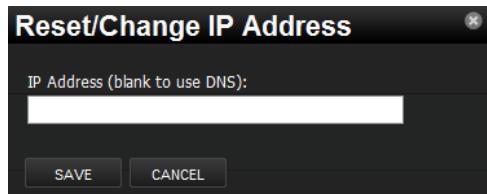
#### 4.3.6. Resetting the Device Manage IP Address

You might want to reset the manage IP address if the IP address of a device has changed and you want to maintain the historical data at the original IP address. To reset the manage IP address of a device:

1. Navigate to the device in the device list.

2. At the bottom of the device overview page, select Reset/Change IP Address from  (Action Menu).

The Reset IP dialog appears.



*Figure 4.11. Reset IP Dialog*

3. Enter the new IP address for the device, or leave the field blank to allow the IP address to be set by DNS.
4. Click **Save**.

The IP address for the device is reset.

#### 4.3.7. Resetting the Device Community

If the system is unable to monitor a device because its SNMP community has changed, you can re-discover the device community by using the list of community strings defined in the zSnmpCommunity configuration property.

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Reset Community from  (Action Menu).

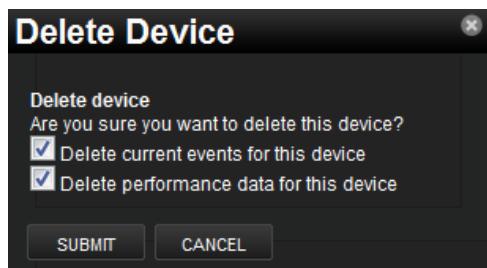
The community for the device is reset.

#### 4.3.8. Deleting a Device

To delete a device from the system:

1. Navigate to the device in the device list.
2. At the bottom of the device overview page, select Delete Device from  (Action Menu).

The Delete Device dialog appears.



*Figure 4.12. Delete Device*

3. Optionally change the selections to delete current events or performance data for the device. By default, events and performance data are removed.
4. Click **Submit**.

The system removes the devices and associated data (if selected), and displays a confirmation message of the action.

#### 4.3.9. Dumping and Loading Devices Using XML

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

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

This command writes the names of your devices (including their device classes, groups, systems) to a file named `mydevicelist.xml`.

To load these devices into another 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
```

The system attempts to discover each of the devices in the XML file.

# Chapter 5. Properties and Templates

Read this chapter to learn more about:

- Configuration properties
- Monitoring templates

## 5.1. Configuration Properties

Configuration properties are individual values you can set up on major system entities, such as:

- **Devices**

*Device configuration properties* control the way devices are monitored.

- **Events**

*Event configuration properties* control the rules that process events as they are received by the system.

- **Networks**

*Network configuration properties* control options when you perform network discovery.

Configuration properties and values can be added to the ZenPacks you create, allowing you to customize the system when you add ZenPacks.

### 5.1.1. Configuration Properties Inheritance and Override

The following diagram illustrates a portion of the standard device class hierarchy. (A *device class* is a special type of organizer used to manage how the system models and monitors devices.)

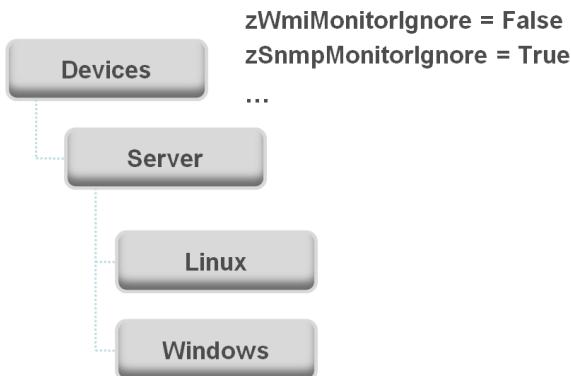


Figure 5.1. Device Class Hierarchy

At the root of the device hierarchy is the Devices object. All device class configuration properties are defined here. Their values are the default values for the entire hierarchy.

The illustration shows two defined configuration properties:

- **zWmiMonitorIgnore** - Turns off all daemons that use WMI. By default, its value at the root of the hierarchy is False.
- **zSNMPMonitorIgnore** - Turns off all daemons that use SNMP. By default, its value at the root of the hierarchy is True.

Through *inheritance*, properties defined at the root of the hierarchy apply to all objects beneath that node. So, at the /Devices/Server/Linux level of the device class hierarchy, the value of these two properties is the same as at /Devices, even though the property is not set explicitly at /Devices/Server/Linux. Inheritance simplifies system configuration, because default values set at the root level apply to all devices irrespective of their device class.

To further customize the system, you can change a specific configuration property further down the hierarchy without having to change the definitions of other configuration properties. As shown in the following illustration, the value of zWmiMonitorIgnore is changed so that WMI monitoring is performed at the /Devices/Server/Windows level.

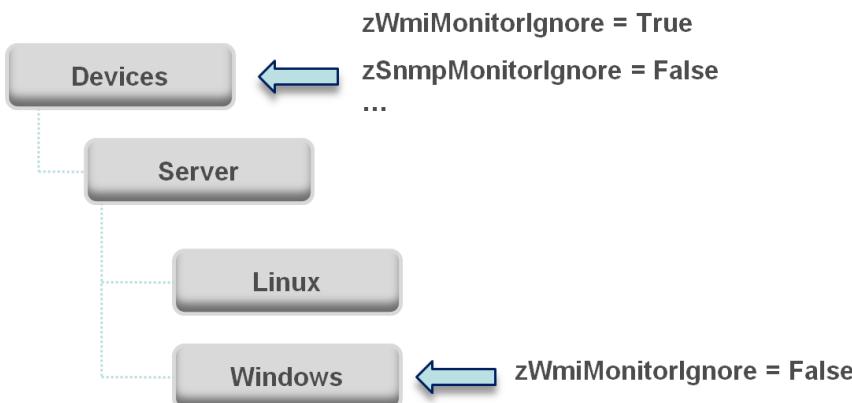


Figure 5.2. Device Class Hierarchy - Locally Defined Value (Override)

This locally defined value for zWmiMonitorIgnore *overrides* the value set at the root of the hierarchy. No other properties at this level are affected by this local change; they continue to inherit the value set at the root.

Configuration properties allow you to configure the system at a very granular level, down to a particular device. For example, in the following illustration, the device named dev.zenoss.com has the value of SNMP community set to private. This overrides the root value (public).

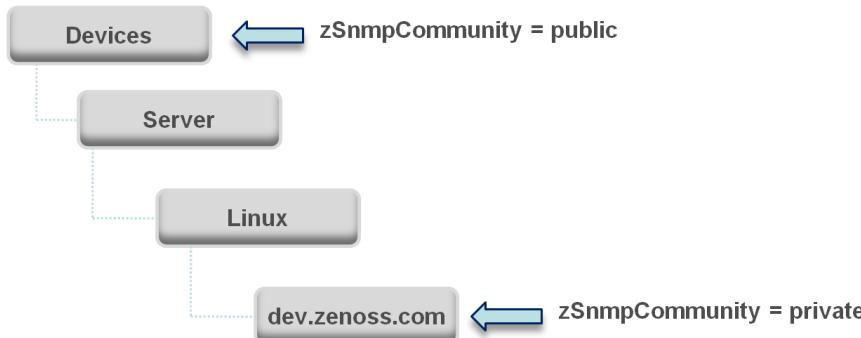


Figure 5.3. Device Class Hierarchy - Value Set on Device

If you change the SNMP Community value of dev.zenoss.com to public, it matches the value set at the root, but is still explicitly defined. Only if you remove the locally defined property does it again inherit the value of the property set at the root.

### 5.1.1.1. Viewing Properties from the User Interface

This section further illustrates the characteristics of configuration properties from the user interface perspective. The following screen shows device configuration properties defined at the root level. To view configuration properties:

1. Select Infrastructure from the navigation bar.

The Devices page appears.

2. Click **Details**.
3. Select Configuration Properties.

Configuration Properties			
Property	Value	Type	Path
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		password	/
zCommandPath	/usr/local/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device	lines	/
zEnablePassword		password	/

Figure 5.4. Defined Device Configuration Properties - Root Level

As shown in the previous screen, the zCollectorClientTimeout configuration property has a default value of 180. In the next screen, the value has been set to 170 at the /Server/Linux device class, overriding the default value at this node of the hierarchy.

Configuration Properties		Type	Path
Property	Value		
zCollectorClientTimeout	170	int	/Server/Linux
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	edit	/Server/Linux
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		password	/
zCommandPath	/usr/local/zenoss/libexec	string	/

Figure 5.5. zCollectorClientTimeout Configuration Property - Local Value Set

To remove the override and once again inherit the value from the root of the hierarchy, go to the Delete Local Property area (located at the bottom of the page), select the overridden property, and then click **Delete**.

Delete Local Property	
zCollectorClientTimeout	<input type="button" value="Delete"/>

Figure 5.6. Delete Local Property

## 5.1.2. Configuration Property Types

Configuration properties can be one of these types:

- **String** - Text value that can be ASCII or Latin-1 encoded
- **Integer** - Whole number
- **Float** - Number that can have a decimal value
- **Boolean** - True or False
- **Lines** - List of values separated by a return. The system stores these as an array.

## 5.1.3. Device Configuration Properties

To view and edit device configuration properties at the root level:

1. Select Infrastructure from the navigation bar.
2. In the tree view, click **Details**.
3. Select Configuration Properties.

To view and edit device configuration properties set at a specific device class, navigate to that class, click **Details**, and then select Configuration Properties.

You also can view and edit device configuration at the individual device level. Select the device from the device list, and then select Configuration Properties from the left panel.

The following table lists device configuration properties.

<b>Property Name</b>	<b>Property Type</b>	<b>Description</b>
zCollectorClientTimeout	int	Allows you to set the timeout time of the collector client in seconds
zCollectorDecoding	string	Converts incoming characters to Unicode.
zCollectorLogChanges	Boolean	Indicates whether to log changes.
zCollectorPlugins	lines	Links to all modeler plugins for this device.
zCommandCommandTimeout	float	Specifies the time to wait for a command to complete.
zCommandCycleTime	int	Specifies the cycle time you use when executing zCommands for this device or organizer.
zCommandExistanceTest	string	***
zCommandLoginTimeout	float	Specifies the time to wait for a login prompt.
zCommandLoginTries	int	Sets the number of times to attempt login.
zCommandPassword	string	Specifies the password to use when performing command logins and SSH.
zCommandPath	string	Sets the default path where ZenCommand plug-ins are installed on the local Zenoss box (or on a remote box where SSH is used to run the command).
zCommandPort	int	Specifies the port to connect to when performing command collection.
zCommandProtocol	string	Establishes the protocol to use when performing command collection. Possible values are SSH and telnet.
zCommandSearchPath	lines	Sets the path to search for any commands.
zCommandUsername	string	Specifies the user name to use when performing command collection and SSH.
zDeviceTemplates	lines	Sets the templates associated with this device. Linked by name.
zFileSystemMapIgnoreNames	string	Sets a regular expression of file system names to ignore.
zFileSystemMapIgnoreTypes	lines	Do not use.
zIcon	lines	Specifies the icon to represent the device wherever device icon is shown, such as on the network map and device status page.
zIfDescription	Boolean	Shows the interface description field in the interface list.
zInterfaceMapIgnoreNames	string	Filters out interfaces that should not be discovered.
zInterfaceMapIgnoreTypes	string	Filters out interface maps that should not be discovered.
zIpServiceMapMaxPort	int	Specifies the highest port to scan. The default is 1024.
zKeyPath	lines	Sets the path to the SSH key for device access.
zLinks	text	Specifies a place to enter any links associated with the device.
zLocalInterfaceNames	string	Regular expression that uses interface name to determine whether the IP addresses on an interface should be incorporated into the network map. For instance, a loopback interface "lo" might be excluded.

<b>Property Name</b>	<b>Property Type</b>	<b>Description</b>
zLocallpAddresses	int	Specifies IP addresses that should be excluded from the network map (for example. 127.x addresses). If you have addresses that you reuse for connections between clustered machines they might be added here as well.
zMaxOIDPerRequest	int	Sets the maximum number of OIDs to be sent by the SNMP collection daemons when querying information. Some devices have small buffers for handling this information so the number should be lowered.
zPingMonitorIgnore	Boolean	Whether or not to ping the device.
zProdStateThreshold	int	Production state threshold at which Zenoss will begin to monitor a device.
zPythonClass	string	DO NOT USE
zRouteMapCollectOnlyIndirect	Boolean	Only collect routes that are directly connected to the device.
zRouteMapCollectOnlyLocal	Boolean	Only collect local routes. (These usually are manually configured rather than learned through a routing protocol.)
zSnmpAuthPassword	string	The shared private key used for authentication. Must be at least 8 characters long.
zSnmpAuthType	string	Use "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.
zSnmpCommunity	string	Community string 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	"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, v2c, v3
zStatusConnectTimeout	float	The amount of time that the zenstatus daemon should wait before marking an IP service down.
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.

Property Name	Property Type	Description
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. The higher the number, the lower the severity. (1 is most severe; 5 is least severe.)
zWinPassword	string	The password used to remotely login if it is a Windows machine.
zWinUser	string	User name used to remotely login if it is a Windows machine.
zWmiMonitorIgnore	Boolean	Use to turn on or off all WMI monitoring.
zXmlRpcMonitorIgnore	Boolean	Use to turn on or off all XML/RPC monitoring.

*Table 5.1. Device Configuration Properties*

### 5.1.4. Event Configuration Properties

To view and edit event configuration properties at the root level:

1. Select Events from the navigation bar, and then select Event Classes.
2. In the left panel, select Configuration Properties.

To view and override event configuration properties for a specific event class, navigate to that class, and then select Configuration Properties.

The following table lists event configuration properties.

Property Name	Property Type	Description
zEventAction	string	Specifies the database table in 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	Lists classes that a clear event should clear (in addition to its own class).
zEventSeverity	int	Overrides the severity value of events from this class. Possible values are 0 – 5.

*Table 5.2. Event Configuration Properties*

### 5.1.5. Network Configuration Properties

To view and edit network configuration properties, select Infrastructure, and then select Networks. The Networks page appears.

## Properties and Templates

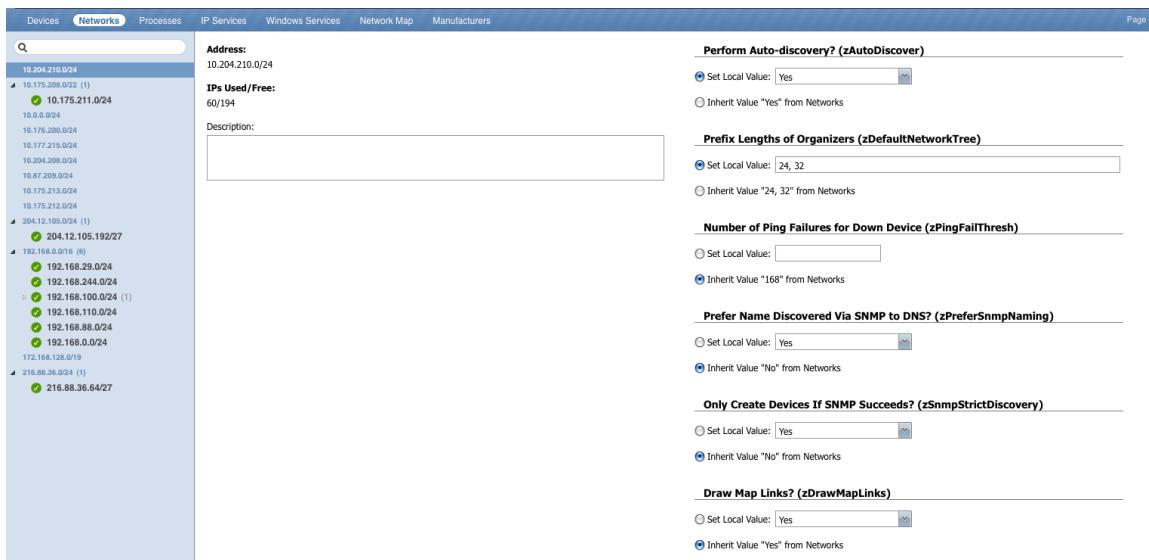


Figure 5.7. Networks

You can view and change network configuration property values and inheritance selections in the Configuration Properties area.

The following table lists network configuration properties.

Property Name	Property Type	Description
zAutoDiscover	Boolean	Specifies whether zendisc should perform auto-discovery on this network.
zDefaultNetworkTree	lines	A network subnet is automatically created for each modeled device, based on that device's subnet mask setting. To create higher-level subnets automatically from the discovery and modeling processes, add the specific subnet mask breakpoints. For example: 8, 16. If you then model a device with, for example, an IP address of 192.168.0.1, and a subnet mask of 255.255.255.0 (corresponding to a /24 subnet), device discovery will create a 192.0.0.0/8 network containing 192.168.0.0/16, containing 192.168.0.0/24, containing your device.
zDrawMapLinks	Boolean	Calculating network links "on the fly" is resource-intensive. If you have a large number of devices that have been assigned locations, then drawing those map links may take a long time. You can use this property to prevent the system from drawing links for specific networks (for example, a local network comprising many devices that you know does not span multiple locations).
zIcon	string	Use to specify device icons that appear on the device status page, Dashboard, and network map.
zPingFailThresh	int	Specifies the number of pings sent without being returned before zendisc removes the device.

Table 5.3. Network Configuration Properties

## 5.2. Templates

The system stores performance configuration data in *templates*. Templates contain other objects that define where and how to obtain performance data, thresholds for that data, and data graphs.

You can define a template anywhere in the device class hierarchy, or on an individual device.

Templates are divided among three types:

- Device
- Component
- Interface

### 5.2.1. Template Binding

The determination of which templates apply to what objects is called *binding*. Templates are bound in different ways, depending on the objects to which they are bound.

#### 5.2.1.1. Device Templates

Device templates are applied to devices, one to each device. The system employs a single rule to bind device templates to devices: the value of the zDeviceTemplates property. For most device classes, this is "Device."

Common device templates are:

- Device
- MySQL
- Apache
- Active Directory
- MSExchangeIS
- MSSQLServer
- IIS

For the Server/Linux/MySQL device class, the zDeviceTemplates property might contain, for example, "Device" and "MySQL." The system would collect CPU and memory information by using the Device template, and MySQL-specific metrics by using the MySQL template.

#### 5.2.1.1.1. Binding Templates

To bind a device template to a device class or device:

1. From the devices list, select a device class or device.
2. Select Bind Templates from  (Action Menu).

The Bind Templates dialog appears.

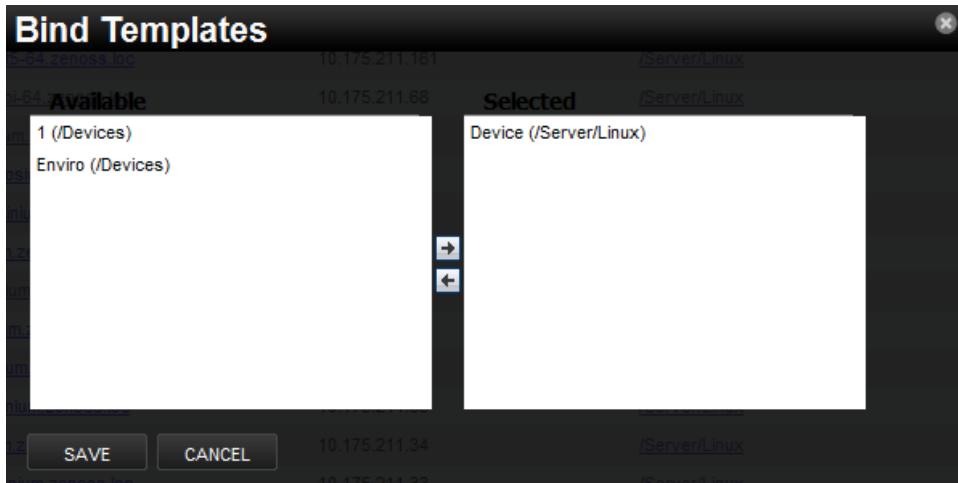


Figure 5.8. Bind Templates

3. Move templates between the Available and Selected lists using the arrows.
4. Click **Save**.

#### 5.2.1.1.2. Resetting Bindings

Resetting template bindings removes all locally bound templates and uses the default template values. To reset bindings for a selected device or device class:

1. Select Reset Bindings from  (Action Menu).  
The Reset Template Bindings dialog appears.
2. Click **Reset Bindings** to confirm the action.

#### 5.2.1.1.2. Component Templates

Component templates are named exactly according to the name of the underlying class that represents a component. For example, the FileSystem template is applied to file systems. Component templates can be applied multiple times to each device, depending on how many of the device's components match the template. Configuration properties do not control the application of component templates.

##### Note

Component templates should not be manually bound.

Common component templates are:

- FileSystem, HardDisk, IPService, OSProcess, WinService
- Fan, PowerSupply, TemperatureSensor
- LTMVirtualServer, VPNTunnel

#### 5.2.1.1.3. Interface Templates

Interface templates are applied to network interfaces by using a special type of binding. Instead of using the name of the underlying class, the system looks for a template with the same name as the interface type. You can find this type in the details information for any network interface.

If Zenoss cannot locate a template that matches the interface type, then it uses the ethernetCsmacd template.

## 5.2.2. Examples

### 5.2.2.1. Example: Defining Templates in the Device Hierarchy

You add a new device at /Devices/Server/Linux named Example1Server. You have not edited the value of its zDeviceTemplates property, so it inherits the value of "Device" from the root device class (/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 also is a template named Device defined at the root level (/Devices), but the system does not use this one because the template at /Devices/Server/Linux overrides it.)

### 5.2.2.2. Example: Applying Templates to Multiple Areas in the Device Hierarchy

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 template at /Devices called WebApplication with the appropriate data sources, thresholds and graphs. You then append the name "WebApplication" to the zDeviceTemplates configuration property for the devices classes, the individual devices running this Web application, or both.

---

# Chapter 6. Core Monitoring

Read the following sections for more information about basic and advanced monitoring, including:

- Availability monitoring
- Performance monitoring
- Monitoring using ZenCommand
- SNMP monitoring
- Monitoring devices remotely through SSH
- Monitoring windows devices

## 6.1. Availability Monitoring

The availability monitoring system provides active testing of the IT infrastructure, including:

- Devices
- Network
- Processes
- Services

Availability monitoring is facilitated by:

- **ZenPing** - The system's Layer-3 aware, topology-monitoring daemon. ZenPing performs high-performance, asynchronous testing of ICMP status. The most important element of this daemon is that Zenoss has built a complete model of your routing system. If there are gaps in the routing model, the power of ZenPing's topology monitoring will not be available. If there are gaps, this issue can be seen in the `zenping.log` file.
- **ZenStatus** - Performs active TCP connection testing of remote daemons.

Zenmodeler discovers the routes to each device in the network. The system tries not to use Internet routing tables, relying instead on Zenmodeler to discover the relationships and create its own network map.

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

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

### 6.1.1. Controlling Ping Cycle Time

Follow these steps to modify the ping cycle time.

1. Select Advanced, and then select Collectors.
2. Click a collector in the list.
3. Select Edit in the left panel.
4. Edit the value of Ping Cycle Time.

On the next configuration cycle, the ping monitor will ping at the newly defined interval.

### 6.1.2. Using the Predefined /Ping Device Class

The /Ping device class is a configuration for devices that you want to monitor only for availability. The system does not gather performance data for devices placed in this class. You can use the /Ping device class as a reference

for your own configuration; or, if you have a device that you want to monitor solely for availability, you can place it under this class.

### 6.1.3. Monitoring Processes

When enabled, the system can monitor the availability of all processes running on your network. As illustrated by the following diagram, ZenProcess checks for the existence of monitored processes. It uses a regular expression match to find PIDs matching the expression to see that these processes are running on a specific device.

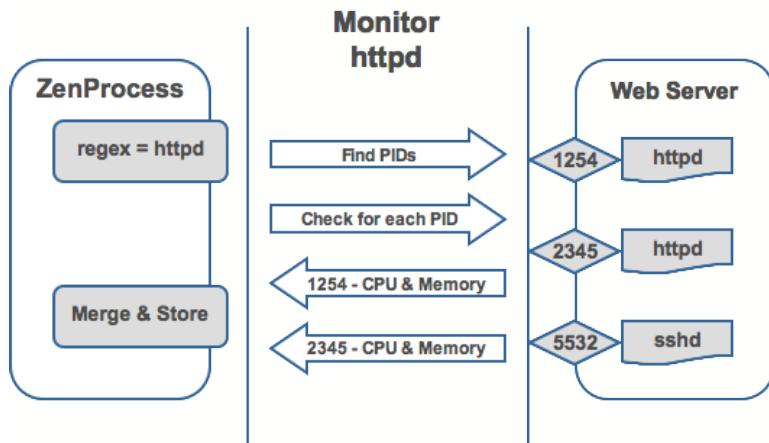


Figure 6.1. Process Monitoring

Use the Processes page (Infrastructure > Processes) to manage and monitor processes that are running on your network.

Name	Count
process0	0
process1	0
process10	0
process100	0
process101	0
process102	0
process103	0
process104	0
process105	0

Figure 6.2. Processes

The tree view lists all running processes. Filter this list by using the active search area at the top of the view.

#### 6.1.3.1. Monitoring a Process

To select a process to monitor, or edit monitoring choices for a process, follow these steps:

1. In the Processes page tree view, select the process to monitor.
2. Make one or more selections:

- **Enable Monitoring (zMonitor)** - By default, Inherit Value is selected for all processes below the Processes node. When selected, the process will inherit monitoring choices from its parent. If you want to individually enable monitoring choices for the process, select the Set Local Value option, and then select a value.
- **Send Event on Restart (zAlertOnRestart)** - By default, Inherit Value is selected for all processes below the Processes node. When selected, the process will inherit alert choices from its parent. If you want to individually select alert choices for the process, select the Set Local Value option, and then select a value.
- **Failure Event Severity (zFailSeverity)** - By default, Inherit Value is selected for all processes below the Processes node. When selected, the process will inherit severity level choices from its parent. If you want to individually select severity levels for the process, select the Set Local Value option, and then select a value.

3. In the Regular Expression area, enter information or make selections:

- **Pattern** - Enter text, as a regular expression, that the system will match to define this process.

**Tip**

For more information about regular expressions, go to:

[http://en.wikipedia.org/wiki/Regular\\_expression](http://en.wikipedia.org/wiki/Regular_expression)

- **Ignore Parameters** - Select this option to ignore command parameters when matching processes. By default, this option is disabled.

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

The next time the device is remodeled (either at the next remodeling interval or when manually initiated), each device (occurrence) where this process is running will appear in the Devices area.

If a process has multiple instances, the system 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 will be graphed for the single process.

To perform process monitoring using the `zenprocess` daemon, the device's SNMP agent must show the process information through the HOST- RESOURCES MIB.

## 6.1.4. Monitoring IP Services

The IP Services page (Infrastructure > IP Services) lets you manage and monitor IP services that are running on your network.

Name	Port	Count
1cl-smcs	3091	
3com-amp3	629	
3com-net-mgmt	2391	
3Com-nsd	1742	
3com-tsmux	106	
3com-webview	2339	
3comfaxrpc	3446	
3d-rfstd	2323	
3ds-lm	1538	
3I-I	1511	
3m-image-lm	1550	
4-tieropmcli	2934	
4-tieropmgw	2933	

**Enable Monitoring? (zMonitor)**

Set Local    Inherit Value "No" from Services

**Failure Event Severity (zFailSeverity)**

Set Local    Inherit Value "Critical" from Services

**Service Instances**

Device	Name	Monitored	Status

Figure 6.3. IP Services

The tree view lists all monitored IP services. Filter this list by using the active search area at the top of the view.

The details area shows:

- Service class description
- TCP port
- Associated service keys

To add or change details for a service class, enter or change information, and then click **Save**.

The lower section of the page lists currently running services in this class (by device), and shows their monitoring status.

#### 6.1.4.1. Enabling IP Service Monitoring

You can choose to monitor:

- Individual services
- Service classes

When monitoring a service class, you can choose not to monitor one or more individual services in the class. For example, the SMTP service class is monitored by default, but may not be a critical service on some devices. In this case, you can disable its monitoring on those devices.

##### Note

If a service is configured to listen only on local host (127.0.0.1), then it is not monitored by default.

To enable monitoring for a service class or service:

1. In the tree view, select the service class or service to monitor.
2. Make one or more selections:
  - **Enable Monitoring (zMonitor)** - By default, Inherit Value is selected for all services below the IPSERVICE node. When selected, the service class or service will inherit monitoring choices from its parent. If you want to individually enable monitoring choices, select the Set Local Value option, and then select a value.
  - **Failure Event Severity (zFailSeverity)** - By default, Inherit Value is selected for all services below the IPSERVICE node. When selected, the service class or service will inherit severity level choices from its parent. If you want to individually select severity levels, select the Set Local Value option, and then select a value.
3. Click **Save** to save your choices.

#### 6.1.4.2. Using the Predefined /Server/Scan Device Class

The predefined /Server/Scan device class is an example configuration for monitoring TCP services on devices using a port scan. If you have a device that you want to monitor for service availability alone, you can place it under this device class. The system will not collect performance data for devices in this class.

#### 6.1.5. Monitoring Windows Services

The IP Services page (Infrastructure > Windows Services) lets you manage and monitor Windows services that are running on your network.

The tree view lists all monitored Windows services. Filter this list by using the active search area at the top of the view.

The details area shows:

- Service class description

- Associated service keys

To add or change details for a service class, enter or change information, and then click **Save**.

The lower section of the page lists currently running service instances in this class (by device), and shows their monitoring status.

### 6.1.5.1. Enabling Windows Service Monitoring

You can choose to monitor:

- Individual services
- Service classes

To enable monitoring for a service class or service:

- In the tree view, select the service class or service to monitor.
- Make one or more selections:
  - Enable Monitoring (zMonitor)** - By default, Inherit Value is selected for all services below the WinService node. When selected, the service class or service will inherit monitoring choices from its parent. If you want to individually enable monitoring choices, select the Set Local Value option, and then select a value.
  - Failure Event Severity (zFailSeverity)** - By default, Inherit Value is selected for all services below the WinService node. When selected, the service class or service will inherit severity level choices from its parent. If you want to individually select severity levels, select the Set Local Value option, and then select a value.
- Click **Save** to save your choices.

## 6.2. Performance Monitoring

Read this chapter to learn about performance monitoring and monitoring templates.

### 6.2.1. About Performance Monitoring

Zenoss uses several methods to monitor performance metrics of devices and device components. These are:

- ZenPerfSNMP** - Collects data through SNMP from any device correctly configured for SNMP monitoring.
- ZenWinPerf** (Enterprise only) - ZenPack that allows performance monitoring of Windows servers.
- ZenCommand** - Logs in to devices (by using telnet or ssh) and runs scripts to collect performance data.
- Other ZenPacks** - Collect additional performance data. Examples include the ZenJMX ZenPack, which collects data from enterprise Java applications, and the HttpMonitor ZenPack, which checks the availability and responsiveness of Web pages.

Regardless of the monitoring method used, the system stores performance monitoring configuration information in *monitoring templates*.

### 6.2.2. About Monitoring Templates

Monitoring templates determine how the system collects performance data for devices and device components. You can define monitoring templates for device classes and individual devices.

Templates comprise three types of objects:

- Data Sources** - Specify the exact data points to collect, and the method to use to collect them.
- Thresholds** - Define expected bounds for collected data, and specify events to be created if the data does not match those bounds.

- **Graph Definitions** - Describe how to graph the collected data on the device or device components.

Before the system can collect performance data for a device or component, it must determine which monitoring templates apply. This process is called *template binding*.

### 6.2.2.1. Viewing Monitoring Templates

To view monitoring templates, select Advanced from the navigation bar, and then select Monitoring Templates.

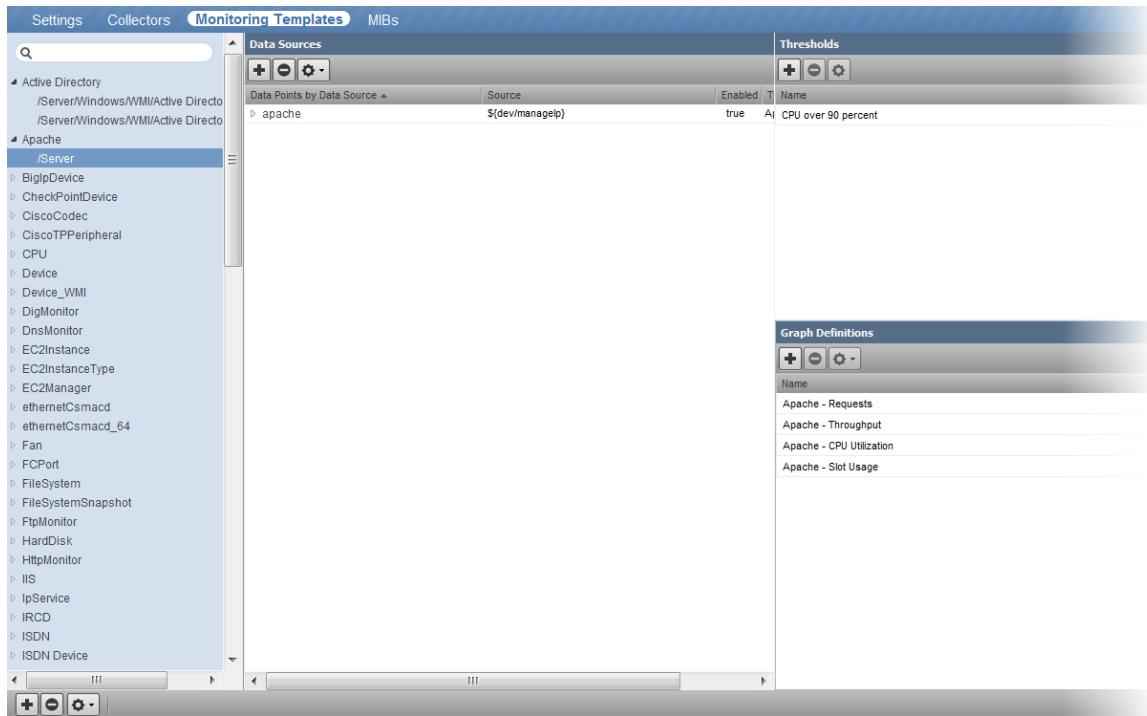


Figure 6.4. Monitoring Template for Load Average Graph

### 6.2.3. Template Binding

Before the system can collect performance data for a device or component, it must determine which templates apply. This process is called template binding.

First, the system determines the list of template names that apply to a device or component. For device components, this usually is the meta type of the component (for example, FileSystem, CPU, or HardDisk). For devices, this list is defined by the zDeviceTemplates configuration property.

After defining the list, the system locates templates that match the names on the list. For each name, it searches the device and then searches the device class hierarchy. Zenoss uses the lowest template in the hierarchy that it can locate with the correct name, ignoring others of the same name that might exist further up the device class hierarchy.

#### 6.2.3.1. Binding Templates

To edit the templates bound to a device:

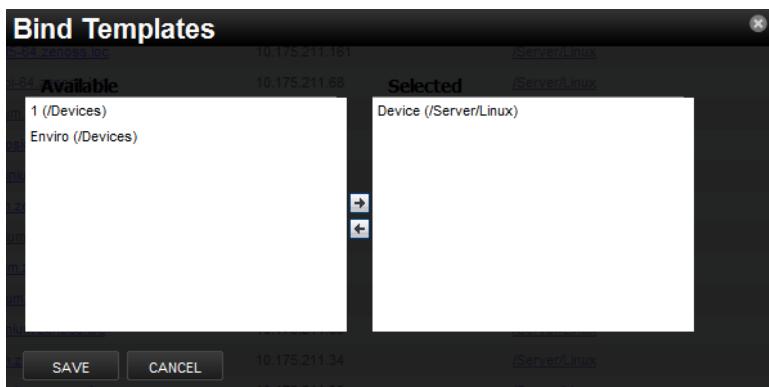
1. From the navigation bar, select Infrastructure.

The device list appears.

2. Select a device in the device list.

3. Select Bind Templates from  (Action menu).

The Bind Templates dialog appears.



*Figure 6.5. Bind Templates*

4. Select a template from the Available list and move it to the Selected list to bind it to the selected device.
5. Click **Save**.

## 6.2.4. Data Sources

Data sources specify which data points to collect and how to collect them. Each monitoring template comprises one or more data sources. The system provides two built-in data source types: SNMP and COMMAND. (Other data source types are provided through ZenPacks.)

- **SNMP** - Define data to be collected via SNMP by the ZenPerfSNMP daemon. They contain one additional field to specify which SNMP OID to collect. (Many OIDs must end in .0 to work correctly.) Because SNMP data sources specify only one performance metric, they contain a single data point.
- **Command** - specify data to be collected by a shell command that is executed on the Zenoss server or on a monitored device. The ZenCommand daemon processes COMMAND data sources. A COMMAND data source may return one or more performance metrics, and usually has one data point for each metric.

Shell commands used with COMMAND data sources must return data that conforms to the Nagios plug-in output specification. For more information, see the section titled Monitoring Using ZenCommand.

### 6.2.4.1. Adding a Data Source

To add a data source to a monitoring template:

1. Select Advanced from the navigation bar, and then select Monitoring Templates.
2. In the tree view, select the monitoring template to which you want to add a data source.
3. In the Data Sources area, select  (Add Data Source) from the Action menu.

The Add Data Source dialog appears.

4. Enter a name for the data source and select the type, and then click **Submit**.

The data source is added to the list in the Data Sources area.

5. Double-click the data source in the list.

The Edit Data Source dialog appears.

6. Enter or select values to define the data source.

### 6.2.5. Data Points

Data sources can return data for one or more performance metrics. Each metric retrieved by a data source is represented by a data point.

#### Defining Data Points

You can define data points to data sources with all source types except SNMP and VMware. Because these data source types each rely on a single data point for performance metrics, additional data point definition is not needed.

To add a data point to a data source:

1. Select Advanced from the navigation bar, and then select Monitoring Templates.
2. In the Data Sources area, highlight the row containing a data source.
3. Select Add Data Point from the Action menu.

The Add Data Point dialog appears.

4. Enter a name for the data point, and then click **Submit**.

#### Note

For COMMAND data points, the name should be the same as that used by the shell command when returning data.

5. Double-click the newly added data point to edit it. Enter information or make selections to define the data point:

- **Name** - Displays the name you entered in the Add a New DataPoint dialog.
- **RRD Type** - Specify the RRD data source type to use for storing data for this data point. (Zenoss uses RRDTool to store performance data.) Available options are:
  - **COUNTER** - Saves 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.
  - **GAUGE** - Does not save the rate of change, but saves the actual value. There are no divisions or calculations. To see memory consumption in a server, for example, you might want to select this value.

#### Note

Rather than COUNTER, you may want to define a data point using DERIVED and with a minimum of zero. This creates the same conditions as COUNTER, with one exception. Because COUNTER is a "smart" data type, it can wrap the data when a maximum number of values is reached in the system. An issue can occur 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 creates statistical anomalies.

- **DERIVE** - Same as COUNTER, but additionally allows negative values. If you want to see the rate of change in free disk space on your server, for example, then you might want to select this value.
- **ABSOLUTE** - Saves the rate of change, but 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, ABSOLUTE stores the current value, divided by the step interval.
- **Create Command** - Enter an RRD expression used to create the database for this data point. If you do not enter a value, then the system uses a default applicable to most situations.

For details about the rrdcreate command, go to:

<http://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html>

- **RRD Minimum** - Enter a value. Any value received that is less than this number is ignored.
  - **RRD Maximum** - Enter a value. Any value received that is greater than this number is ignored.
6. Click **Save** to save the defined data point.

## 6.2.6. Data Point Aliases

Performance reports pull information from various data points that represent a metric. The report itself knows which data points it requires, and which modifications are needed, if any, to put the data in its proper units and format.

The addition of a data point requires changing the report.

### CPU Utilization Report

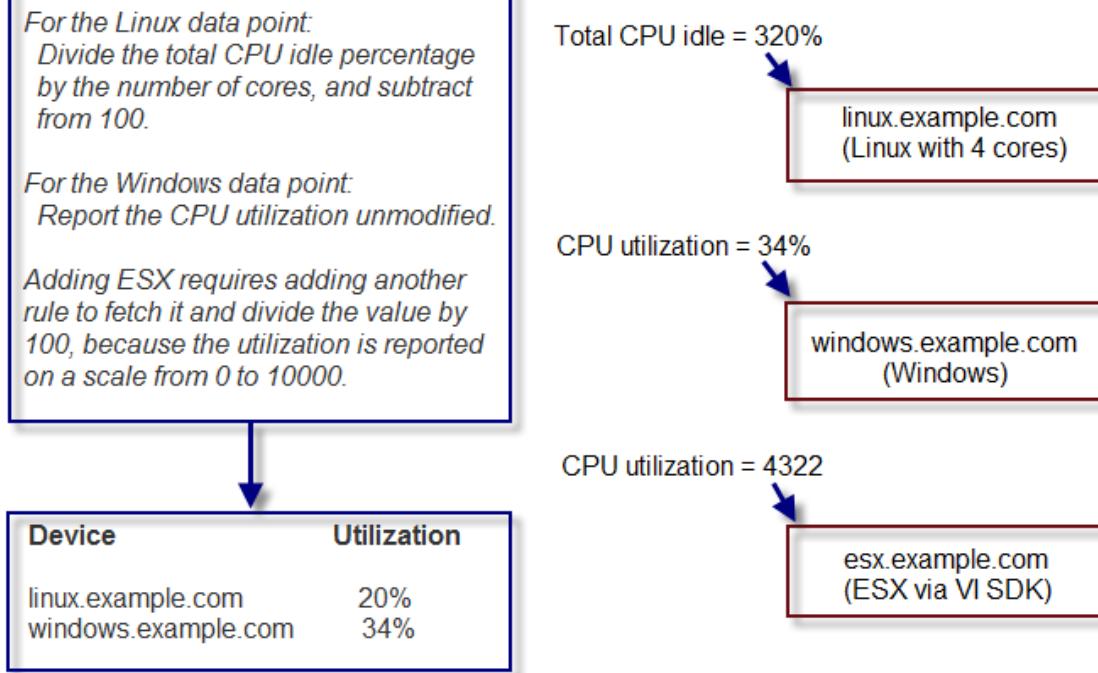


Figure 6.6. CPU Utilization Report

To allow for more flexibility in changes, some reports use *data point aliases*. Data point aliases group data points so they can be more easily used for reporting. In addition, if the data points return data in different units, then the plugin can normalize that data into a common unit.

An alias-based report looks up the data points that share a common alias string, and then uses them. This approach allows you to add data points without changing the report.

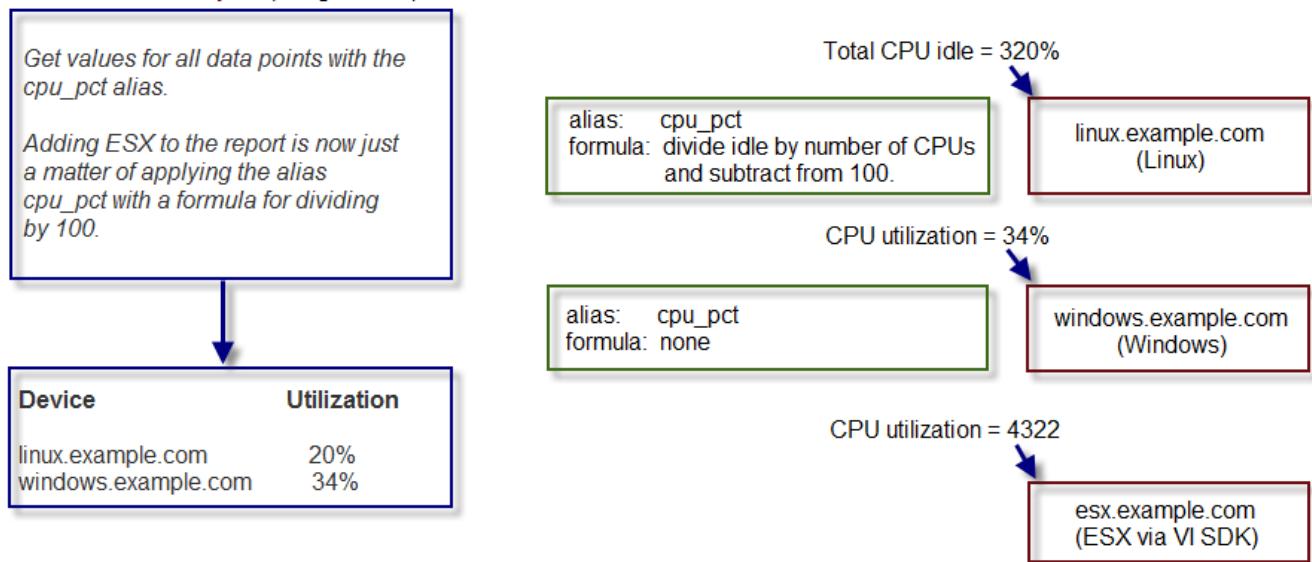
**CPU Utilization Report** (using aliases)

Figure 6.7. Alias-Based CPU Utilization Report

In the simplest cases, data from the target data points are returned in units expected by a report. For cases in which data are not returned in the same units, an alias can use an associated formula at the data point. For example, if a data point returns data in kilobytes, but the report expects data in bytes, then the formula multiplies the value by 1024.

### 6.2.6.1. Alias Formula Evaluation

The system evaluates the alias formula in three passes.

#### 6.2.6.1.1. Reverse Polish Notation

When complete, the alias formula must resolve to a Reverse Polish Notation (RPN) formula that can be used by RRDtool. For the simple conversion of kilobytes into bytes, the formula is:

```
1024,*
```

For more information on RRDtool and RPN formulas, browse to this site:

[http://oss.oetiker.ch/rrdtool/doc/rrdgraph\\_rpn.en.html](http://oss.oetiker.ch/rrdtool/doc/rrdgraph_rpn.en.html)

#### 6.2.6.1.2. Using TALES Expressions in Alias Formulas

For cases in which contextual information is needed, the alias formula can contain a TALES expression that has access to the device as context (labeled as "here"). The result of the TALES evaluation should be an RRD formula.

For example, if the desired value is the data point value divided by total memory, the formula is:

```
 ${here/hw/totalMemory},/
```

For more information on TALES, refer to the appendix in this guide titled "TALES Expressions," or to the TALES Specification 1.3, at:

<http://wiki.zope.org/ZPT/TALESSSpecification13>

### 6.2.6.1.3. Using Python in Alias Formulas

You also can embed full Python code in an alias formula for execution. The code must construct a string that results in a valid RRD formula. To signal the system to evaluate the formula correctly, it must begin with:

```
__EVAL:
```

Using the same example as in the previous section (division by total memory), the formula is:

```
__EVAL:here.hw.totalMemory + ",/"
```

### 6.2.6.2. Adding a Data Point Alias

To add an alias to a data point:

1. Navigate to a data source on a monitoring template.
2. Double-click a data point in the list to edit it.

The Edit Data Point dialog appears.

3. Enter the alias name and the formula.

#### Note

If the data point returns values in the desired units, then leave the Formula value blank.

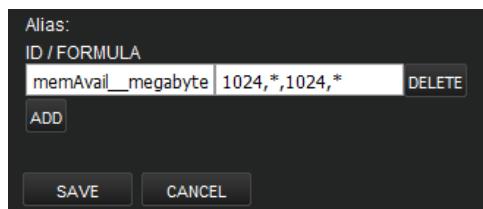


Figure 6.8. Add Data Point Alias

4. Click **Save**.

### 6.2.6.3. Reports That Use Aliases

For information about reports that use aliases, refer to the chapter titled "Reporting."

The following table shows performance reports that use aliases, and the aliases used. To add data points to a report, add the alias, and then ensure the values return in the expected units.

#### CPU Utilization

Alias	Expected Units
loadAverage5min	Processes
cpu_pct	Percent

### 6.2.7. Thresholds

Thresholds define expected bounds for data points. When the value returned by a data point violates a threshold, the system creates an event.

### 6.2.7.1. MinMax Threshold

The system provides one built-in threshold type: the MinMax threshold. (Other threshold types are provided through ZenPacks.)

MinMax thresholds inspect incoming data to determine whether it exceeds a given maximum or falls below a given minimum. You can use a MinMax threshold to check for these scenarios:

- *The current value is less than a minimum value.* To do this, you should set only a minimum value for the threshold. Any value less than this number results in creation of a threshold event.
- *The current value is greater than a maximum value.* To do this, you should set only a maximum value for the threshold. Any value greater than this number results in creation of a threshold event.
- *The current value is not a single, pre-defined number.* To do this, you should set the minimum and maximum values for the threshold to the same value. This will be the only "good" number. If the returned value is not this number, then a threshold event is created.
- *The current value falls outside a pre-defined range.* To do this, you should set the minimum value to the lowest value within the good range, and the maximum value to the highest value within the good range. If the returned value is less than the minimum, or greater than the maximum, then a threshold event is created.
- *The current value falls within a pre-defined range.* To do this, you should set the minimum value to the highest value within the bad range, and the maximum value to the lowest value within the bad range. If the returned value is greater than the maximum, and less than the minimum, then a threshold event is created.

### 6.2.7.2. Adding Thresholds

Follow these steps to define a MinMax threshold for a data point:

1. Select Advanced from the navigation bar, and then select Monitoring Templates.
2. In the Thresholds area, click  (Add Threshold).

The Add Threshold dialog appears.

3. Select the threshold type and enter a name, and then click **Add**.
4. Double-click the newly added threshold in the list to edit it.

The Edit Threshold dialog appears.

**Edit Threshold**

Name:	high utilization						
Type:	MinMaxThreshold						
DataPoints:	<table border="1"> <tr> <td>Available</td> <td>Selected</td> </tr> <tr> <td>ifInErrors_ifInErrors ifInUcastPackets_ifInUcastPackets ifOperStatus_ifOperStatus ifOutErrors_ifOutErrors ifOutOctets_ifOutOctets ifOutUcastPackets_ifOutUcastPackets</td> <td>ifInOctets_ifInOctets ifOutOctets_ifOutOctets</td> </tr> <tr> <td><input type="button" value="→"/></td> <td><input type="button" value="←"/></td> </tr> </table>	Available	Selected	ifInErrors_ifInErrors ifInUcastPackets_ifInUcastPackets ifOperStatus_ifOperStatus ifOutErrors_ifOutErrors ifOutOctets_ifOutOctets ifOutUcastPackets_ifOutUcastPackets	ifInOctets_ifInOctets ifOutOctets_ifOutOctets	<input type="button" value="→"/>	<input type="button" value="←"/>
Available	Selected						
ifInErrors_ifInErrors ifInUcastPackets_ifInUcastPackets ifOperStatus_ifOperStatus ifOutErrors_ifOutErrors ifOutOctets_ifOutOctets ifOutUcastPackets_ifOutUcastPackets	ifInOctets_ifInOctets ifOutOctets_ifOutOctets						
<input type="button" value="→"/>	<input type="button" value="←"/>						
Severity:	Warning						
<input checked="" type="checkbox"/> Enabled							
Minimum Value:							
Maximum Value:	(here.speed or 1e9) / 8 * .75						
Event Class:	/Perf/Interface						
Escalate Count:	0						
<input type="button" value="SAVE"/> <input type="button" value="CANCEL"/>							

Figure 6.9. Edit Threshold

## 5. Enter or select values to define the threshold:

- **Name** - Displays the value for the ID you entered on the Add a New Threshold dialog.
- **Data Points** - Select one or more data points to which this threshold will apply.
- **Severity** - Select the severity level of the first event triggered when this threshold is breached.
- **Enabled** - Select True to enable the threshold, or False to disable it.
- **Minimum Value** - If this field contains a value, then each time one of the select data points falls below this value an event is triggered. This field may contain a number or a Python expression.

When using a Python expression, the variable `here` 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 frequently is reported in bits/second, where the performance data is bytes/second.

- **Maximum Value** - If this field contains a value, then each time one of the selected data points goes above this value an event is triggered. This field may contain a number or a Python expression.
- **Event Class** - Select the event class of the event that will be triggered when this threshold is breached.

- **Escalate Count** - Enter the number of consecutive times this threshold can be broken before the event severity is escalated by one step.
6. Click **Save** to save the newly defined threshold.

## 6.2.8. Performance Graphs

You can include any of the data points or thresholds from a monitoring template in a *performance graph*.

To define a graph:

1. Select Advanced from the navigation bar, and then select Monitoring Templates.

2. In the Graph Definitions area, click  (Add Graph).

The Add Graph Definition dialog appears.

3. Enter a name for the graph, and then click **Submit**.
4. Double-click the graph in the list to edit it. Enter information or select values to define the graph:
  - **Name** - Optionally edit the name of the graph you entered in the Add a New Graph dialog. This name appears as the title of the graph.
  - **Height** - Enter the height of the graph, in pixels.
  - **Width** - Enter the width of the graph, in pixels.
  - **Units** - Enter a label for the graph's vertical axis.
  - **Logarithmic Scale** - Select True to specify that the scale of the vertical axis is logarithmic. Select False (the default) to set the scale to linear. You might want to set the value to True, for example, if the data being graphed grows exponentially. Only positive data can be graphed logarithmically.
  - **Base 1024** - Select True if the data you are graphing is measured in multiples of 1024. By default, this value is False.
  - **Min Y** - Enter the bottom value for the graph's vertical axis.
  - **Max Y** - Enter the top value for the graph's vertical axis.
  - **Has Summary** - Select True to display a summary of the data's current, average, and maximum values at the bottom of the graph.

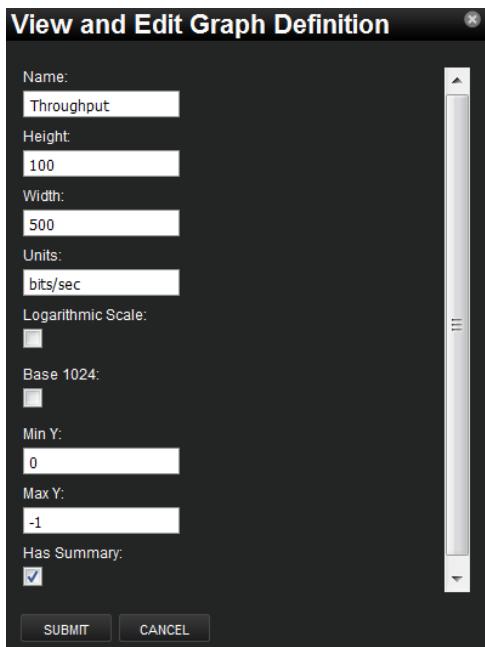


Figure 6.10. Graph Definition

1. Click **Submit** to save the graph.

### 6.2.8.1. Graph Points

Graph points represent each data point or threshold that is part of a graph. You can add any number of graph points to a graph definition by adding data points or thresholds.

From the Graph Definitions area of the Monitoring Templates page:

1. Select Manage Graph Points from the Action menu.

The Manage Graph Points dialog appears.

2. From the Add menu, add a data point, threshold, or custom graph point.
3. Select values, and then click **Submit**.

the new graph point appears in the Graph Points list.

#### Note

Thresholds are always drawn before other graph points.

#### 6.2.8.1.1. Re-sequencing Graph Points

To re-sequence graph points, drag a graph point row in the Manage Graph Points dialog. (Click and drag from an "empty" part of the row.)

#### 6.2.8.1.2. DataPoint Graph Points

DataPoint graph points draw the value of data points from the template on a graph.

##### 6.2.8.1.2.1. Adding DataPoint Graph Points

To define a DataPoint graph point:

1. From the Add menu on the Manage Graph Points dialog, select Data Point.

The Add Data Point dialog appears.

2. Select one or more data points defined in this template. On data point graph point is created for each data point you select from the list.
3. Optionally select the Include Related Thresholds option. If selected, then any graph points are created for any thresholds that have been applied to the selected data points as well.
4. Click **Submit**.

#### 6.2.8.1.2.2. Editing DataPoint Graph Points

Double-click the name of the graph point to go to its edit page. Enter information or select values to edit the graph point:

- **Name** - This is the name that appears on the Graph Definition page. By default, it appears in the graph legend.
- **Line Type** - Select Line to graph the data as a line. Select Area to fill the area between the line and the horizontal axis with the line color. Select None to use this data point for custom RRD commands and do not 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 now. You might set this value, for example, to asses total packets if measuring packets in and packets out.
- **Format** - Specify the RRD format to use when displaying values in the graph summary. For more information on RRDTool formatting strings, go to:

[http://oss.oetiker.ch/rrdtool/doc/rrdgraph\\_graph.en.html](http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html)

- **RPN** - Optionally 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 want to graph it as bytes, enter an RPN value of "8,/" to divide by 8. For more information about RRDTool RPN notation, go to:

<http://oss.oetiker.ch/rrdtool/tut/rpntutorial.en.html>

- **Limit** - Optionally specify a maximum value for the data being graphed.
- **Consolidation** - Specify the RRD function used to graph the data point's data to the size of the graph. Most of the time, the default value of AVERAGE is appropriate.
- **Color** - Optionally specify a color for the line or area. Enter a six-digit hexadecimal color value with an optional two-digit hex value to specify an alpha channel. An alpha channel value is only used if 'stacked' is True.
- **Legend** - Name to use for the 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** - Lists the RRD variables defined in this graph definition. These values can be used in the RPN field.

#### 6.2.8.1.3. Editing Threshold Graph Points

Threshold graph points graph the value of thresholds from the template.

To edit a threshold graph point, double-click it in the list:

You can edit values for Name, Color, and Legend for a threshold graph point. Refer to the definitions in the section titled [Editing DataPoint Graph Points](#) for more information.

#### 6.2.8.1.4. Editing 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, go to:

[http://oss.oetiker.ch/rrdtool/doc/rrdgraph\\_data.en.html](http://oss.oetiker.ch/rrdtool/doc/rrdgraph_data.en.html)

For details on other RRD commands, go to:

[http://oss.oetiker.ch/rrdtool/doc/rrdgraph\\_graph.en.html](http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html)

#### 6.2.8.2. Custom Graph Definition

Custom graph definitions allow you to specify your own set of RRD commands to draw a graph. The graph points specified are used to define data that is available to the commands you specify here; however, the graph points are not drawn unless you explicitly draw them with the commands you specify. The Available RRD Variables lists the values defined by the graph points that are available for use.

To access custom graph definitions, select Custom Graph Definition from the Action menu in the Graph Definitions area.

#### 6.2.8.3. Graph Commands

Graph Commands show an approximate representation of the RRD commands that will be used to draw a graph. This representation provides helpful debugging information when using custom graph points or custom graph definitions.

To view graph commands, select Graph Commands from the Action menu in the Graph Definitions area.

### 6.3. Monitoring Using ZenCommand

Read the following sections for more information about monitoring using ZenCommand.

#### 6.3.1. About ZenCommands

Zenoss has the ability to run Nagios® and Cactii plug-ins through the ZenCommand process. ZenCommand can run plugins locally and remotely by using a native SSH transport. When run, the system tracks the return code of each plugin and then creates events with plugin output. Additionally, it can track performance information from a plugin.

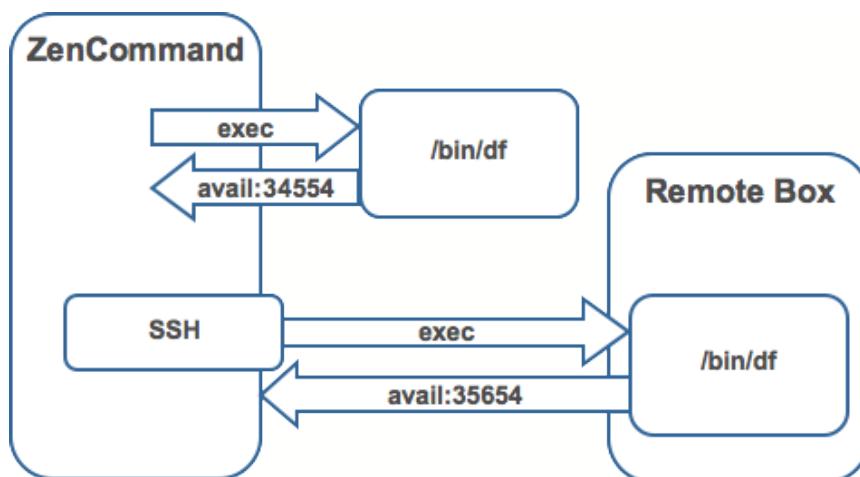


Figure 6.11. Running ZenCommands

### 6.3.2. Example: Writing a ZenCommand (check\_http example)

You can use a ZenCommand plugin (check\_http) to check for specific content of a Web page. (This implicitly checks for server/page 200 status as well.)

The following example procedure shows how to set up a ZenCommand plugin to check specific content. The steps show how to test the plugin, and then integrate it.

1. As the zenoss user, test the plugin from the command line. Enter the following 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
```

#### Note

The check\_http -h command displays all plugin options.

2. Add the device you want to check (one running a "www" Web site) to the Zenoss system, setting the discovery protocol to "none."
3. Navigate to the device, and then select Monitoring Templates from the left panel.
4. Create a local copy.

The default device template is overridden with the device template specific to this device.

5. In the template, remove the sysUpTime data source. (In this example, SNMP is not used for the device.)
6. Add a new description to the template.
7. Add a new data source called rootWebCheck.
8. In the rootWebCheck data source, set the following variables:
  - Source Type = COMMAND
  - Component = rootWebCheck
  - Cycle Time = 30
9. 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 want to monitor.

The command template field is a TALES expression. You can make substitutions in the command that will make it generic for any device added to this class.

10. Set the -H flag to the IP of the device against which this command will be run, as follows:

```
check_http -H ${here/manageIp}
```

11. 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.

```
check_http -H ${here/manageIp}-r textstring1
```

Where *textstring1* is text you know is on the resulting Web page.

12. For this example, the command should be generic. 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

13. Return to the template and change the command to be.

```
check_http -H ${here/manageIp}-r ${here/cWebMatchRegex}
```

14. Add the regex value into cWebMatchRegex (used in the example above).

15. Test the ZenCommand from the command line.

### 6.3.3. Example: Collect Data from A ZenCommand

To collect and display data from the ZenCommand check\_http example, you can log the data to see something like response time in a graphical format.

1. Navigate to /Web/Device template.
2. Go to the data source created in the check\_http example.
3. Add a data point named "time." (No modifications are needed to the data point.)
4. Test the command again. You should see a log message that starts with:

```
DEBUG:zen.zencommand:storing responseTime = 1.0
```

5. Make a graph to display the data. In the device template, create a graph called "Web Response Time."
6. For this graph, set the following values:
  - Data Sources = rootWebCheck\_responseTime
  - Units = Seconds
  - Min Y = 0
7. Go to Graphs for the device www.website.com (the Web site you were using to check) to see the graph. Graph data will not appear until collection is run several times. Restart zencommand so that the new configuration takes effect immediately.

### 6.3.4. Plugin Format for ZenCommands

Nagios® plugins are configured by using a command template that is much like the RRDTemplates used for performance monitoring.

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 the first line of stdout, plus a return code.

#### Note

Zenoss return codes differ from Nagios return codes, as follows:

Value	Zenoss	Nagios
0	Clear	OK
1	Data Source	WARNING
2	Data Source+1	CRITICAL

Value	Zenoss	Nagios
3	Data Source	UNKNOWN

For complete information about Nagios plugin guidelines, browse to this location:

<http://nagiosplug.sourceforge.net/developer-guidelines.html>

A Nagios® command has several fields:

- name – Specifies the name of the command object.
- enabled – Indicates whether this command should be used on a given device.
- component – Specifies the component name to use when zencommand sends events to the system.
- event class – Specifies the event class to use when sending events to the system.
- severity – Sets the default severity to use when sending events to the system.
- cycle time – Sets the frequency a command should be run (in seconds).
- command template – Specifies the command to run.

The command template string is built by using Zope TALES expressions. Several variables are passed when evaluating the template. They are:

- zCommandPath – Path to the zencommand plug-ins on a given box it comes from the configuration property zCommandPath. zCommandPath is automatically added to a command if a path is absent from the beginning of the command.
- devname – Device name of the device against which the command is being evaluated.
- dev – Device object against which the command is being evaluated.
- here – 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, specifies its name as a string.
- now – Current time.

Template values are accessed like shell variables. They are the same as the expression syntax used in the appendix titled TALES Expressions (in this guide).

## Examples

Run an http check against all devices by using the URL /zport/dmd:

```
check_http -H ${devname} -u /zport/dmd
```

In a template named FileSystem, the following command will run against all file systems on a device:

```
check_disk -w 10% -c 5% -p ${compname}
```

### 6.3.5. Testing ZenCommands

You can test ZenCommand data sources by using the zentestcommand shell script.

From the command line, run:

```
zentestcommand -d DeviceName --datasource=DataSourceName
```

where *DeviceName* is the device on which you want to run the command, and *DataSourceName* is the name of a data source on a template associated with the device.

The zentestcommand script prints the results of the command to standard output.

## 6.4. SNMP Monitoring

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, use the command:

```
$ snmpget -v1 -cpn public 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 commands to gather certain information.

- a. Walk a basic system MIB.

```
snmpwalk -v1 -cpn public <device name> system
```

- b. Walk an interface description

```
snmpwalk -v1 -cpn public <device name> ifDescr
```

- c. Get a single value.

```
snmpget -v1 -cpn public <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
```

## 6.5. Monitoring Devices Remotely Through SSH

You can monitor devices remotely through SSH. To monitor devices remotely, you must install the Zenoss plugins on each remote device you want to monitor.

Follow the steps in the following sections to set up remote monitoring.

### 6.5.1. Installing Plugins on the Remote Machine

The plugins are packaged in two formats:

- **Native format (RPM)** - Recommended in Red Hat-based systems that support Red Hat Package Management. By using the RPM distribution, you can easily update the package when newer versions are released.
- **Source distribution** - Assembled using setuptools. When using the source distribution, you do not need root privileges to install the plugins.

### 6.5.1.1. Plugin Installation Technique: RPM

The RPM for the plugins is a noarch RPM, which means it can be installed on any architecture (such as i386, amd64, or ia\_64). The only external dependency needed to install the plugins RPM is Python. Most Linux distributions include Python in their standard loads.

To install the plugins RPM, use the following command:

```
$ sudo rpm -Uvh zenoss-plugins-*.rpm
```

where 'zenoss-plugins-\*.rpm' is the latest plugin RPM file.

### 6.5.1.2. Plugin Installation Technique: setuptools

Enter these commands to install the plugins into directories that are accessible to all users:

```
$ python setup.py build
```

```
$ sudo python setup.py install
```

If you do not have appropriate privileges to install the system software, refer to the following information about installing the plugins using a non-privileged account, at:

<http://dev.zenoss.org/trac/wiki/ZenossPlugins>

Alternatively, you can use setuptools's built-in `easy_install` command to install the plugins. To use `easy_install` to download and install the plugins, run the following command:

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

where 'Zenoss-Plugins' is the name of the latest plugin file.

### 6.5.1.3. Testing the Plugin Installation

The entry point to the 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 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 must 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 `zenplugin.py` with the plugin name as the argument. The following command line illustrates:

```
$ zenplugin.py cpu
```

### 6.5.1.4. Troubleshooting Plugin Installation

#### 6.5.1.4.1. "Command not found" when running `zenplugin.py`

If you receive a "command not found" error when running the `zenplugin.py` command, make sure that the directory into which it was installed is included in your PATH. If you installed by using RPM, you can use the command "`rpm -q zenoss-plugins | grep zenplugin.py`". If you installed via setuptools pay close attention to the "Installing..." messages to see the full directory paths.

#### 6.5.1.4.2. "platform 'XXX' is not implemented. no plugins exist"

This message indicates that plugins may not be fully implemented for your particular platform. If you receive this message, and want to investigate support for your platform, email the output of the following command to the Support team:

```
$ python -c 'import sys; print sys.platform'
```

#### 6.5.1.5. Changing Zenoss to Monitor Devices Remotely Using SSH

You must edit system 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 configuration properties value for the group. After selecting the device class, click Details, and then select Configuration Properties.

The Configuration Properties page appears.

Configuration Properties		Type	Path
Property	Value		
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/Discovered
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistenceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		password	/
zCommandPath	/usr/local/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device MySQL	lines	/
zEnablePassword		password	/
zFileSystemMapignoreNames		string	/

Figure 6.12. Device Class Configuration Properties

You must make changes to the following configuration properties:

- zCollectorPlugins
- zCommandPassword
- zCommandPath
- zCommandUsername
- zSnmpMonitorIgnore
- zTransportPreference

The following table lists sample values set up for remote devices. These have a pre-shared key (with no password) set up from the collector to the remote boxes. (It also can use password authorization if the password is entered into zCommandPassword.)

Configuration Properties	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

Two passes are required for full modeling. The first pass obtains the platform type (so that the system knows which plugins to run). The second pass provides detailed data on interfaces and file systems.

Run the command:

```
$ zenmodeler run -d enter_server_name_here
```

Run the command a second time to use the plugins the command gathered on the first pass.

#### 6.5.1.6. Using the Predefined /Server/Cmd Device Class

The /Server/Cmd device class is an example configuration for modeling and monitoring devices using SSH. The configuration properties have been modified as described in the previous sections, 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 pre-configured templates and configuration properties. You will still need to set the zCommandUsername and zCommandPassword properties to the appropriate SSH login information for each device.

## 6.6. Monitoring Windows Devices

### 6.6.1. Device Preparation for Windows Devices

In all Zenoss versions, WMI is used to monitor the Windows event log and state of Windows services.

Before you can monitor Windows devices, you must ensure that:

- DCOM is enabled for WMI connections
- The hostname of the system collector does not exceed fifteen characters

If you are using Zenoss Core, you must additionally ensure that an SNMP agent is enabled on Windows devices. If your system is running Windows Vista, for example, follow these steps to see if the SNMP agent is enabled:

1. From the Start menu list, right-click Computer, and then select Manage from the list of options.
  2. From the Computer Management panel navigation area, expand Services and Applications, and then select Services.
- The Services list appears.
3. Locate the listing for SNMP Service. If it does not show a status of "Started," then click Start (the service).

#### Note

If SNMP Service does not appear in the list, then you may have to enable the SNMP feature (from the "Turn Windows features on and off" selection in the Control Panel).

Optionally, you can use SNMP Informant™ to collect CPU, memory, and disk I/O statistics. SNMP Informant agents collect information from Windows devices via WMI on the server where they are installed, and then convert system, state, and operational data into SNMP OIDs for broadcast. The system can then process the SNMP OID information and generate events and alerts based on this information. See the section titled Monitoring Windows Performance with SNMP Informant (in this chapter) for more information.

#### Note

If you are using Zenoss Enterprise, SNMP Informant is not needed (its functionality is included in these versions).

### 6.6.2. Setting Windows Configuration Properties

You must set the following configuration properties to collect information from Windows servers. In Zenoss, navigate to the configuration properties for each device, and then set the appropriate values for:

- **zWmiMonitorIgnore** - Turns on or off all WMI monitoring. Set the value of Ignore to False to turn on Windows monitoring.

You should set this property at the Server/Windows class level, so that any device placed in this class has Windows monitoring automatically enabled.

- **zWinUser** - Must be set as 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 log in to the Windows machine.

### 6.6.3. Testing WMI on a Windows Server

Follow these 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 the User and Password fields.
5. Click **Query**.
6. Enter “select \* from win32\_service” to return a dialog with a list of services on the device.

### 6.6.4. Optional Windows Configuration

The system can gather additional, detailed OS and hardware information from Windows devices if you have these agents installed on your Windows device:

- Dell Open Manage Agent
- HP Insight Management Agent

### 6.6.5. Modeling Services on Windows Devices

Zenoss uses ZenWin to perform Windows Service Monitoring over WMI. ZenWin monitors the up and down availability of Windows services.

The WinServiceMap WMI plugin is included in zCollectorPlugins on the /Server/Windows device class. WinServiceMap retrieves all services that can be monitored on a device, regardless of whether it is up or down.

Windows services are (by default) not monitored. To monitor a specific Windows service, follow these steps:

1. Navigate to Infrastructure > Windows Services.
2. Select the service you want to monitor from the list in the left panel.
3. Select Set Local Value for Enable Monitoring? (zMonitor), and then click **Save**.

## 6.6.6. Collecting Windows Eventlog Events

The system uses ZenEventlog to collect WMI event log events. Enable the following configuration properties to define how Windows event log events are processed and monitored:

- **zWinEventLog** - Tells the system whether or not to read the event log.
- **zWinEventLogMinSeverity** - Sets the minimum severity to collect from the Windows event log. The lowest number indicates the highest severity (1 is the most severe; 5 is least severe).

## 6.6.7. Monitoring Windows Performance with SNMP Informant

Zenoss can use information from SNMP Informant to collect SNMP information from Windows devices.

Install the free version of SNMP Informant from this location:

<http://www.snmp-informant.com>

To make sure SNMP Informant is running and set up 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, the system gathers and uses SNMP information the same as any other device sending SNMP traps.

## 6.6.8. Running winexe Commands on Windows Servers

You can use winexe commands to run commands on monitored Windows servers from within the system.

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 (IMPORTANT! 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 user name.
-N, --no-pass	Do not 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 7. Event Management

## 7.1. About Events

Events, and the graphs generated from performance monitoring, are the primary operational tools for understanding the state of your environment. This chapter: defines events and describes the event management system.

- Defines events
- Describes important event management system features, such as de-duplication and auto-clear correlation
- Provides more information about managing events through the user interface

### 7.1.1. Basic Event Fields

To enter the event management system, an event must contain values for the device, severity, and summary fields. If an event is missing any of these fields, then Zenoss rejects it.

Basic event fields are:

- device
- ipAddress
- eventState
- severity
- summary
- message
- evid

#### 7.1.1.1. device and ipAddress Fields

The device field is a free-form text field that allows up to 128 characters. Zenoss accepts any value for this field, including devices that are not in the database. If the device field contains an IP address, then the system queries the database for devices with a matching address. If it finds a match, it changes the device field to the found device name.

The ipAddress field is a free-form text field that allows up to 15 characters. This field is not required. If the system cannot successfully locate a device based on the event's device field content, it attempts to find the device based on the event ipAddress field content, if present.

Zenoss automatically adds information to incoming events that match a device in its database. Fields added are:

- **prodState** - Specifies the device's current production state.
- **Location** - Specifies the location (if any) to which the device is assigned.
- **DeviceClass** - Classifies the device.
- **DeviceGroups** - Specifies the groups (if any) to which the device is assigned.
- **Systems** - Systems (if any) to which the device is assigned.
- **DevicePriority** - Priority assigned to the device.

For more information about these fields, refer to the chapters titled "Production States and Maintenance Windows" and "Organizers and Path Navigation."

### 7.1.1.2. eventState Field

The eventState field defines the current state of the event. This field is often updated after an event has been created. Values for this numeric field are 0-2, defined as follows:

Number	Name
0	New
1	Acknowledged
2	Suppressed

### 7.1.1.3. severity Field

The severity field defines the severity of the event. Values for this numeric field are 0-5, defined as follows:

Number	Name	Color
0	Clear	Green
1	Debug	Grey
2	Info	Blue
3	Warning	Yellow
4	Error	Orange
5	Critical	Red

### 7.1.1.4. summary and message Fields

The summary and message fields are free-form text fields. The summary field allows up to 128 characters. The message field allows up to 65535 characters. These fields usually contain similar data.

The system handles these fields differently, depending on whether one or both are present on an incoming event:

- If only summary is present, then the system copies its contents into message and truncates summary contents to 128 characters.
- If only message is present, then the system copies its contents into summary and truncates summary contents to 128 characters.
- If summary and message are both present, then the system truncates summary contents to 128 characters.

As a result, data loss is possible only if the message or summary content exceeds 65535 characters, or if both fields are present and the summary content exceeds 128 characters.

To ensure that enough detail can be contained within the 128-character summary field limit, avoid reproducing information in the summary that exists on other fields (such as device, component, or severity).

### 7.1.1.5. evid

The evid field is the event identifier, or event ID. It is a 36-character, unique identifier for every event that comes into the system. An incoming event should never have an evid assigned to it, because the system creates it immediately before the event is inserted into the database. If an incoming event does have an assigned evid, then the system ignores it and replaces it with a generated evid.

## 7.1.2. Other Fields

Events include numerous other standard fields. Some control how an event is mapped and correlated; others provide information about the event.

The following table lists additional event fields.

Field	Description
depuid	Dynamically generated fingerprint that allows the system to perform de-duplication on repeating events that share similar characteristics.
component	Free-form text field (maximum 255 characters) that allows additional context to be given to events (for example, the interface name for an interface threshold event).
eventClass	Name of the event class into which this event has been created or mapped.
eventKey	Free-form text field (maximum 128 characters) that allows another specificity key to be used to drive the de-duplication and auto-clearing correlation process.
eventClassKey	Free-form text field (maximum 128 characters) that is used as the first step in mapping an unknown event into an event class.
eventGroup	Free-form text field (maximum 64 characters) that can be used to group similar types of events. This is primarily an extension point for customization. Currently not used in a standard system.
stateChange	Last time that any information about the event changed.
firstTime	First time that the event occurred.
lastTime	Most recent time that the event occurred.
count	Number of occurrences of the event between the firstTime and lastTime.
prodState	Production state of the device when the event occurred. If an event is still active when a device's production state is changed, the event's prodState will be updated accordingly.
suppid	If this event has been suppressed by another event, then suppid contains the other event's evid.
manager	Deprecated. The monitor field replaces this field.
agent	Typically the name of the daemon that generated the event. For example, an SNMP threshold event will have zenperfsnmp as its agent.
DeviceClass	Device class of the device that the event is related to.
Location	Location of the device that the event is related to.
Systems	Pipe-delimited list of systems that the device is contained within.
DeviceGroups	Pipe-delimited list of systems that the device is contained within.
facility	Only present on events coming from syslog. The syslog facility.
priority	Only present on events coming from syslog. The syslog priority.
ntevid	Only present on events coming from Windows event log. The NT Event ID.
ownerid	Name of the user who acknowledged this event.
clearid	Only present on events in history that were auto-cleared. The evid of the event that cleared this one.
DevicePriority	Priority of the device that the event is related to.
eventClassMapping	If this event was matched by one of the configured event class mappings, contains the name of that mapping rule.
monitor	In a distributed setup, contains the name of the collector from which the event originated.

### 7.1.3. Details

In addition to the standard fields, the system also allows events to add an arbitrary number of additional name/value pairs to events to give them more context. The name and value of these details are limited to 255 characters in length.

### 7.1.4. De-Duplication

Zenoss uses an event "de-duplication" feature, based on the concept of an event's fingerprint. Within the system, this fingerprint is the "dedupid." All of the standard events that the system creates as a result of its polling activities are de-duplicated, with no setup required. However, you can apply de-duplicating to events that arrive from other sources, such as syslog, SNMP traps, or a Windows event log.

The most important de-duplication concept is the *fingerprint*. In all cases, an event's fingerprint (or dedupid) is composed of a pipe-delimited string that contains these event fields:

- device
- component (can be blank)
- eventClass
- eventKey (can be blank)
- severity
- summary (omitted from the dedupid if eventKey is non-blank)

When the component and eventKey fields are blank, a dedupid appears similar to:

`www.example.com||/Status/Web||4|WebTx check failed`

When the component and eventKey fields are present, a dedupid appears similar to:

`router1.example.com|FastEthernet0/1|Perf/Interface|threshName`

When a new event comes into the system, the dedupid is constructed. If it matches the dedupid for any active event, the existing event's count field is incremented by one, and its lastTime field is updated to be the current time. If it does not match the dedupid of any active events, then it is inserted into the active event table with a count of 1, and the firstTime and lastTime fields are set to the current time.

The following illustration depicts a de-duplication scenario in which an identical event occurs three times, followed by one that is different in a single aspect of the dedupid fingerprint.

Event	dedupid	count
 device: router1 eventClass: /Perf/CPU eventKey: cpuThresh severity: 3	router1  /Perf/CPULcpuThreshold 3	1
	router1  /Perf/CPULcpuThreshold 3	2
	router1  /Perf/CPULcpuThreshold 3	3
 device: router1 eventClass: /Perf/CPU eventKey: cpuThresh severity: 4	router1  /Perf/CPULcpuThreshold 4	1

Figure 7.1. Event De-Duplication

If you want to change the way de-duplication behaves, you can use an event transform to alter one of the fields used to build the dedupid. You also can use a transform to directly modify the dedupid field, for more powerful cross-device event de-duplication.

### 7.1.5. Auto-Clear Correlation

The auto-clearing feature is similar to the de-duplication feature. It also is based on the event's fingerprint. The difference is which event fields make up the fingerprint, and what happens when a new event matches an existing event's fingerprint.

All of the standard events created as a result of polling activities do auto-clearing by themselves. As with de-duplication, you would invoke auto-clearing manually only to handle events that come from other sources, such as syslog, a Windows event log, or SNMP traps.

The auto-clear fingerprint for an event is built by using the combination of these fields:

- device
- component (can be blank)
- eventKey (can be blank)
- eventClass (including zEventClearClasses from event class configuration properties)

When a new event comes into the system with a special 0 (Clear) severity, Zenoss checks all active events to see if they match the auto-clear fingerprint of the new event. All active events that match the auto-clear fingerprint are moved from the active events table to history, and their clearid field is set to the evid of the event that cleared them.

If an event is cleared by the clear event, it is also inserted into the event history; otherwise, it is dropped. This is done to prevent extraneous clear messages from filling your events database.

The following illustration depicts a standard ping down event and its associated clear event.

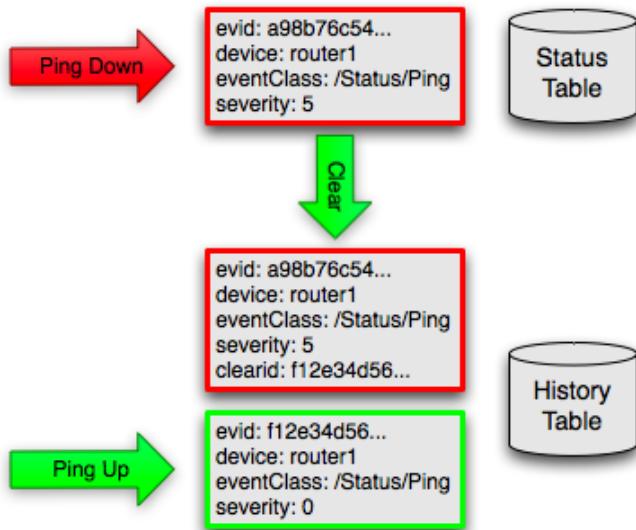


Figure 7.2. Event Auto-Clear

If you need to manually invoke the auto-clearing correlation system, you can use an event transform to make sure that the clear event has the 0 (Clear) severity set. You also need to ensure that the device, component, and eventClass fields match the events you intend to clear.

## Note

Avoid making clear events too generic; otherwise, you may inadvertently clear a wider variety of events that you intend.

### 7.1.6. Event Consoles

Zenoss features multiple event consoles that allow you to view and manage events. Each console shows different events subsets, depending on your current context.

Event consoles are:

- **Master** - To access this console, click Events on the navigation bar. You can view all events from this console.
- **Custom** - Users can create custom event consoles from the Event Views selection in user preferences. Each custom event console has access to the same events as the global console, but can be filtered more specifically.
- **Contextual** - Contextual event consoles are found throughout the system. Each time you see an Events selection for a device, device organizer, component, or event class, you can view event information that has been automatically filtered to show events specific to the current context.

#### 7.1.6.1. Master Event Console

The master event console is the system's central nervous system, enabling you to view and manage events. It displays the repository of all events that have been collected by the system.

Figure 7.3. Event Console

The following procedures provide more information about using the event console to:

- Select, sort and filter events
- Work with live search
- Save or refresh a view
- View event details
- Acknowledge events
- Return events to new status
- Classify events
- Move events to history or return them to active status

- Export event data
- Create events

#### 7.1.6.1.1. Selecting Events

To select one or more events in the event console, you can:

- Click a row to select a single event
- Ctrl-Click rows to select multiple events, or Shift-Click to select a range of events
- Click Select to select all, none, new, acknowledged, or suppressed events

#### 7.1.6.1.2. Sorting and Filtering Events

To customize your view, you can sort and filter events by any column that appears in the master event console.

To sort events, click a column header. Clicking the header toggles between ascending and descending sort order.

Filter options appear below each column header.

The screenshot shows the Event Console interface. At the top, there's a toolbar with buttons for Select, Export, and Configure. Below the toolbar is a header row with columns: Status, Severity, Device, Component, Event Class, and Summary. Under the Status column, there are three filter dropdowns labeled '...', '...', and '...'. The first dropdown has 'New' selected, the second 'Acknowledged', and the third 'Suppressed'. The main list area contains 12 event entries. Each entry includes a status icon (green for New, blue for Acknowledged, grey for Suppressed), a device name, a component name, an event class, and a detailed message. For example, the first entry is 'bibeault.zenoss /Status/Ping ip 10.87.209.23 is down'.

Figure 7.4. Event Console Filter Options

You can filter the events that appear in the list in several ways, depending on the field type. Date fields (such as First Seen and Last Seen) allow you to enter a value or use a date selection tool to limit the list. For other fields, such as Device, Component, and Event Class, enter a match value to limit the list.

The Count field allows you to filter the list when compared to a value:

- $n$  - Displays events with counts greater than or equal to that value.
- $<n$  - Displays events with counts less than that value.
- $\leq n$  - Displays events with counts less than or equal to that value.
- $=n$  - Displays events with counts equal to that value.

To clear filters, select **Configure > Clear filters**.

#### 7.1.6.1.3. Working with Live Search

By default, the system uses a "live search" feature to help you locate information. From the event console, you can search for information by:

- Device (name or IP address)

- Component
- Event class
- Summary

With live search enabled (the default behavior), the system filters available information immediately. It presents increasingly refined information with each character you type in the search window. When disabled, search responds only after you enter one or more characters and then press Enter.

To disable or enable the live search feature, select or de-select the Enable live search option in the Configure list of options.

#### 7.1.6.1.4. Saving an Event Console View

You can save your event console view by bookmarking it for quick access later. To do this:

1. Select **Configure > Save this configuration.**

A dialog containing a link to the current view appears.

2. Click and drag the link to the bookmarks link on your browser's menu bar.

A link titled "Event Console" appears in your bookmarks list.

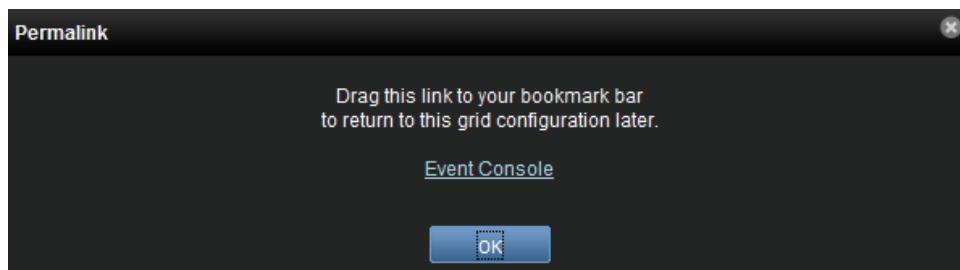


Figure 7.5. Saving a Custom View (Bookmark)

**Tip:** You may want to re-title the bookmark, particularly if you choose to save more than one event console view.

#### 7.1.6.1.5. Refreshing the View

You can refresh the list of events manually or specify that they refresh automatically. To manually refresh the view, click **Refresh**. You can manually refresh at any time, even if you have an automatic refresh increment specified.

To set up automatic refresh, select one of the time increments from the Refresh list.

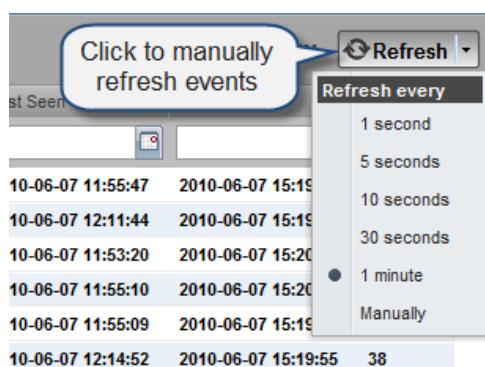


Figure 7.6. Automatic Refresh Selections

### 7.1.6.1.6. Viewing Event Details

You can view details for any event in the system. To view details, double-click an event row.

**Tip:** Be sure not to click other links in the row. These go to other pages.

The Event Details area appears.

localhost zenwin heartbeat failure

**Device:** zenosst.zenoss.loc [?](#)

**Component:** zenwin [?](#)

**Event Class:** /Status/Heartbeat [?](#)

**Status:** 0

**Start Time:** 2009/09/08 10:22:53.000

**Stop Time:** 2009/09/08 14:02:35.000

**Count:** 219

Show more details... Click for more event details.

LOG

ADD

Figure 7.7. Event Details

To see more information about the event, click Show more details. To display the event information in a new window, click the icon located at the top right.

You can use the Log area to add specific information about the event. Enter details, and then click **Add**.

### 7.1.6.1.7. Acknowledging Events

You may want to mark an event as "acknowledged" to indicate, for example, that you have taken action to remedy a problem. To mark events as acknowledged:

1. Select one or more events in the event console view.
2. Click

A check mark appears for each acknowledged event.

### 7.1.6.1.8. Returning Events to New Status

You may want to return a previously acknowledged event to "new" status (revoke its "acknowledged" status). To do this:

1. Select one or more events in the event console view.
2. Click

A check mark no longer appears in the event row, and the event is returned to "new" status.

### 7.1.6.1.9. Classifying Events

Classifying events lets you associate events shown as **/Unknown** with a specific event class. To classify events:

1. Select one or more **/Unknown** events in the event console view.

2. Click .

The Classify Events dialog appears.

3. Select an event class from the list of options, and then click **Submit**.

### Note

You can also classify events from event history.

#### 7.1.6.1.10. Moving Events to History (Close)

When you no longer want to actively monitor event (such as after you acknowledge it, for example), you can move it to history. To do this:

1. Select one or more events in the event console view.
2. Click .

The selected events are moved to history.

To view events in history, click the Event History link (located at the bottom left of the Event Console page).

#### 7.1.6.1.11. Returning Events to Active Status

You can return events that have been moved to history to active status. When you do this, the events reappear in the event console.

To return events in history to active status:

1. Click Event History to go to the event history page.
2. Select one or more events.
3. Click .

The selected events are returned to active status and appear in the event console.

#### 7.1.6.1.12. Exporting Event Data

You can export data from the event console to a comma-separated value (.csv) or XML file. To do this, select **Export > CSV** or **Export > XML**. By default, the exported file is named `events.Extension`.

### 7.1.6.2. Creating Events

To create events from the event console, click .

For more information about manual event creation, see the section titled "Creating Events Manually."

### 7.1.7. Event Sources

Events come into the system in two ways. *Generated events* are created as a result of active polling. *Captured events* are transmitted by external actions into the system.

#### 7.1.7.1. Generated Events

These standard daemons are responsible for generating events. They automatically perform appropriate de-duplication and auto-clearing.

- **zenping** - Ping up/down events
- **zenstatus** - TCP port up/down events
- **zenperfsnmp** - SNMP agent up/down events, threshold events
- **zencommand** - Generic status events, threshold events
- **zenprocess** - Process up/down events, threshold events
- **zenwin** - Windows service up/down events

### 7.1.7.2. Captured Events

Captured events are those events that the system does not specifically know will occur in advance. De-duplication is performed on these events, but in some cases may need to be tuned. By default, no auto-clearing is done on captured events. Event transforms must be used to create the auto-clear correlations.

These standard daemons are responsible for collecting captured events:

- **zensyslog** - Events created from syslog messages.
- **zentrapp** - Events created from SNMP traps and informs.
- **zeneventlog** - Events created from the Windows event log.

There are a number of APIs available for submitting events into the system. For more information, see the Zenoss Developer's Guide.

Any ZenPacks you install may optionally include their own daemons. For more information, see Zenoss Extended Monitoring.

## 7.1.8. Creating Events Manually

You can manually create events. While this is not something you would do as part of normal system operation, it can be helpful when you are attempting to test mappings and transforms you have created.

### 7.1.8.1. Creating Events through the User Interface

To create events manually through the user interface:

1. Navigate to Events, and then click  (Add Event).

The Create Event dialog appears.

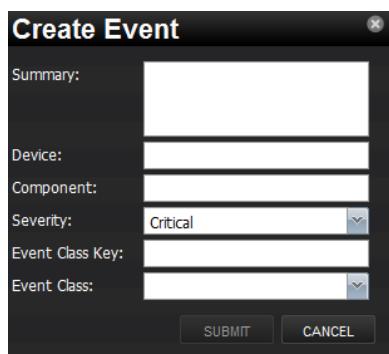


Figure 7.8. Create Event Dialog

2. Complete the basic event fields. Event class mappings are applied only for events that do not already have an event class.

### 7.1.8.2. Creating Events from the Command Line

To send events from the command line, use the `zensendevent` script, in this format:

```
zensendevent Options summary
```

Common options include:

- `-d DEVICE, --device=DEVICE`
- `-i IPADDRESS, --ipAddress=IPADDRESS`
- `-y EVENTKEY, --eventkey=EVENTKEY`
- `-p COMPONENT, --component=COMPONENT`
- `-k EVENTCLASSKEY, --eventclasskey=EVENTCLASSKEY`
- `-s SEVERITY, --severity=SEVERITY`
- `-c EVENTCLASS, --eventclass=EVENTCLASS`

#### 7.1.8.2.1. Example

The following example shows how to use the `zensendevent` script to simulate a ping down event:

```
zensendevent -d router1.example.com -s Critical -c /Status/Ping "Router down"
```

## 7.1.9. Event Classes

*Event classes* are a simple organizational structure for the different types of events that the system generates and receives. This organization is useful for driving alerting and reporting. You can, for example, create an alerting rule that sends you an email or pages you when the availability of a Web site or page is affected by filtering on the / Status/Web event class.

Following is a subset of the default event classes. You can create additional event classes as needed.

- /Status - Used for events affecting availability.
  - /Status/Ping - Ping up/down events
  - /Status/Snmp - SNMP up/down events
  - /Status/Web - Web site or page up/down events
- /Perf - Used for performance threshold events.
  - /Perf/CPU - CPU utilization events
  - /Perf/Memory - Memory utilization or paging events
  - /Perf/Interface - Network interface utilization events
  - /Perf/Filesystem - File system usage events
- /App - Application-related events.
- /Change - Events created when the system finds changes in your environment.

### 7.1.9.1. Event Class Configuration Properties

Just as device classes and devices have configuration properties, so do event classes and event class mappings. Configuration properties are applied hierarchically, with the most specific property being applied.

The following configuration properties are available on event classes and class mappings.

- **zEventAction** - How and where affected events are stored when they occur.
  - **status** - Active events table
  - **history** - Historical event table
  - **drop** - Events are not stored
- **zEventClearClasses** - Optional list of event class names whose active events will be cleared by clear events occurring in this class.
- **zEventSeverity** - The severity of affected events is changed to this value unless the Original value is used.

A good example of how the system uses the event class configuration properties is found in the /Change event class. Within the /Change event class' configuration properties, zEventAction is set to drop and zEventSeverity is set to Info. This configuration causes all of the changes in your environment to be stored as info severity events in the history table.

For more information about event manipulation techniques, see the section titled "Mapping and Transformation."

### 7.1.10. Mapping and Transformation

The event mapping and transformation system allows you to perform a wide range of operations, from altering the severity of certain events to altering nearly every field on an event, based on complex rules.

You cannot alter the following fields through event transformation. (This is because they are set after transformation has been performed.)

- evid
- firstTime
- lastTime
- count

The following illustration shows the path followed by an incoming event in the event mapping system.

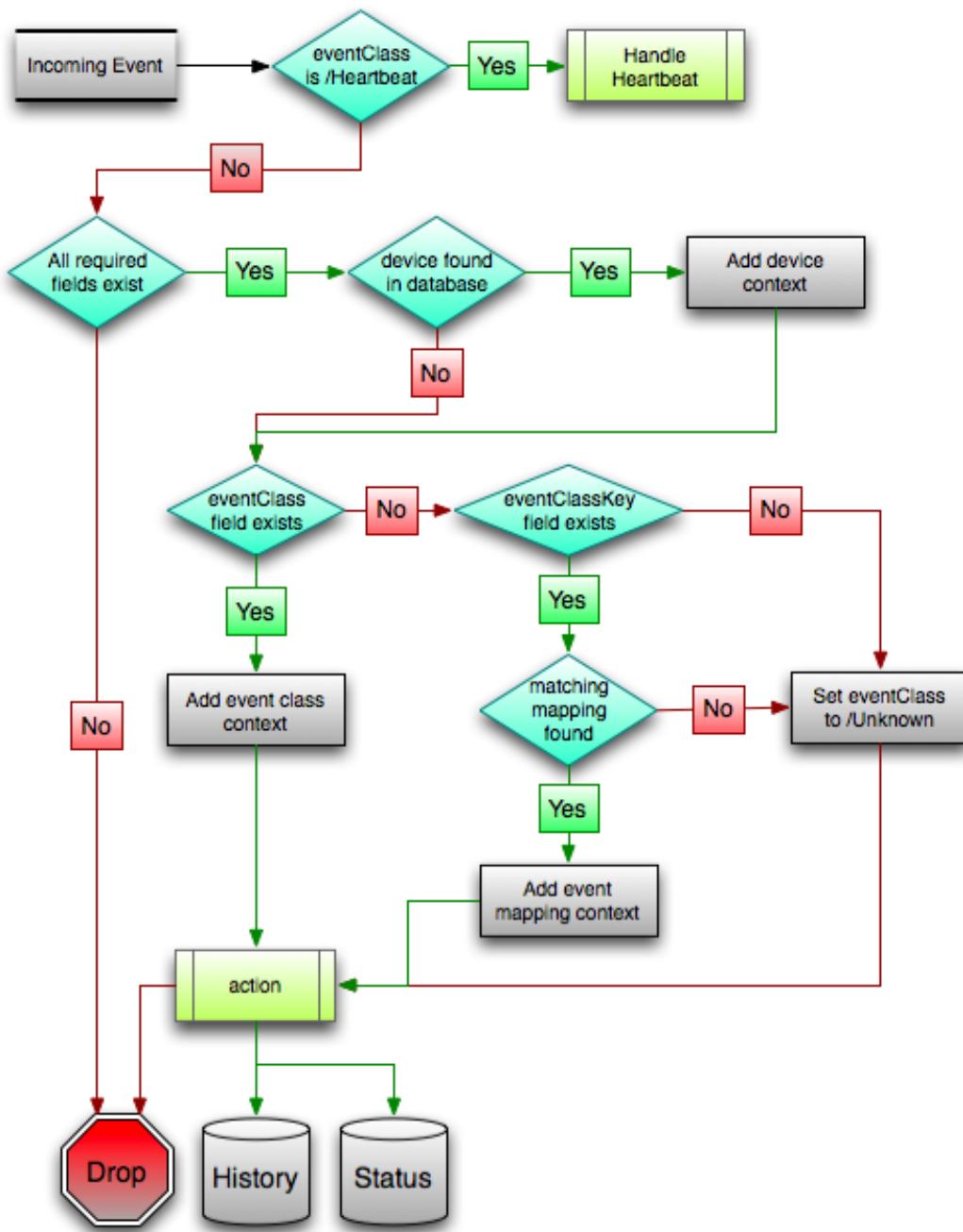


Figure 7.9. Event Processing

The mapping and transformation process begins with the "eventClass field exists" decision. This also is one of the more important differentiators in how you must handle a particular type of event.

#### 7.1.10.1. Event Class Mappings

To view event class mappings, go to Events in the navigation menu, and then select Mappings in the left panel. This allows you to see all event class mappings in a single location. The EventClass column shows which event class the mapping is in.

You can create event class mappings directly from the event classes, but does this requires that you know the eventClassKey. A simpler way to create event class mappings is through the event console:

1. Select an event that you want to match in the event console.

2. Click  (Map to event class).

The Classify Events dialog appears.

3. Select the event class to which you want to map the event, and then click **Submit**.

This creates the event class mapping with the correct eventClassKey and example text against which you can develop your regular expression.

When editing an event class mapping, you can control which events it will match, as well as other properties:

- **Event Class Key** - Must match the incoming event's eventClassKey field for this mapping to be considered as a match for events.
- **Sequence** - Sequence number of this mapping, among mappings with an identical event class key property. Go to the Sequence tab to alter its position.
- **Rule** - Provides a programmatic secondary match requirement. It takes a Python expression. If the expression evaluates to True for an event, this mapping is applied.
- **Regex** - The regular expression match is used only in cases where the rule property is blank. It takes a Perl Compatible Regular Expression (PCRE). If the regex matches an event's message field, then this mapping is applied.
- **Transform** - Takes Python code that will be executed on the event only if it matches this mapping. For more details on transforms, see the section titled "Event Class Transform."
- **Explanation** - Free-form text field that can be used to add an explanation field to any event that matches this mapping.
- **Resolution** - Free-form text field that can be used to add a resolution field to any event that matches this mapping.

The sequence area of an event class mapping allows you to provide more than one mapping for the same eventClassKey. In this case, the sequence is evaluated in ascending order until a full (rule or regex) match is found.

Mappings have the same configuration properties as event classes. Any configuration property set locally on a mapping will override the same property set on the event class. This works in the same hierarchical, most specific match, concept that device class and device configuration properties work.

When a captured event occurs, it will not have a pre-defined event class. For this type of event, you must create an event class mapping if you want to affect the event. If a captured event occurs and none of the event class mappings in the system match it, its event class will be set to /Unknown, and it will retain all of the default properties with which it began.

The next step of evaluation for events without an event class is to check on the eventClassKey field. This controls which event class mapping the event will match. If the event has a blank eventClassKey, or its eventClassKey does not match any event class mappings in the system, the special "defaultmapping" eventClassKey is searched for instead. This provides for a way to map events even if they have a blank or unpredictable eventClassKey.

### 7.1.10.2. Event Class Transform

When a generated event occurs, it has an event class assigned to it. This causes the event class mapping step to be skipped. The only way to affect the fields of one of these events is through the event class' configuration properties and transform.

To access the transform for an event class:

1. Navigate to the event class from Events > Event Classes.
2. From the Action menu, select Transform.
3. Enter information into the dialog (as Python code), and then click **Save**.

The objects available in this Python context are `evt` (the event); and, if the event matches a device that exists in the system database, a `device` object.

#### Example

The following example shows how you can validate that a device object exists before using it to drop events from a particular location.

```
if device and "Hawaii" in device.getLocationName():
    evt._action = "drop"
```

### 7.1.11. Event Life Cycle

In addition to manual methods for getting events into the status or history tables, there are automated processes that move events from status into history. The *event life cycle* is defined as all of the ways that events can be added to, moved in, and deleted from the database.

The following illustration depicts the event life cycle.

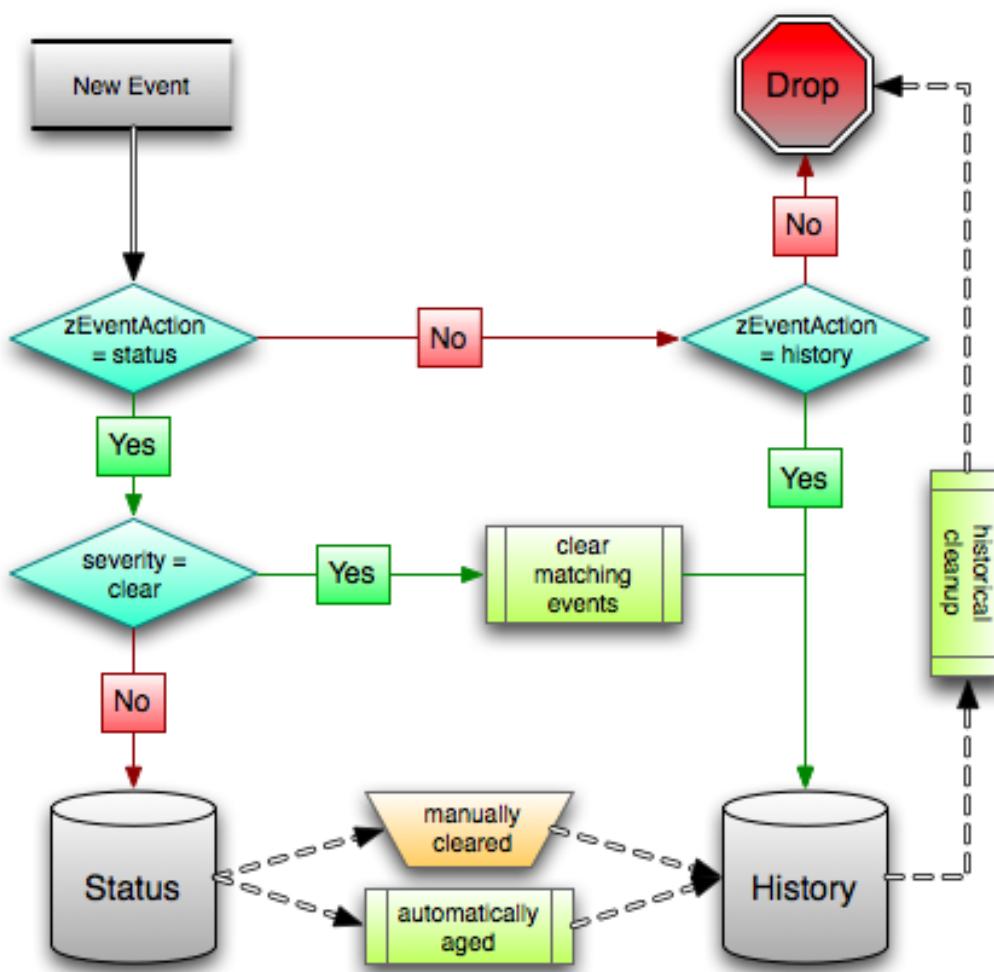


Figure 7.10. Event Life Cycle

### 7.1.11.1. Automatic Event Aging

From the event manager (Events > Event Manager), you can set up automatic aging of certain events from the status table to the history table. This allows you to have lower severity events that do not reoccur for a specified length of time to be automatically archived to the history table.

Properties that control this behavior are:

- **Event Aging Threshold (hours)** - By default, set to 4 hours.
- **Don't Age This Severity and Above** - By default, set to Error.

With the default settings, Debug, Info, and Warning events that do not occur for four hours are automatically moved into the history table.

### 7.1.11.2. Automatic Historical Event Cleanup

You can set up automatic purging of events from the history table from the event manager. When events are purged from the history table, they can be recovered only from backups. For this reason, the default setting is 0, which specifies that events are never automatically purged from history.

The event manager property that controls this behavior is Delete Historical Events Older Than (days).

## 7.1.12. Event Commands

Event commands allow the system to run arbitrary shell commands when events occur that match pre-configured criteria. This allows almost any action to be taken in response to events that occur.

Common uses of event commands include:

- *Auto-remediation of events.* You can use SSH to remotely restart services on a UNIX system when they fail, or winexe to do the same for Windows services.
- *Integration with external systems.* This includes sending SNMP traps to other management systems, or opening tickets in your incident management system.
- *Extending alerting mechanisms.* Currently, Zenoss supports only email and pagers as alerting mechanisms "out of the box" through normal alerting rules. You could use event commands to alert through instant messaging systems, or by playing sounds.

The event commands that you configure are evaluated and executed by the `zenactions` daemon once each minute (just as in alerting rules).

### 7.1.12.1. Creating Event Commands

To create or edit event commands, go to the Event Manager, and then select Commands in the left panel. From here, you can adjust these properties:

- **Enabled** - If set to True, then the command is evaluated and executed.
- **Default Command Timeout (secs)** - Length of time in seconds the system will wait for the commands to run that you specify in the Command and Clear Command fields. If the command takes longer than this, the system kills it.
- **Delay (secs)** - Specifies the minimum age (in seconds) of an event before the command will be executed on it. This prevents commands from being run for flapping events.
- **Repeat Time (secs)** - If the command runs, then it will run again in the specified seconds if the triggering event is still active. Setting this value to 0 causes the command to be run only one time.
- **Command** - Specifies the command that will be executed in the shell when the criteria specified in the Where field match an event. This command is executed as the zenoss user, and uses TALES syntax for variable substitution. (For more information about TALES, see the appendix titled "TALES Expressions.") Available variables are `evt`, `device`, and `component`.
- **Clear Command** - Similar to the Command property. This is executed only when an event that originally matched the criteria is cleared.
- **Where** - Defines the criteria that an event must match to trigger this event command.

## 7.1.13. Capturing Email Messages as Events

ZenMail and ZenPop allow you to capture email messages as events. This capability can be useful for situations in which embedded systems (such as WAPs, NAS devices, or RAID controllers) rely on email notification for events.

### 7.1.13.1. 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 server's IP address as the relay.

ZenMail supports these 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 in which an SMTP server is already running on the Zenoss server and you do not want to interfere with the existing mail delivery system. Semantics are the same as the no argument version (FROM address is used as the device IP).

### 7.1.13.2. ZenPop

ZenPop allows you to retrieve event email from a POP server. ZenPop supports these 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. Typically this is used during initial testing so that you do not have to resend test messages to the POP account. Some email systems (such as Google) do not actually delete messages when the DELE command is issued.
- `--pophost` - The hostname or IP address of the POP server from which to retrieve messages.
- `--popport` - The TCP port the POP server listens on. Defaults to 110. Used in situations where the POP provider listens on another port (for example, Google on port 995).
- `--popuser` - The user name that contains email messages to retrieve.
- `--poppass` - The password to use for the user name provided.
- `--cycletime` - The time to sleep between polls. After all email is retrieved, ZenPop sleeps for this amount of time before waking up and attempting to pull new email.

### 7.1.13.3. Translating Message Elements to the Event

Zenoss translates various message elements to the event, as follows:

- **FROM Field** - If the FROM field is an IP address, then the system associates the event with the device with the same IP address. If the FROM field is a fully qualified domain name, then the system resolves it to an IP address, and then performs the device association using the resolved IP address. The resolution of hostname uses "A" records rather than "MX" records.
- **TO Field** - The system 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 uses only the FROM field.
- **SUBJECT Field** - ZenMail and ZenPop use the SUBJECT as the event summary.
- **Message Body** - ZenMail and ZenPop use the first mime attachment as the event details. The system ignores secondary message bodies (typically HTML-encoded versions of the message). It also ignores attachments (such as files).

## 7.1.14. SNMP Traps and Event Transforms

An SNMP trap is a message that is initiated by a network element and sent to the network management system. Often, traps indicate a failure of some sort, such as a router message indicating a power supply failure, or a printer message indicating an "out-of-ink" condition.

If an SNMP trap enters the system, and Zenoss cannot identify the event (the event is classified as "/Unknown"), then you can classify the event so that the system handles it consistently.

### 7.1.14.1. Classifying SNMP Traps

To classify an SNMP trap event:

1. From the Event Console, select the unknown event or events.
2. Click  (Map to event class).

- The Classify Events dialog appears.
- Select /App, and then click **Submit**.

To edit this classification:

- From the Navigation area, select Events > Event Classes.
- In the left panel, select Mappings.
- Select the event map you created.
- In the left panel, select Edit.

The edit page appears. This page contains rules used to map the event to the /App category. This rule, since it matches the trap by a specific OID, is all that is needed.

In the Transform area, you can enter code to modify the summary. For example, if you want to set the summary string to "Spam Filter Detects Virus," then you can enter:

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

A trap has a header with some standard information, followed by a sequence of attribute/values.

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
```

However, the OID and the data is still in there. Instead, 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, as well:

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

Zenoss uses MIBs to translate SNMP traps that contain raw OID values. Loading an MIB into the system allows it to translate numeric OIDs such as .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.

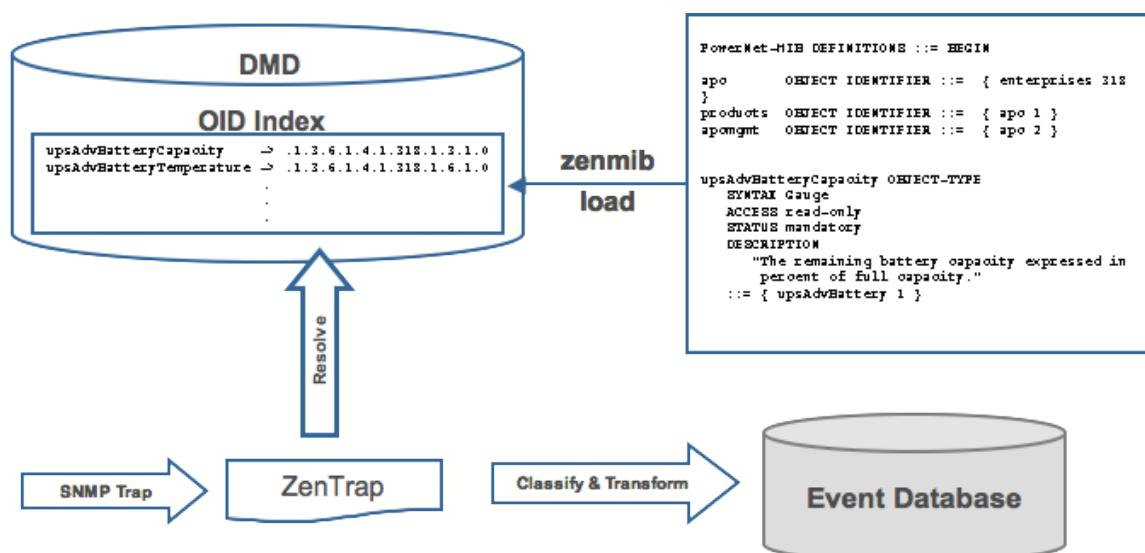


Figure 7.11. SNMP TRAP Transform

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
```

### 7.1.14.2. Example: Sending Test Traps

Follow these steps to 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 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.

Now, load some MIBs into the system so that this OID is translated into a better format:

1. Copy the demonstration MIB into \$ZENHOME/share/mibs/site.
2. Run **zenmib** to load it:

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

3. The MIB loaded, but is missing some other definitions. Copy them:

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

4. Run **zenmib** again and load the definitions into the system:

```
$ 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. 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
```

### 7.1.14.3. Transforming Events with Event Mappings

To modify events as they arrive, create an event map through the user interface:

1. Create an event class.
2. Go to the event console and create an event mapping in this class from the existing event.
3. Edit the map.
4. In the Transform area, update the event with detail data. The entry field allows you to insert Python scripts. The event is provided as "evt" and the device as "dev."

In this case, extract the sysLocation event detail and make it the summary with:

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

5. Save the event mapping.

If you move the event to history and resend the trap, the summary for the trap should now read the device name in the location you assigned.

If you encounter problems with the transform, check the zentrap.log file for errors that occurred.

### 7.1.14.4. Event Transforms Based on Event Class

When an event arrives in the system, you can change values (such as severity). For example, you can make the summary more informative, or change severity according to text within the summary.

Each event class allows for a short Python script to be executed when an event arrives.

#### Example

A user may want full file system threshold events on /data to be critical. Add 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, "evt" and "device" objects are available in the script of the transform.

---

# Chapter 8. Production States and Maintenance Windows

## 8.1. About Production States and Maintenance Windows

*Production state* determines the level of monitoring and alerting applied to an individual device. Typically, alerting rules specify that the system will monitor and create events for devices that are in the "Production" production state. *Maintenance windows* are planned time periods used to temporarily modify alerting rules so that event-generated alerts are temporarily halted.

Read this chapter for more information about defining:

- Production states
- Maintenance windows

## 8.2. Production States

Production state determines whether a device is monitored, and can be used to control several elements of the event system, such as whether an event will produce a remote alert (email or page).

Choose a production state for a device based on whether you want:

- The device to be monitored
- The device to appear on the dashboard
- Alerting to occur

The following table lists production states and their characteristics.

Production State	Devices Monitored?	Appear on Dashboard?	Remote Alerts Sent?
Production	yes	yes	yes
Pre-Production	yes	no	no
Test	yes	no	yes
Maintenance	yes	may appear	no
Decommissioned	no	no	no

Typically, devices begin in the system in "Pre-Production." In this state, devices are monitored by default, but no remote alerting occurs, and events are not shown on the dashboard. Once a device is in full "Production" state, monitoring occurs 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.

When you add a device to the system, its default state is Production.

### 8.2.1. Setting the Production State for Devices

To set the production state for a device:

1. Click a device name in the list of devices.  
The device Overview page appears.
2. Select a production state from the list of options, and then click **Save**.

To set the production state for a group of devices:

1. Select a category of devices (by class, group, system, or location) from the hierarchy.
2. From the list of options in the Production State column, select a production state.

The newly selected state is applied to all devices that appear in the list.

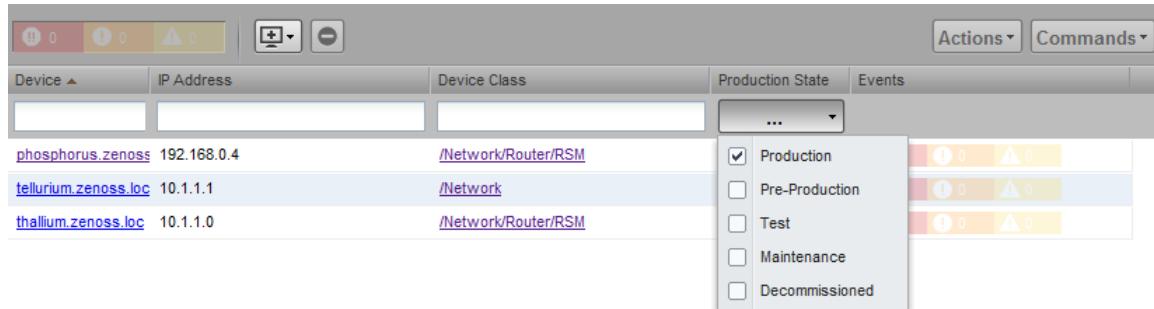


Figure 8.1. Select Production State (Multiple Devices)

## 8.3. Maintenance Windows

Maintenance windows allow scheduled production state changes of a device or all devices in a system, group, or location. You might want to set up a maintenance window, for example, to prevent alerts and warnings while you perform configuration changes or reboot a device.

### Note

In lieu of setting up a maintenance window, you can change the production state for a device manually at the time you want to make changes.

When the maintenance window starts, the production state of the device is set to the value of Start Production State (Maintenance). When the maintenance window closes, the production state of the device reverts to the value of Stop Production State (the state the device was in prior to maintenance).

### 8.3.1. Maintenance Window Events

When a maintenance window starts, an event is created with the following information:

- depuid - zenactions | *Monitor* | *MaintenanceWindowName* | *TargetOrganizerOrDevice*
- prodState - *StartProductionState*
- severity - Info
- summary/message - Maintenance window starting *MaintenanceWindow* for *Target*
- eventClass - /Status/Update
- eventClassKey - mw\_change
- maintenance\_devices - *Target*
- maintenance\_window - *MaintenanceWindow*

When a maintenance window stops, an event is created with the following information:

- severity - Clear
- summary/message - Maintenance window stopping *MaintenanceWindow* for *Target*

- prodState - -99 (meaning "unknown.")

Maintenance window events auto-clear, meaning that stop events clear start events.

## 8.3.2. Creating and Using Maintenance Windows

You can create a maintenance window for an individual device or group of devices in the devices hierarchy.

### 8.3.2.1. Create a Maintenance Window for a Single Device

To create a maintenance window for a device:

1. Click a device name in the devices list.

The device Overview page appears.

2. In the left panel, select Administration.

3. In the Maintenance Windows area, select Add Maint Window from  (Action menu).

The Add Maintenance Window dialog appears.

4. Enter a name for the maintenance window, and then click **OK**.

The newly defined maintenance window appears in the list.

5. Click the name in the list to show the maintenance window status page.

6. Define the attributes for the maintenance window:

- **Name** - Name of the maintenance window.
- **Enabled** - Select True to make the maintenance window active, or False to de-activate it.
- **Start** - Specify a date and time for the window to become active.
- **Duration** - Specify the length of time for the window to be in effect.
- **Start Production State** - Define the production state for the window when the maintenance period begins.
- **Stop Production State** - Shows the production state that will be in effect when the maintenance period ends.

7. Click **Save**.

### 8.3.2.2. Create a Maintenance Window for a Group of Devices

To create a maintenance window for a group of devices:

1. Select a group of devices from the devices hierarchy or devices list.

2. Click **Details**.

3. In the left panel, select Administration

4. In the Maintenance Windows area, select Add Maint Window from  (Action menu).

The Add Maintenance Window dialog appears.

5. Enter a name for the maintenance window, and then click **OK**.

The newly defined maintenance window appears in the list.

6. Click the name in the list to show the maintenance window status page.

7. Define the attributes for the maintenance window:

- **Name** - Name of the maintenance window.
  - **Enabled** - Select True to make the maintenance window active, or False to de-activate it.
  - **Start** - Specify a date and time for the window to become active.
  - **Duration** - Specify the length of time for the window to be in effect.
  - **Start Production State** - Define the production state for the window when the maintenance period begins.
  - **Stop Production State** - Shows the production state that will be in effect when the maintenance period ends.
8. Click **Save**.

# Chapter 9. Organizers and Path Navigation

## 9.1. About Organizers and Path Navigation

You can group system objects, including devices, sub-systems, configuration properties, and templates. A device, for example, can belong to multiple classifications, including:

- Groups
- Systems
- Locations
- Device classes

In the following illustration, the device `tilde.zenoss.loc` belongs to five different classifications. Configuration properties and monitoring settings for each of these groups are applied to this device.

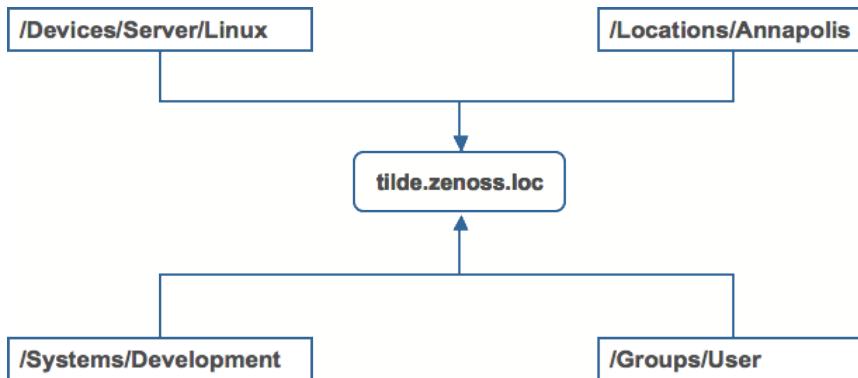


Figure 9.1. Device Groupings

Read the following sections to learn more about how organizers classify devices in the system.

## 9.2. Classes

The most important organizers are *classes*, which comprise:

- Device classes
- Event classes
- Service classes
- Product classes

Templates and configuration properties can 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 procedures are illustrated using device classes and sub-classes, but the same concepts apply to event classes, service classes, and product classes. When you add a device to the system, you should (after providing the network name or IP address) specify its device class. Templates and configuration properties can be set at any level in the device class hierarchy.

### 9.2.1. Viewing Device Classes

To view device classes and the devices they contain, select Infrastructure from the navigation bar.

The device list appears. The top of the devices hierarchy lists device classes. Click a class name to view devices in that class, or expand it to show sub-classes.

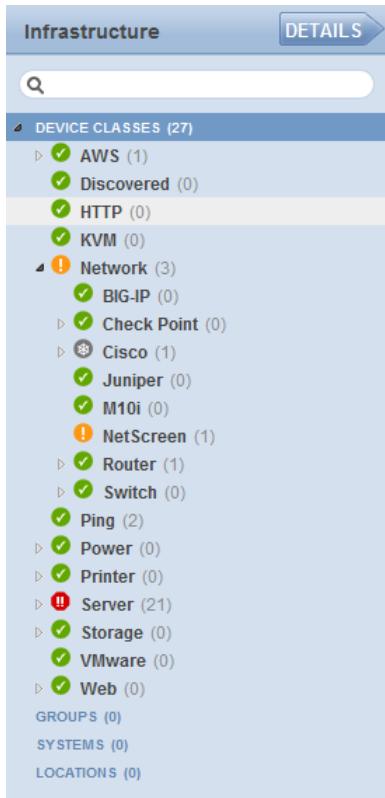


Figure 9.2. Devices Hierarchy

An indicator appears next to each listed class to show whether there are events associated with any devices in that class.

## 9.2.2. Adding Classes

To create a device class:

1. Select Infrastructure from the navigation bar.

The device list appears.

2. Select the location in the device class hierarchy where you want to add a new class.

3. Click  (Add).

The Add Device Class dialog appears.

4. Enter a name and description for the new device class, and then click Submit.

The new device class appears in the hierarchy. You can now move devices into this class. Select one or more devices in the device list, and then drag them to the new class.

### 9.2.2.1. Moving a Class

To move a class in the hierarchy:

1. Select the class in the hierarchy.

2. Drag the class to its new location.

The Move Organizer confirmation dialog appears.

3. Click **OK** to confirm the action.

The class appears at its new location in the hierarchy.

### 9.2.3. Setting Configuration Properties at the Class Level

To set configuration properties at the device class level:

1. Select a device class in the devices hierarchy.
2. Click Details, and then click Configuration Properties.

The Configuration Properties page for the selected device class appears.

zProperties Configuration		Type	Path
Property	Value		
zCollectorClientTimeout	180	int	/
zCollectorDecoding	latin-1	string	/
zCollectorLogChanges	True	boolean	/
zCollectorPlugins	Edit	lines	/Discovered
zCommandCommandTimeout	15.0	float	/
zCommandCycleTime	60	int	/
zCommandExistanceTest	test -f %s	string	/
zCommandLoginTimeout	10.0	float	/
zCommandLoginTries	1	int	/
zCommandPassword		password	/
zCommandPath	/usr/local/zenoss/libexec	string	/
zCommandPort	22	int	/
zCommandProtocol	ssh	string	/
zCommandSearchPath		lines	/
zCommandUsername		string	/
zDeviceTemplates	Device	lines	/

Figure 9.3. Device Class Configuration Properties

3. Edit configuration properties definitions as desired, and then click **Save**.

Definitions are applied to all devices currently in, and added to, this class (unless overridden at a lower level in the hierarchy).

## 9.3. Systems

Systems are intended to follow virtual setups, such as those in a network setup or systems grouped by functionality.

### 9.3.1. Adding Systems

To add a system or sub-system:

1. Select Infrastructure from the navigation bar.

The device list appears.

2. Select the location in the systems hierarchy where you want to create a system.
3. Click  (Add).

The Add System dialog appears.

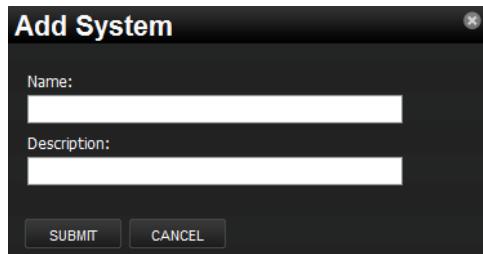


Figure 9.4. Add System

4. Enter a name and description for the system, and then click **Submit**.

The system appears in the hierarchy. You can now move devices to this system. Select one or more devices in the device list, and then drag them to the new system.

### 9.3.1.1. Moving a System

To move a system:

1. Select the system in the hierarchy.
2. Drag the system to its new location.

The Move Organizer confirmation dialog appears.

3. Click **OK** to confirm the action.

The system appears at its new location in the hierarchy.

## 9.4. Groups

Groups are functional divisions that allow you to assign attributes to multiple objects with similar functions. Groups can be used, for example, to arrange objects along departmental lines. Groups do not appear on the Dashboard.

### 9.4.1. Adding Groups

To add a group:

1. Select Infrastructure from the navigation bar.  
The device list appears.
2. Select the location in the groups hierarchy where you want to create a group.
3. Click  (Add).

The Add Group dialog appears.

4. Enter a name and description for the group, and then click **Submit**.

The group appears in the hierarchy. You can now move devices to this group. Select one or more devices in the device list, and then drag them to the new group.

#### 9.4.1.1. Moving a Group

To move a group:

1. Select the group in the hierarchy.
2. Drag the group to its new location.

The Move Organizer confirmation dialog appears.

3. Click **OK** to confirm the action.

The group appears at its new location in the hierarchy.

### 9.5. Locations

Locations are logical groupings for physical systems. They can be as general as city and state, or as specific as rack or closet. Locations do not appear on the Dashboard.

#### 9.5.1. Adding Locations

To add a location:

1. Select Infrastructure from the navigation bar.

The device list appears.

2. Select the place in the Locations hierarchy where you want to create a location.

3. Click  (Add).

The Add Location dialog appears.

4. Enter a name and description for the location, and then click **Submit**.

The location appears in the hierarchy. You can now move devices to this location. Select one or more devices in the device list, and then drag them to the new location.

#### 9.5.1.1. Moving a Location

To move a location:

1. Select the location in the hierarchy.
2. Drag the location to its new place.

The Move Organizer confirmation dialog appears.

3. Click **OK** to confirm the action.

#### 9.5.2. Integration with Google Maps

The system can map locations by using a Google Maps mashup feature that sets 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 in that location. Network connections that span locations are represented on the map by lines that also are color-coordinated appropriately to the status of that connection.

To view the Google Map for a network location:

1. Select Infrastructure from the navigation bar.

- The device list appears.
2. Select a location in the hierarchy.
  3. Click **Details**.
  4. Select Map.

The network map appears.

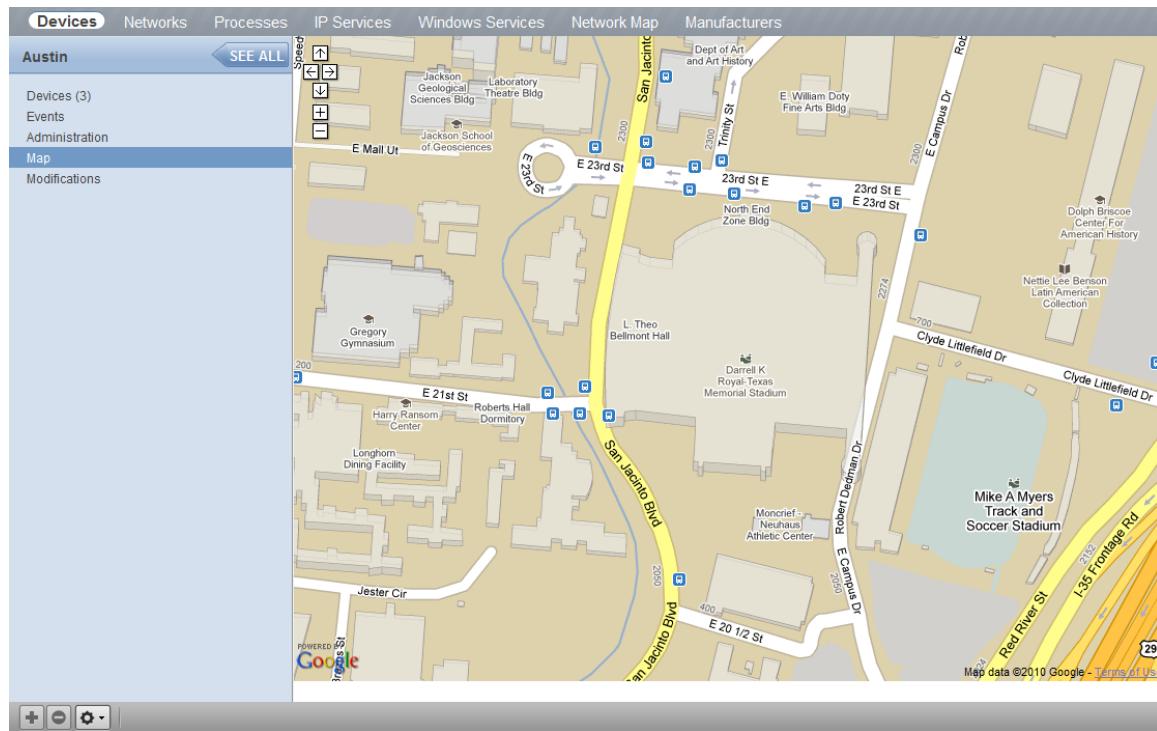


Figure 9.5. Network Location - Google Map

## Note

You also can view the network map from a dashboard portlet.

### 9.5.2.1. Obtaining an API Key

Before you can use the Google Maps feature, you must obtain or register for a Google Maps API key.

#### For Enterprise Implementations

To obtain a Google Maps API key, open a support case through the customer support portal, at:

<http://support.zenoss.com>

Be sure to provide your Zenoss server key; this is required to generate the Google Maps API key. To find your server key, select Settings, and then select the Versions tab.

#### For Core Implementations

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.

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 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.

### 9.5.2.2. Setting an Address for a Location

Follow these steps to set a location address:

1. Select Infrastructure from the navigation bar.

The device list appears.

2. Select a location in the hierarchy.
3. Select Edit from  (Action menu).

The Edit Organizer dialog appears.

4. In the Description field, enter a description for the location.
5. In the Address field, enter a complete address that can be resolved by Google Maps. (To test your address in advance, enter it at <http://maps.google.com>.)
6. Click **Submit**.

The selected address for the location is created. You must add at least one device to the location for the location "dot" to appear on the map.

### 9.5.2.3. Clearing the Google Maps Cache

Sometimes there are issues with drawing the maps and seeing the network status of locations or connections. Clearing the Geocode cache will solve these problems. To clear the Geocode cache:

1. Select Infrastructure from the navigation bar.

The device list appears.

2. Select a location in the hierarchy.
3. Select Clear Geocode Cache from  (Action menu).

A confirmation dialog appears.

4. Click **OK**.

### 9.5.2.4. Network Links

If two devices in the same network are in different map-able 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.

#### 9.5.2.4.1. Drawing Map Links (zDrawMapLinks Configuration 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. To save time, you can tell the system 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).

To edit the value for this property:

1. Select Infrastructure > Networks from the navigation bar.

The Networks page appears.

2. Select one or more networks or sub-networks for which you want to disable map links.
3. Select Set Local Value in the Draw Map Links area.
4. Select No from the list of options.

5. Click **Save**.

#### Note

This setting will be inherited by networks or sub-networks below this selection in the hierarchy. If you have few networks for which links would be drawn, you might want to disable map links on /Networks, enabling it only on a network where you know a location-spanning WAN connection exists.

#### 9.5.2.5. Google Maps Example

This example will show you how to:

- Create and display Google map links of devices
- Send a test event to see how the links are affected by system changes
  1. Disable map links. (Refer to the procedure in Drawing Map Links for instructions.)
  2. Create two locations: "New York" and "Los Angeles." (Refer to the procedure in Adding Locations for instructions.)
  3. Enter the following values in the Address field of the Add Location dialog: "New York, NY" and "Los Angeles, CA" respectively.
  4. Set the location of a device to New York. Locate another device on the same network and set its location to Los Angeles.
  5. Select Locations in the hierarchy, click **Details**, and then select Map.

New York and Los Angeles are represented by dots on the map; however, no link is drawn between these locations.

6. Select Networks and re-enable map linking.
7. Select Infrastructure, then select Locations in the hierarchy.
8. Click **Details**, and then select Map.

A green line is now drawn between New York and Los Angeles.

9. Send an event with a severity of Critical to the device in New York. (For information about creating events, refer to the chapter titled Event Management.) Do not specify a component.
10. Return to the Locations map.

The dot representing New York is now red, but the link between New York and Los Angeles remains green.

11. Navigate to the New York device and determine the ID of the component that is connected to the network shared with the Los Angeles device.
12. Send another test event, this time specifying that component.
13. Return to the Locations map.

The link between New York and Los Angeles is now red.

## 9.6. Inheritance

Inheritance is defined by how many attributes are applied to a device at different points in the device hierarchy.

The following diagram shows an example of how and where configuration properties can be set throughout the device class tree.

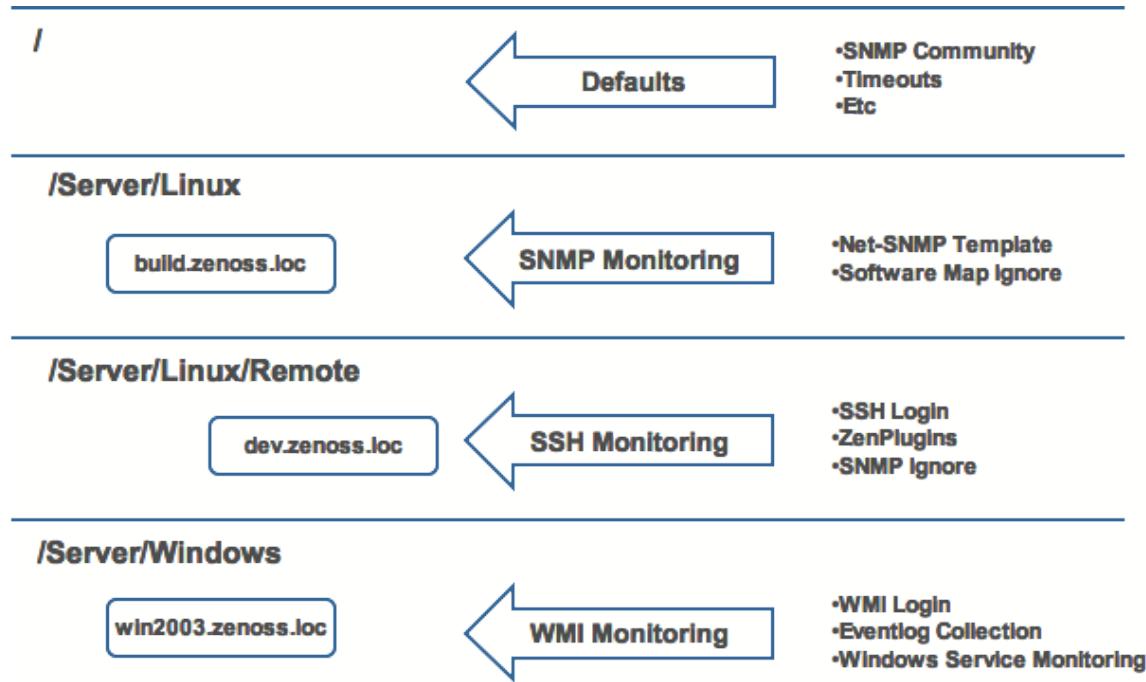


Figure 9.6. Device Class Tree and Inheritance

In this example, you can see that the default properties can be set at the highest level (/). However, as you travel further down the hierarchy, you see that you can override any of the configuration properties set at the root level.

The next two lines show how the device tree further defines properties for Linux servers. If you wanted, for example, to set up and use SNMP monitoring for all Linux servers (inclusive of build.zenoss.loc), you could change these properties at the /Server/Linux level.

Further, if you wanted to change how you collect information for remote Linux servers, you could create a sub-group in /Server/Linux called /Server/Linux/Remote, setting these servers to use SSH monitoring and changing the associated properties for that sub-group.

Also within the /Server group you could create another sub-group for Windows servers that changes the configuration properties specifically for WMI monitoring.

All of these configuration properties and groupings co-exist, with any changes made lower in the hierarchy taking priority.

---

# Chapter 10. User Commands

## 10.1. About User Commands

User commands allow you to execute arbitrary shell commands from the Zenoss master server. A user command is executed on the server rather than the remote device unless the command explicitly uses SSH to connect to the remote device.

You can define and run user commands on a device or organizer (device class, system, group, or location). You also can define commands globally.

The following sections provide procedures for defining and running:

- Global user commands
- User commands for a single device
- User commands for multiple devices in an organizer

## 10.2. Defining Global User Commands

Global commands appear in the Commands list of options located at the top of the Devices page.

To define global user commands:

1. Select Advanced > Settings from the Navigation bar.
2. In the left panel, select Commands.
3. In the Define Commands area, select Add User Command from  (Action menu).

The Add User Command dialog appears.

4. Enter a name for the command, and then click **OK**.

The Define Commands page appears.

5. In the Description field, enter a description of what the command will do.
6. In the Command section, enter the TALES expression-based command you want to run on the device.
7. Enter your system account password for confirmation, and then click **Save**.

The command is saved and added to the commands menu.

### Note

Global commands also can be edited from a specific device. Changes to a global command from a device are not limited to that device.

### 10.2.1. Running Global User Commands

To run a global user command, select one or more devices in the devices list, and then select a command from the Commands list of options.

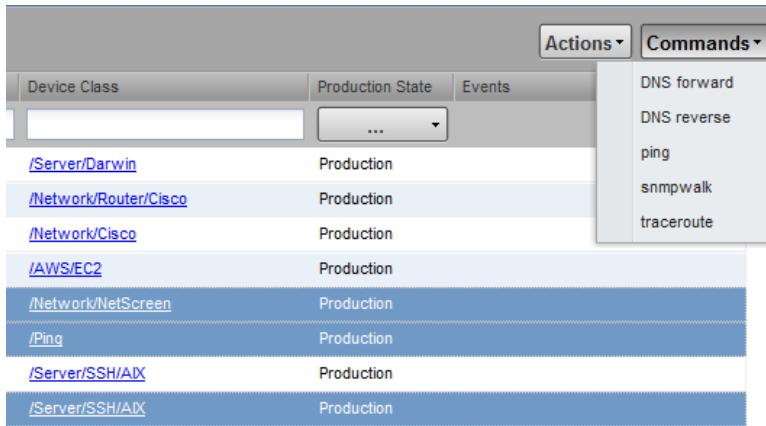


Figure 10.1. Run Commands

## 10.3. Defining User Commands for a Single Device

To define a user command for a device:

1. Select Infrastructure from the navigation bar.
2. The Devices page appears.
3. Select a device from the list.
4. In the left panel, select Administration.
5. In the Define Commands area, select Add User Command from (Action menu).

The Add User Command dialog appears.

5. Enter a name for the command, and then click **OK**.

The Define Commands page appears.

The screenshot shows the 'Define User Command' dialog box. On the left is a sidebar with 'User Command' and 'Modifications' sections. The main area has a title 'Define Commands |'. It contains four input fields: 'Name' (with 'test\_command' entered), 'Description' (empty), 'Command' (empty), and 'Confirm Your Password' (empty). At the bottom is a 'Save' button.

Figure 10.2. Define User Command

6. In the Description field, enter a description of what the command will do.
7. In the Command section, enter the TALES expression-based command you want to run.
8. Enter your system account password for confirmation, and then click **Save**.

The command is saved and added to the commands menu.

9. Optionally test the command by running it.

### 10.3.1. Running User Commands for a Single Device

To run a command defined for a single device:

1. Click the device name in the device list.

The device summary page appears.

2. Select the command from the Commands list of options located at the bottom of the page.

## 10.4. Defining User Commands for All Devices in an Organizer

To define a user command for all devices in an organizer:

1. Select an organizer in the devices hierarchy.
2. Click **Details**.
3. In the left panel, select Administration.
4. In the Define Commands area, select Add User Command from  (Action menu).

The Add User Command dialog appears.

5. Enter a name for the command, and then click **OK**.

The Define Commands page appears.

6. In the Description field, enter a description of what the command will do.
7. In the Command section, enter the TALES expression-based command you want to run on the device.
8. Enter your system account password for confirmation, and then click **Save**.

The command is saved and added to the commands menu.

9. Optionally test the command by running it.

### 10.4.1. Running User Commands for Devices in an Organizer

To run a command defined for all devices in an organizer:

1. From the devices tree view, select an organizer.
2. Select one or more devices in the filtered view.
3. Select the command from the Commands list of options located at the top of the page.

## 10.5. User Command Example: Echo Command

This example shows how to create an echo user command. You can see the use of TALES expressions in the definition of this command.

1. Add a command called "echoDevice"
2. In the command definition, echo the name and IP address of the device:

```
echo name = ${here/id} ip = ${here/manageIp}
```

In a TALES expression, 'here' is the object against which the expression is executed. Some TALES expressions in the system have other variables (such as evt for event, and dev or device for the device). See the Appendix titled TALES Expressions for more information about TALES expressions syntax.

3. Select a device and then run the command.
4. Edit the command to add more information:

```
echo name = ${here/id} ip = ${here/manageIp} hw = ${here/getHWProductName}
```

5. Run the command against a group of devices and view the command outputs.

---

# Chapter 11. Managing Users

## 11.1. About User Accounts

Each user has a unique user ID, which allows you to assign group permissions and alerting rules that are unique to each user. Unique IDs also help ensure secure access to the system.

To create and manage user accounts, you must be logged in to the system admin account, or as a user with extended privileges.

Read the following sections to learn how to:

- Create and edit user accounts
- Work with user groups
- Assign users to roles
- Set up access control lists (ACLs)

## 11.2. Creating User Accounts

To create a user account:

1. From the navigation bar, select Advanced.

The Settings page appears.

2. In the left panel, select Users.

The users and groups administration page appears.

3. From  (Add Menu), select Add New User.

The Add User dialog appears.

4. In the Username field, enter a unique name for the account.
5. In the Email field, enter the user account email address. Any alerts that you set up for this user will be sent to this address.
6. Click **OK**.

The user appears in the User List.

After creating the account, edit the account to provide a password and additional user details.

## 11.3. Editing User Accounts

To access and edit user account information:

1. In the Users list, click the name of the user you want to edit.

The edit user page appears.

State at time: 2010/04/19 15:39:14

New Password	<input type="text"/>	<input type="button" value="Reset Password"/>
Confirm New Password	<input type="text"/>	
Roles	<input type="checkbox"/> Manager <input type="checkbox"/> ZenManager <input checked="" type="checkbox"/> ZenUser	
Groups	<input type="checkbox"/>	
Email	<input type="text" value="shasting@zenoss.com"/>	<input type="button" value="test"/>
Pager	<input type="text"/>	
Default Page Size	<input type="text" value="40"/>	
Default Admin Role	<input type="checkbox"/> ZenUser	
Default Admin Level	<input type="text" value="1"/>	
Network Map Start Object	<input type="checkbox"/>	
Enter your password to confirm changes <input type="text"/> <input type="button" value="Save"/>		

Figure 11.1. Edit User

2. Make changes to one or more settings:

- **New Password / Confirm New Password** - Enter and confirm a new password for the user.
- **Reset Password** - Facilitates user self-service by allowing a user to reset his password. Click to reset and email the new password to the email address associated with the user's account.
- **Roles** - Assign one or more roles (user privileges) to the user. To edit or assign roles, you must be a system Admin or be assigned the Manager role.

For more information about user roles, and for a list of available roles and the privileges they provide, see the section titled Roles in this chapter.

- **Groups** - Specify one or more groups to which this user belongs.
- **Email** - Enter the user's email address. To verify that the address is valid, click the test link.
- **Pager** - Enter the user's pager number.
- **Default Page Size** - Controls how many entries (by default) appear in tables. Enter a value for the default page size. The default value is 40.
- **Default Admin Role** - Select the default role that this user will have for administered objects associated with him.
- **Network Map Start Object** - Specify the default view for this user in the network map.

Enter your password, and then click **Save** to confirm and save your changes.

### 11.3.1. Associating Objects with Specific Users

You can associate any object in the system with a particular user, for monitoring or reporting purposes. Once associated with a user, you can then assign the user a specific role that applies to his privileges with respect to that object.

For more information about object-specific roles, see the section titled "Roles."

To create an object association:

1. From Advanced > Settings, select Administered Objects in the left panel.

The list of administered objects appears.



*Figure 11.2. Administered Objects - Add Object*

2. Select an object type from the Administered Objects Action menu. You can add:

- Device
- Device class
- System
- Group
- Location

The Add Administered Device dialog appears.

3. Specify the component you want to add as an administered object, and then click **OK**.

The object appears in the Administered Devices list for the user.

Name	Role	Level
<input checked="" type="checkbox"/> test	ZenUser	1

*Figure 11.3. Administered Objects - Objects Added*

4. Optionally, change the role that is associated for this user on this object.

#### Note

The default role assigned to the user for an administered object is specified by the Default Admin Role field on the Edit page.

5. Click **Save** to save changes.

### 11.3.1.1. Adding Administrators

You also can associate an object with a user by adding an administrator to the object. To do this:

1. Navigate to the object you want to add to the user's list of administered objects.
2. Select Administration.

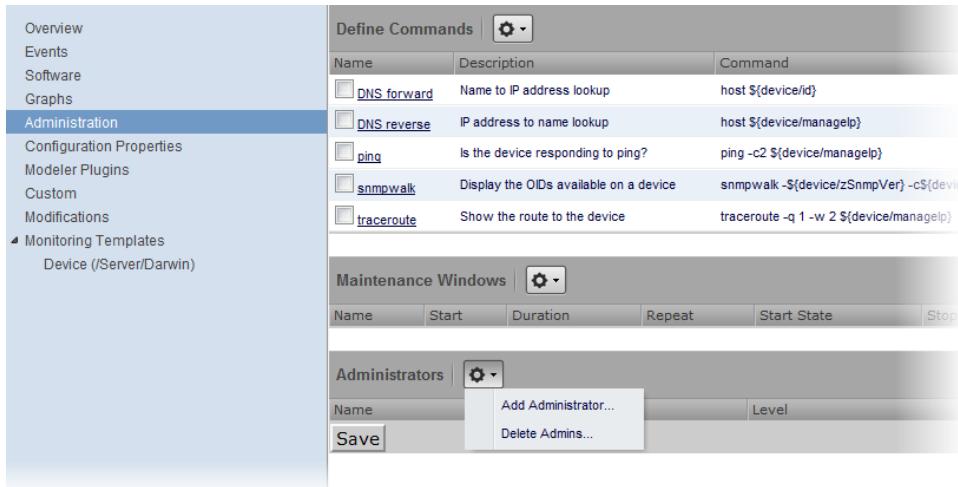


Figure 11.4. Administered Objects - Add Administrator

3. Select Add Administrator from the Actions menu in the Administrators area.

The Add Administrator dialog appears.

4. Select an administrator from the list, and then click **OK**.

The administrator appears in the object's Administrators list. The object is added to the user's Administered Objects list.

## 11.4. User Groups

Zenoss allows you to create user groups. By grouping users, you can aggregate rules and apply them across multiple user accounts.

### 11.4.1. Viewing User Groups

To view user groups, select Advanced > Settings, and then select Users from the left panel.

The groups area shows each user group and the users assigned to that group.

### 11.4.2. Creating User Groups

You can create user groups to aggregate rules and apply them across multiple user accounts.

To create a user group:

1. Select Advanced > Settings.
2. In the left panel, select Users

The Users page appears.

3. From the Groups area Action menu, select Add New Group.

The Add Group dialog appears.

4. In the Group field, enter a name for this user group, and then click **OK**.

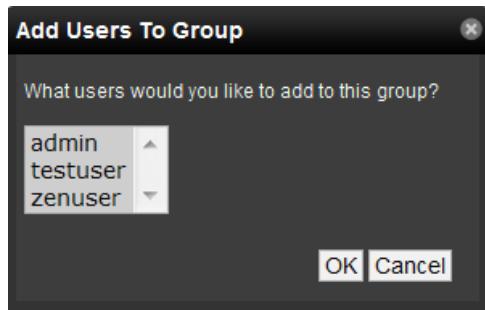
The group name appears in the Groups list.

5. Click the name of the group you created.

The Users in Group page appears.

6. From the Action menu, select Add User.

The Add User to Group dialog appears.



*Figure 11.5. Add User to Group*

7. From the User list of selections, select one or more users you want to add to the group, and then click **OK**.

The user or users you select appear in the list of users for this group.

You also can 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.

## 11.5. Roles

A role is a group of permissions that you can assign to users or groups.

The following table lists available roles.

Role	Definition
ZenUser	Provides global read-only access to system objects.
ZenManager	Provides global read-write access to system objects.
Manager	Provides global read-write access to system objects. Additionally provides read-write access to the underlying Zope object database.
ZenOperator	Installed by the ZenOperatorRole ZenPack. This ZenPack is available only to Enterprise customers. The ZenOperator role provides users the ability to manage events. Combine the ZenOperator role with the ZenUser role to allow users read-only access to the system, but also allow them to acknowledge events, move events to history, and add log messages to events.

## 11.6. Device Access Control Lists

### 11.6.1. About Device Access Control Lists

#### Note

This feature is available only with Zenoss Enterprise.

The Device Access Control List (ACL) Enterprise ZenPack (ZenDeviceACL) adds fine-grained security controls to the system. For example, this control can be used to give limited access to certain departments within a large

organization or limit a customer to see only his own data. A user with limited access to objects also has a more limited view of features 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.

## 11.6.2. Key Elements

Following are key elements of device ACLs.

### 11.6.2.1. Permissions and Roles

Actions in the system are assigned permissions. For instance to access the device edit screen you must have the “Change Device” permission. Permissions are not assigned directly to a user; instead, permissions are granted to roles, which are then assigned to a user. A common example is the ZenUser role in 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 the 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 page), it is “global.” When creating a restricted user you may not want to give that user any global role.

### 11.6.2.2. 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. The system grants access to objects is granted using the user's or user group's administered objects. To limit access, you must not assign a “global” role to the user or group.

### 11.6.2.3. 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.

### 11.6.2.4. Assigning Administered Object Access

For each user or group there is an Administered Objects selection, which lets you add items for each type of administered object. After adding an object 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 ZenManager 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 the example above, if the device was within the organizer the user would inherit the ZenManager role on the device.

### 11.6.2.5. Portlet Access Control

Within Zenoss Core, portlet access can be controlled. This is important for device ACLs.

## 11.6.3. Setup and Configuration Examples

Refer to the following examples for setup and configuration steps.

### 11.6.3.1. Restricted User with ZenUser Role

1. As admin or any user account with Manager or ZenManager role, create a user named acctest. Set a password for the user.
2. Make sure that no role is assigned to the user.
3. Edit the user's administered objects.

4. Add an existing device to the user.

The device's role will default to ZenUser.

5. Log out of your browser, or open a second browser and then log in as acltest.

6. Select Infrastructure.

You should see only the device you assigned to acltest.

7. Navigate to the device and notice that the edit capabilities are not available. This is because you are in read-only mode for this device.

### **11.6.3.2. Restricted User with ZenManager Role**

Following the example above:

1. Change the acltest user's role to "ZenManager" on the device. (You must do this as a user with ZenManager global rights.)
2. Go back to the acltest user's administered objects and set the role on the device to ZenManager.
3. As acltest, navigate back to the device. You now have access to edit the user.

### **11.6.3.3. Adding Device Organizers**

1. Go to Groups and create a group called "RestrictGroup."
2. Go to the acltest user's administered objects and add the group to the user.
3. Logged in as acltest, notice that groups can be added to a user.
4. 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.

### **11.6.3.4. Restricted User Organizer Management**

1. Give the acltest user ZenManager on your restricted group.
2. As acltest, you can now add sub-organizers under the restricted group.

### **11.6.3.5. 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.

## **11.6.4. Detailed Restricted Screen Functionality**

### **11.6.4.1. Dashboard**

By default, the dashboard is configured with three portlets:

- Object Watch List
- Device Issues
- Production State

These have content that will be restricted to objects for a given user.

### **11.6.4.2. Device List**

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

#### **11.6.4.3. Device Organizers**

Device organizers control groups of devices for a restricted user. Every device added to the group will be accessible to the user. Permissions will be inherited down multiple tiers of a device organizer.

#### **11.6.4.4. Reporting**

Reports are limited to device reports and performance reports.

# Chapter 12. Reporting

## 12.1. About Reporting

Zenoss aggregates and reports, over time, on the data you set it up to monitor. The system provides a range of defined and custom report options, including:

- Device reports
- Event reports
- Performance reports
- User reports
- Graph reports
- Multi-graph reports
- Custom device reports

To work with reports, select Reports from the navigation bar. The Reports list appears in the tree view. Expand a list item to see available reports in that category.

You can organize reports and report organizers by dragging and dropping in the tree view.

The screenshot shows the Zenoss Reports List interface. On the left, a tree view displays various report categories: REPORT CLASSES (35), Device Reports (8), Custom Device Reports (0), Graph Reports (0), Multi-Graph Reports (0), Performance Reports (7), Event Reports (3), and User Reports (1). The 'New Devices' category under 'Device Reports' is currently selected. To the right of the tree view is a table titled 'New Devices' with columns: Name, Class, First Seen, Collection, and Change. The table lists numerous devices, each with a blue link to its details page. At the bottom of the interface are standard window control buttons (minimize, maximize, close).

Name	Class	First Seen	Collection	Change
gatecolo.zenoss.loc	/Network/NetScreen	2010/06/07 12:09:46	2010/06/07 23:56:59	2010/06/07 23:56:59
talk.google.com	/Ping	2010/06/07 12:09:31	1969/12/31 19:00:00	2010/06/07 12:09:31
mail.zenoss.com	/Ping	2010/06/07 12:09:16	1969/12/31 19:00:00	2010/06/07 12:09:16
bbibeault.zenoss.loc	/Server/Darwin	2010/06/07 12:08:52	2010/06/07 23:56:56	2010/06/07 12:08:52
test-ubuntu804.zenoss.loc	/Server/SSH/Linux	2010/06/07 12:07:48	2010/06/07 23:56:48	2010/06/07 12:07:48
test-rhel54.zenoss.loc	/Server/Linux	2010/06/07 12:07:35	2010/06/07 23:55:57	2010/06/07 12:07:35
test-tomcat.zenoss.loc	/Server/Tomcat	2010/06/07 12:06:26	2010/06/07 23:55:56	2010/06/07 12:06:26
test-iboss.zenoss.loc	/Server/Boss	2010/06/07 12:04:52	2010/06/07 23:55:48	2010/06/07 12:04:52
test-solaris10.zenoss.loc	/Server/SSH/Solaris	2010/06/07 12:02:06	2010/06/07 23:55:39	2010/06/07 12:02:06
test-solaris9.zenoss.loc	/Server/Solaris	2010/06/07 12:01:45	2010/06/07 23:54:22	2010/06/07 12:01:45
test-aix61.zenoss.loc	/Server/SSH/AIX	2010/06/07 12:00:23	2010/06/07 23:54:16	2010/06/07 12:00:23
test-aix53-1.zenoss.loc	/Server/SSH/AIX	2010/06/07 11:59:06	2010/06/07 23:53:53	2010/06/07 11:59:06
test-hpux.zenoss.loc	/Server/SSH/HP-UX	2010/06/07 11:58:22	2010/06/07 23:53:08	2010/06/07 11:58:22
colo3560q.zenoss.loc	/Network/Cisco	2010/06/07 11:57:54	2010/06/07 23:52:46	2010/06/07 11:57:54
colo2800.zenoss.loc	/Network/Router/Cisco	2010/06/07 11:57:32	2010/06/07 23:52:44	2010/06/07 11:57:32
test-win2008-ad.zenoss.loc	/Server/Windows/WMI/Active Directory/2008	2010/06/07 11:57:00	2010/06/07 23:52:43	2010/06/07 11:57:00
test-exch2003-1d.zenoss.loc	/Server/Windows/WMIMSEExchange	2010/06/07 11:56:25	2010/06/07 23:52:42	2010/06/07 11:56:25
test-exch2007-1d.zenoss.loc	/Server/Windows/WMIMSEExchange/2007	2010/06/07 11:55:48	2010/06/07 23:52:25	2010/06/07 11:55:48
test-sql2005-1d.zenoss.loc	/Server/Windows/WMIMSSQLServer	2010/06/07 11:55:10	2010/06/07 23:52:10	2010/06/07 11:55:10
test-sql2005-1d.zenoss.loc	/Server/Windows/WMIMSSQLServer	2010/06/07 11:54:46	2010/06/07 23:51:06	2010/06/07 11:54:46
test-win2003-1d.zenoss.loc	/Server/Windows/WMI	2010/06/07 11:54:32	2010/06/07 23:51:01	2010/06/07 11:54:32
test-win2008-1d.zenoss.loc	/Server/Windows/WMI	2010/06/07 11:53:56	2010/06/07 23:50:59	2010/06/07 11:53:56
test-winxp-1.zenoss.loc	/Server/Windows/WMI	2010/06/07 11:53:17	2010/06/07 23:50:52	2010/06/07 11:53:17
test-winvista-1.zenoss.loc	/Server/Windows/WMI	2010/06/07 11:52:59	2010/06/07 23:50:46	2010/06/07 11:52:59

Figure 12.1. Reports List

### 12.1.1. Organizing Reports

You can create report organizers at multiple levels. Select an existing organizer or the top of the reports hierarchy (Report Classes), and then select Add Report Organizer from (Add Menu).

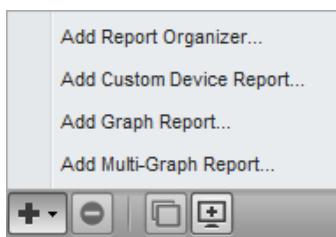


Figure 12.2. Add Report Organizer

## 12.2. Basic Reports

Basic reports included in the system are divided among:

- Device reports
- Event Reports
- Performance Reports
- User reports

The following sections provide brief descriptions of reports offered in these categories.

### 12.2.1. Device Reports

Device reports aggregate and display data over system devices and device attributes. Reports in this category include:

- **All Devices** - Lists all devices with ping and SNMP status information.
- **All Monitored Components** - Shows all of the components currently being monitored. This does not include all components in the system, only those on which the system is currently collecting performance data. The information that appears in this report is:
  - Device name
  - Component
  - Type of component (when available)
  - Description
  - Status of each device
- **Device Changes** - Shows information about the history of any changes that the system detects when modeling each device. This report shows only devices with changes. Information available in this report includes:
  - Device name
  - Device class
  - Time when the device was first seen
  - Last time the system collected data on the device
  - Any changes made to configuration at last data collection
- **Model Collection Age** – Lists devices that have not, but should have been, modeled during the past 48 hours. Information available in this report includes:
  - Device name
  - Device class
  - Time when the device was first seen by the system

- Last time the system collected data on the device
- Any changes made to configuration at last data collection
- **New Devices** – Lists devices that were recently added. The report shows:
  - Device name
  - Device class
  - Time when the device was first seen by the system
  - Model collection age
- **Ping Status Issues** – Lists devices that currently have, or have had, ping issues. The report shows:
  - Device name
  - Device class
  - Product name
  - Current state of the device
  - ping and SNMP status
- **SNMP Status Issues** - Lists devices that currently have, or have had, SNMP issues. The report shows:
  - Device name
  - Device class
  - product name
  - Current state of the device
  - ping and SNMP status
- **Software Inventory** - Lists software running on devices. This report includes:
  - Manufacturer
  - Product
  - Count

## 12.2.2. Event Reports

Event reports aggregate data about events, event mappings, and event classes.

- **All Heartbeats** – Shows heartbeats for monitored devices. Heartbeats are reported by component and number of seconds.
- **All Event Classes** – Shows all of the event classes that reside in the system. The report breaks these classes down by sub-classes, 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** – Shows all of the event mappings you have created in the system. You can sort the report by event class key, evaluation, or number of events for each event class.

## 12.2.3. Performance Reports

Performance reports provide performance data across the system.

- **Aggregate Reports** – Show the performance graphs for devices in the system, in graphical format. Common performance statistics include:
  - PU usage
  - Aggregate free memory

- Aggregate free swap
- Network output and input

Click the graph to edit graph parameters. You can:

- Change the values for width, height, Min Y, and Max Y axis.
- Specify which devices are aggregated
- Set the time span for the graph
- **Availability Report** – Shows the percentage of time that a device or component is considered available. You can filter this report on device, component, event class, or severity. You also can further limit the time frame for the availability.

The value for a date range is calculated by summing the duration of all events of a particular class with a production state of "Production" and with a severity greater than or equal to a specified severity. This sum is then divided by the duration of the time range, and then subtracted from 1 and multiplied by 100 to get the percent available, as in:

$$(1 - \text{Total event down time}) / (\text{date range}) * 100$$

#### Note

Events that occur only once are not used in calculating device availability. Specifically, events whose firsttime and lasttime fields are the same are not used in the calculation. These could represent an event that occurs and is subsequently cleared by the next event, or an event that has happened only once in the specified date range.

- **CPU Utilization Report** – Shows monitored interfaces, devices, load average, and % utility. You can customize start and end dates, and summary type (average or maximum).

This report uses data point aliases. (For more information about data point aliases, see "Data Point Aliases" in the chapter titled Core Monitoring.) To add data points to a report, add the alias, and then ensure the values return in the expected units.

Alias	Expected Units
loadAverage5min	Processes
cpu_pct	Percent

- **Filesystem Utilization Report** – Shows total bytes, used bytes, free bytes, and percentage utilization for each device. You can customize start and end dates, and summary type (average or maximum).

This report uses data point aliases. (For more information about data point aliases, see "Data Point Aliases" in the chapter titled Core Monitoring.) To add data points to a report, add the alias, and then ensure the values return in the expected units.

Alias	Expected Units
usedFilesystemSpace__bytes	bytes

- **Interface Utilization** - Shows the traffic through all network interfaces monitored by the system. Columns included in the report are:
  - Device - Interface's device.
  - Interface - Interface.
  - Speed - Interface's rated bandwidth, in bits per second.

- Input - Average traffic going out of the interface, in bits per second.
- Output - Average traffic coming in to the interface, in bits per second.
- Total - Total average traffic across the interface, in bits per second.
- % Util - Average fraction of the interface's bandwidth consumed.

This report uses data point aliases. (For more information about data point aliases, see "Data Point Aliases" in the chapter titled Core Monitoring.) To add data points to a report, add the alias, and then ensure the values return in the expected units.

Alias	Expected Units
inputOctets__bytes	bytes/sec
outputOctets__bytes	bytes/sec

- **Memory Utilization** – Provides system-wide information about the memory usage for devices in the system. This report shows:
  - Total memory
  - Available memory
  - Cache memory
  - Buffered memory
  - Percentage of memory utilized

This report uses data point aliases. (For more information about data point aliases, see "Data Point Aliases" in the chapter titled Core Monitoring.) To add data points to a report, add the alias, and then ensure the values return in the expected units.

Alias	Expected Units
memoryAvailable__bytes	bytes
memoryBuffered__bytes	bytes
memoryCached__bytes	bytes

- **Threshold Summary** – Identifies the devices that approach or exceed their thresholds. 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 also can change the start and end dates for the reporting data.

#### 12.2.4. User Reports

User Reports are based on user account information and changes in the system.

- **Notification Schedules** – Shows all of the alerting rules and their associated details.

### 12.3. Graph Reports

Graph reports allow you to assemble graphs from devices and device components into a single report.

Graph reports display only those graphs that already exist on devices or components in the system. You cannot define or alter graphs from a graph report.

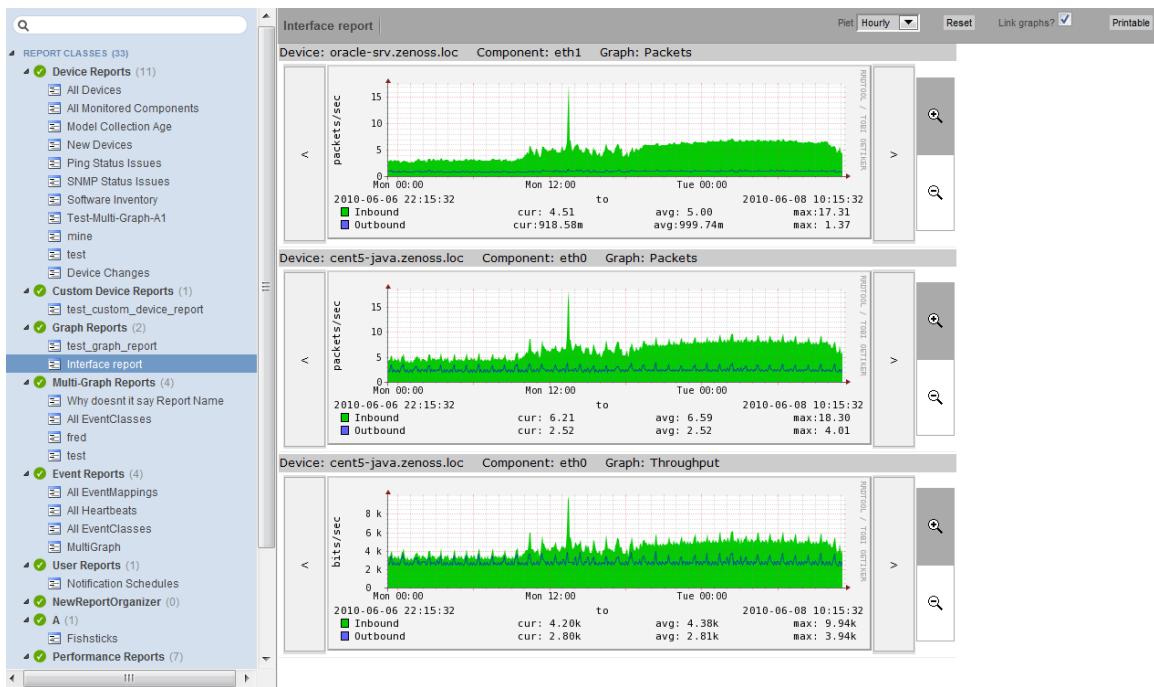


Figure 12.3. Graph Report

Graph reports are available in two views: a "normal" view (similar to the graph views for devices and device classes), and a print view.

### 12.3.1. Creating a Graph Report

Follow these steps to create a graph report:

1. In the tree view, select Add Graph Report from (Add Menu).
2. In the Create Graph Report dialog, enter a name for the report, and then click **Submit**.

The Graph Report edit page appears.

3. Enter information or make selections to define the report:

- **Name** - The name of the report, as defined in the Create Graph Report dialog.
- **Title** - Enter a descriptive title to display in the list of reports for the report organizer.
- **Number of Columns** - Specify the number of columns (1-3) in which graphs will be displayed on the report.
- **Comments** - Enter comments to display at the top of the printable version of the report. This is a TALES evaluated string that can contain HTML formatting. The variables available to the TALES expression are:
  - now (current date and time)
  - report (report object)

4. Click **Save** to save the new graph report.

#### 12.3.1.1. Adding Graphs

The Add New Graph section of the edit page allows you to add one or more graphs to the report. To do this:

1. Select one or more devices.

**Tip**

You can narrow the list of devices by entering a search string, and then clicking **Filter**.

2. Optionally, select one or more components from the Components list. This list displays the names of all components defined on at least one of the selected devices.
3. Select one or more graphs from the Graph list. This list displays the names of all graphs available for the selected devices; or, if you have selected one or more components, the graphs available for the components.
4. Click **Add Graph to Report**.

The system evaluates each selected device (or component, if selected), and searches for graphs. Matching graphs are added to the graph report. The graph is listed at the bottom of the page in the Graphs area.

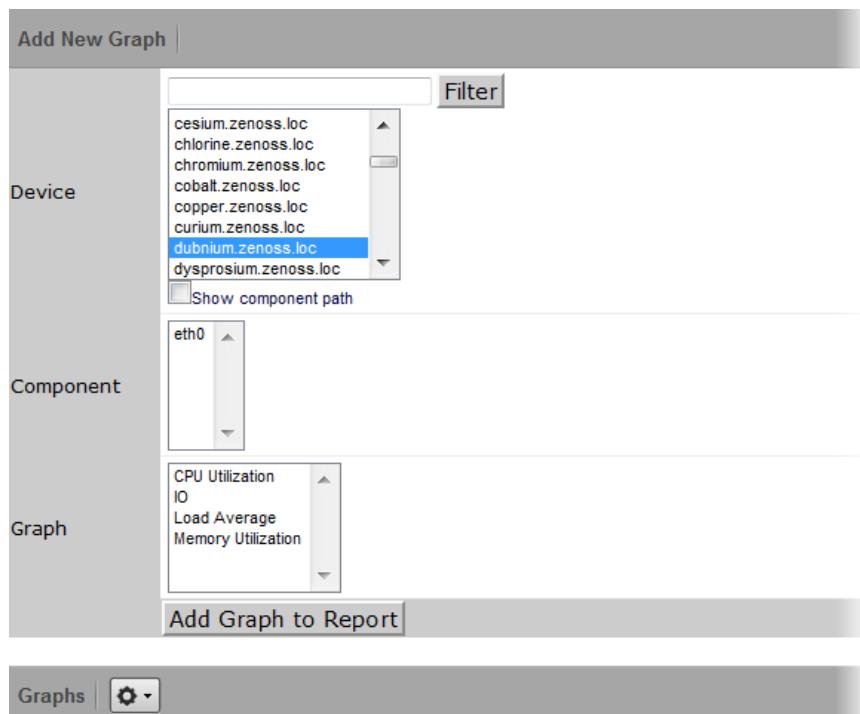


Figure 12.4. Add Graph

Graph reports maintain a static list of graphs. This list does not change when graphs are added or deleted from monitoring templates.

For example, you have two devices with associated graphs:

- DeviceA - Has a single graph (Graph1)
- DeviceB - Has two graphs (Graph1 and Graph 2)

If you select DeviceA and DeviceB on the Graph Report edit page, the list of graphs will include Graph1 and Graph2. If you select both graphs and then click Add Graph to Report, then three graphs are added to the report:

- DeviceA - Graph1
- DeviceB - Graph1
- DeviceB - Graph2

If at a later time you create a second graph (Graph2) on DeviceA's monitoring templates, that new graph will not automatically appear on the graph report. (You must edit the report to add it.) Similarly, if you later remove a graph from DeviceB's template (or delete DeviceB from the system), you must manually remove the graph (or device) from the graph report.

### 12.3.2. Customizing Graph Text

The Graphs area of the edit page lists the graphs that are included in the report. Click a graph name to view its edit page. From here, you can edit the text that appears with the graph when viewing the report:

- **Summary** - Displays above the graph in the normal report view. May contain TALES expressions with these variables:
  - dev - Device
  - comp - Component
  - graph - Graph
- **Comments** - Display to the left of the graph in the printable view. May contain TALES expressions with these variables:
  - dev - Device
  - comp - Component
  - graph - Graph

### 12.3.3. Printing Graphs

To see the print view, click **Printable** at the top of the report view. The system renders a printable view.

### 12.3.4. Organizing Graphs

In normal and print views, graphs are shown from left to right, in the number of columns specified on the report edit page. If the report contains more than three graphs, the additional graphs are ordered left to right on subsequent lines.

In the report view, you can alter the sequence of graphs in the edit page to control the location each graph appears in that view. To change the sequence, edit the values in the Seq column next to each graph name, and then choose Re-sequence Graphs.

## 12.4. Multi-Graph Reports

Multi-graph reports combine data from different devices and components into a single report. You can create a graph definition and have it drawn once for each of a group of devices and components that you define. Alternatively, you can combine the data for those graphs into a single graph.

The groups of devices and components you assemble are called *collections*. Specifying the graph definitions to apply to collections is done through graph group objects.

Multi-graph reports include their own graph definitions, and thus do not use the graph definitions that are defined in monitoring templates. (To create a report that includes graphs defined on templates, use a graph report.)

Like graph reports, multi-graph reports offer two different views: normal and print.

### 12.4.1. Creating A Multi-Graph Report

To create a multi-graph report:

1. From a report organizer or sub-organizer in the tree view, select Add Multi-Graph Report from the Add Menu.  
The Create Multi-Graph Report dialog appears.
2. Enter a name for the report, and then click **Submit**.
3. In the report edit page, enter or select values for:
  - **Name** - Displays at the top of the report.
  - **Title** - Displays at the top of the printable version of the report.
  - **Number of Columns** - Specifies the number of columns (1-3) in which graphs will appear on the report.

After creating the multi-graph report, you must also add one or more:

- Collections, which contain the devices and components you want to graph
- Graph definitions, which describe the graphs you want on the report
- Graph groups, which specify the collection and graph definition.

The screenshot displays four stacked configuration panels for a multi-graph report:

- General Report Settings:** Shows "State at time: 2010/04/14 12:31:56". Fields include Name (test\_multi-graph\_report), Title (empty), Number of Columns (1), and a Save button.
- Collections:** Shows a table with columns: Name (empty) and Number of Items (empty). A gear icon is present.
- Graph Definitions:** Shows a table with columns: Name (empty), Graph Points (empty), Units (empty), and Height (empty). A gear icon is present.
- Graph Groups:** Shows a table with columns: Seq (empty), Name (empty), Collection (empty), and Graph Definition (empty). A gear icon is present.

Figure 12.5. Multi-Graph Report Edit Page

#### 12.4.1.1. Adding Collections

A *collection* comprises one or more *collection items*. A collection item can be a list of device classes, systems, groups, locations, or specific devices or components. A single collection may contain as many collection items as desired.

A multi-graph report must contain at least one collection. Collections are shown in the Collections area of the report's edit page.

To create a collection:

1. Select Add Collection from (Action Menu).

The Add a Collection dialog appears.

2. Enter a name for the collection, and then click **OK**.

To add collection items to a collection:

1. Select a value for Item Type. The Item type menu lets you select:
  - 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.
  - Specific Device/Component - Selecting this option reveals a list of all devices. 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.
2. Select a value for Include Suborganizers. If true, then the collection will also include all organizers recursively beneath the selected organizer. These collection items are dynamic: when devices are added or removed from the organizers, they will appear or disappear from the report.
3. Click **Add to Collection** to create a new collection item for each of the selected organizers or specific device.

You can re-order collection items. Their listed order determines the order in which the graphs are drawn, or the order that data is drawn on a combined graph.

Reports > Multi-Graph Reports > test\_multi-graph\_report > test\_collection >

State at time: 2010/06/08 10:13:19

Name	test_collection				
	<input type="button" value="Save"/>				
<b>Add To Collection</b>					
Item Type	Device Class				
Device Class	<input type="checkbox"/> /Discovered <input type="checkbox"/> /Network <input type="checkbox"/> /Server <input type="checkbox"/> /Printer <input type="checkbox"/> /Power <input type="checkbox"/> /KVM <input type="checkbox"/> /Ping <input type="checkbox"/> /HTTP <input type="checkbox"/> /AWS <input type="checkbox"/> /Web				
Include Suborganizers?	<input checked="" type="checkbox"/> True				
<input type="button" value="Add to Collection"/>					
<b>Collection Items</b>					
<table border="1"> <thead> <tr> <th>Seq</th> <th>Name</th> <th>Item Description</th> <th>Number of Devices/Components</th> </tr> </thead> </table>		Seq	Name	Item Description	Number of Devices/Components
Seq	Name	Item Description	Number of Devices/Components		

Figure 12.6. Multi-Graph Report Collection

### 12.4.1.2. Adding Graph Definitions

In the context of multi-graph reports, graph definitions are very similar to those in monitoring templates. Settings on the graph definition define basic parameters; graph points are added to specify which data should be drawn. (For more information on creating graph definitions, see the chapter titled Performance Monitoring.)

The most significant differences between graph definitions in the two contexts is how data point graph points and threshold graph points are added. When adding a data point graph point to a graph definition in a performance template, you can select from a list of data points that are defined on that template. In the context of a multi-graph report, there are no graph point definitions listed; you must enter the name of the data point on the data point graph point dialog.

To add a graph definition:

1. Select Add Graph from  (Action Menu) in the Graph Definitions area of the graph edit page.

The Add a New Graph dialog appears.

2. Enter a name for the graph, and then click **OK**.

The edit page appears.

Seq	Name	Type	Description
State at time: 2010/04/14 14:45:28			
Name	test_graph		
Height	100		
Width	500		
Units			
Logarithmic Scale	False		
Base 1024	False		
Min Y	-1		
Max Y	-1		
Has Summary	True		
<b>Save</b>			

Figure 12.7. Multi-Graph Report Graph Definition

From the Action menu on this page, you can:

- Add data points and thresholds
- Delete the graph point
- Re-sequence graph points

#### 12.4.1.3. Adding Graph Groups

Graph groups combine a graph definition with a collection to produce graphs for the report. Without at least one graph group, the report will show no graphs.

To create a graph group:

1. Select Add Group from (Action Menu) in the Graph Groups area of the graph edit page.

The Add a New Graph Group dialog appears.

2. Enter a name for the graph group, and then click **OK**.
3. Enter information or make selections on the edit page:
  - **Name** - Identifies the graph group on the multi-graph report page. It does not appear on the report.
  - **Collection** - Select a collection that has been defined for this report.
  - **Graph Definition** - Select a graph definition that has been defined for this report.
  - **Method** - Choose to have the graph drawn once for each device and component in the collection, or combine the data from all devices and components into a single graph. Options are:
    - Separate graph for each device - The graph definition is used to draw one graph for each device and component in the collection. Graphs will appear in the list in roughly the same order they are specified in the collection.
    - All devices on a single graph - Draws one graph with the data from all devices and components included.
4. Click **Save** to save the graph group.

State at time: 2010/04/14 14:54:15 |

Name	<input type="text" value="test_graph_group"/>
Collection	<input type="button" value="▼"/>
Graph Definition	<input type="button" value="test_graph ▼"/>
Method	<input type="button" value="Separate graph for each device ▼"/>
<input type="button" value="Save"/>	

Figure 12.8. Multi-Graph Report Graph Group

#### 12.4.1.3.1. Graph Order

Graph groups are drawn in the order listed on the multi-graph report edit page. You can change the order of the graph groups:

1. In the Seq column of the Graph Groups area, enter numbers next to one or more graph groups.
2. Select Re-sequence items from Actions Menu.

The page refreshes and displays the graph groups in the re-sequenced order.

#### Note

If a graph group results in multiple graphs, then the graphs are drawn in the order that the collection items are listed the corresponding collection. If a collection item specifies a device organizer, then the order of the devices drawn from that collection item is indeterminate.

## 12.5. Creating Custom Device Reports

Follow these steps to create a custom device report:

1. From the Web interface, select Reports.
2. In the tree view, select Custom Device Reports.
3. From  (Add Menu), select Add Custom Device Report.

The Create Custom Device dialog appears.

4. Enter a name for the report, and then click **Submit**.

The graph edit page appears.

5. Define report parameters, as follows.
  - **Name** - Optionally edit the report name.
  - **Title** - Enter a report title. This title shows in the report display, and is distinct from the report name.
  - **Path** - Specify the path in the hierarchy where you want the system to store the report.
  - **Query** - Specify the actual query string for the report. If you want to limit the report to just those devices with a serial number, for example, you can set the Query value to:

```
here.hw.serialNumber != ""
```

- **Sort Column** - Specify the column on which you want to sort the report by default.

- **Sort Sense** - Specify the sense that the system uses to sort: asc (ascending sort) or desc (descending sort).
- **Columns** - Specify the data to be retrieved and displayed in the report. For example, you could specify:
  - getId - Gets the name of any devices.
  - getManageIp - Gets the IP addresses of the devices.
  - getHWSerialNumber - Grabs serial numbers for the devices.

#### Note

For a complete list of valid options, refer to the information on Device schema in the appendix titled "TALES Expressions."

- **Column Names** - Optionally specify column names to make the column headers more descriptive.

For the columns specified in the previous example, you could use these column names:

- Device
- Address
- Serial #

6. Click **Save** to save the report.

## 12.6. Exporting Reports

You can export data from any report as comma-separated value (.csv) files.

To export report data:

1. Click **Export All** (located at the bottom of a report).
2. In the dialog that appears, open or save the `zenoss_export.csv` file.

### 12.6.1. Advanced Task: Add An Export Button to a Report

You can edit any report to add an Export All button to export data.

The report format appears similar to 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="report"

<!-- Normal Report Markup Here -->

</tal:block>
</tal:block>
</tal:block>
```

Notable details in this example are:

- The first definition is a call to a method that retrieves the objects for the report. This might be a list, tuple, or iterable class.
- The second `tal:define` line ensures that there is a list in the event being exported. (Do not do this unless doing an export; large reports may encounter performance issues if an iterable is unnecessarily converted to a list.)
- `tablename` is defined 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 to use when not including the export functionality.
- If the Export All button does not appear to function, you may need to use `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.

## 12.7. Scheduling Reports

By default, all reports run on demand, presenting information when you run the report. To schedule reports, you can use the `reportmail` tool. This allows you to select a report and email its output to a list of recipients according to a pre-determined schedule. `reportmail` is a command line tool that is designed primarily to be run out of the UNIX crontab, allowing for flexible scheduling.

The following steps demonstrate how you would set up the standard Availability Report to be emailed to `<joe@example.com>` every Monday morning at 6 a.m.

1. Locate the URL to the Availability Report by navigating to it in the Web interface.

Device	Component	Systems
osmium.zenoss.loc		
yttrium.zenoss.loc		
silicon.zenoss.loc		
boron.zenoss.loc		
strontium.zenoss.loc		

2. Log in to your Zenoss server and switch to the `zenoss` user by running the command:

**SU - zenoss**

3. Create a script that will invoke `reportmail` with the appropriate options with the following commands:

```
mkdir -p $ZENHOME/scripts
cat <<EOF > $ZENHOME/scripts/emailAvailabilityToJoe.sh
#!/bin/sh
REPORTS_URL=http://public-demo.zenoss.com/zport/dmd/reports#reports:.zport.dmd.Reports.Performance%20Reports.

$ZENHOME/bin/reportmail \
--user=admin \
--passwd=PASSWD \
--from="zenoss@example.com" \
--address="joe@example.com" \
--subject="Zenoss: Availability Report" \
--url="$REPORTS_URL/Performance Reports/Availability Report"
EOF
```

```
chmod 755 $ZENHOME/scripts/emailAvailabilityToJoe.sh
```

### Note

For help on `reportmail` options, run `reportmail --help` on your Zenoss server.

4. Run `crontab -e` to edit the zenoss user's crontab. You will be presented with an editor that allows you to set up scheduled commands.
5. Add a new line, and then enter the following job definition:

```
# Email availability report to Joe every Monday morning at 6am.
0 6 * * 1 bash -lc '$ZENHOME/scripts/emailAvailabilityToJoe.sh'
```

### Note

Run the following command on your system server for help on `crontab` formatting:

```
man 5 crontab
```

6. Save the `crontab`.

## 12.7.1. ReportMail Command Line Arguments

The following table contains all of the command line arguments for the `reportmail` tool.

Argument	Explanation
<code>-u URL, --url=URL</code>	Uniform Resource Locator of report to send. This also can be the URL of any other page in the system.
<code>-U USER, --user=USER</code>	User to log in to the system. This user must have permission to view the supplied URL.
<code>-p PASSWD, --passwd=PASSWD</code>	Password to log in to the system.
<code>-a ADDRESS, --address=ADDRESS</code>	Email address for report delivery (may be given more than once.) Default value comes from the user's profile.
<code>-s SUBJECT, --subject=SUBJECT</code>	Subject line for email message. Default value is the title of the page.
<code>-f FROMADDRESS, --from=FROMADDRESS</code>	Origination address for the email being sent.
<code>-d DIV, --div=DIV</code>	DIV to extract from the HTML at URL. The default value is <code>contentPane</code> , which will work for all default reports.
<code>-c COMMENT, --comment=COMMENT</code>	Comment to include in the body of CSV reports. This is used only if the URL returns comma-separated value data. Most default reports can return CSV-formatted data by appending <code>?doExport</code> to the end of the URL.

## 12.8. Using Reports to Troubleshoot System Daemons

This section shows how to find and view certain reports that can aid you when troubleshooting system daemons.

From the Web interface, navigate to Reports. Use the tree view to locate the various reports listed below to see the reports. The troubleshooting items indicate what you might find in the various reports.

Troubleshooting Items	Report Name
zenmodeler issues	Model Collection Age
Internal issues	All Heartbeats
zendisc errors, adding devices	New Devices
Alerting rule issues, will show all rules in the system	Notification Schedules
Summary of SNMP status across the system, including non-monitored devices	SNMP Status Issues
Which devices are monitored and whether ping is turned on or off	Ping Status Issues

## 12.9. Advanced Reports

The EnterpriseReports ZenPack, available only for Zenoss Enterprise, adds advanced reports to the standard core reports. See Enterprise Reports in Zenoss Extended Monitoring for more details.

---

# Chapter 13. ZenPacks

## 13.1. About ZenPacks

ZenPacks extend and modify the system to add new functionality. This can be as simple as adding new device classes or monitoring templates, or as complex as extending the data model and providing new collection daemons.

You can use ZenPacks to add:

- Monitoring templates
- Data sources
- Graphs
- Event classes
- Event and user commands
- Reports
- Model extensions
- Product definitions

Simple ZenPacks can be created completely within the user interface. More complex ZenPacks require development of scripts or daemons, using Python or another programming language.

ZenPacks can be distributed for installation on other Zenoss systems.

### 13.1.1. Provided ZenPacks

A range of provided ZenPacks add and extend system functionality. These ZenPacks are grouped as Core ZenPacks (available to all users) and Enterprise ZenPacks, which are available only to Zenoss Enterprise implementations.

The guide titled *Zenoss Extended Monitoring* provides detailed descriptions, installation information, and configuration details for Core and Enterprise ZenPacks.

The following sections provide information and procedures to help you:

- Install ZenPacks
- Create ZenPacks
- Package and distribute ZenPacks
- Remove ZenPacks

## 13.2. Installing ZenPacks

ZenPacks are distributed as `.egg` files. You can install ZenPacks from the command line on the Zenoss server, or from the user interface.

### 13.2.1. Installing from the Command Line

Use these commands to install a ZenPack file and then restart the system:

```
zenpack --install <filename>
```

```
zenoss restart
```

If you have the source code for the ZenPack you can install directly from that rather than from an `.egg` file. The command is the same; however, you must specify the directory containing the source code. This copies the source code to `$ZENHOME/ZenPacks`:

```
zenpack --install <directoryname>
zenoss restart
```

If you are developing a ZenPack, you should maintain your source code outside of `$ZENHOME/ZenPacks` for two reasons:

- if you issue a `zenpack --remove` command it will delete your code from that location and you will lose your files unless you have them backed up elsewhere.
- 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 file system.

Using the `--link` option, you can install the ZenPack but have the system use your code from its current location. Instead of installing your code in `$ZENHOME/ZenPacks`, the system will create a link in that location that points to your source code directory.

```
zenpack --link --install <directoryname>
zenoss restart
```

### 13.2.2. Installing from the User Interface

To upload and install a ZenPack `.egg` file from the user interface:

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select ZenPacks.
3. From  (Action menu), select Install ZenPack.

The Install ZenPack dialog appears.

4. Browse to and select the `.egg` file you want to install, and then click **OK**.

The file is uploaded to the Zenoss server and installed.

#### Note

After installing the ZenPack, you should restart the system.

### 13.2.3. Installing All Core ZenPacks from RPM

The core ZenPacks, along with third party ZenPacks, are available for download individually from:

<http://community.zenoss.org/community/zenpacks>

At that location 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 the ZenPacks download area specific to your version.
2. Make sure ZEO is running (as the `zenoss` user):

```
zeoctl start
```

3. Install the rpm (as root user):

```
rpm -ihv <rpm file>
```

#### 4. Restart Zope and ZenHub:

```
zopectl restart
zenhub restart
```

### 13.2.4. Viewing Loaded ZenPacks

To see which ZenPacks are loaded on your system:

1. From the navigation bar, select Advanced.

The Settings page appears.

2. Select ZenPacks in the left panel.

The list of loaded ZenPacks appears.

Pack	Package	Author	Version	Egg
ZenPacks.zenoss	zenoss	Zenoss	2.0.1	Yes
ZenPacks.zenoss	zenoss	Zenoss	1.0.0	Yes
ZenPacks.zenoss	zenoss	Zenoss	1.1.3	Yes
ZenPacks.zenoss.ApacheMonitor	zenoss	Zenoss	2.1.2	Yes
ZenPacks.zenoss.BigIpMonitor	zenoss	Zenoss	2.2.2	Yes
ZenPacks.zenoss.BrocadeMonitor	zenoss	Zenoss	2.0.4	Yes
ZenPacks.zenoss.CheckPointMonitor	zenoss	Zenoss	1.0.4	Yes
ZenPacks.zenoss.CiscoMonitor	zenoss	Zenoss	2.6.0	Yes
ZenPacks.zenoss.DellMonitor	zenoss	Zenoss	2.1.0	Yes
ZenPacks.zenoss.Diagram	zenoss	Zenoss	1.1.1	Yes
ZenPacks.zenoss.DigMonitor	zenoss	Zenoss	1.0.2	Yes
ZenPacks.zenoss.DistributedCollector	zenoss	Zenoss	2.2.2	Yes
ZenPacks.zenoss.DnsMonitor	zenoss	Zenoss	2.0.2	Yes
ZenPacks.zenoss.EnterpriseCollector	zenoss	Zenoss	1.0.0	Yes
ZenPacks.zenoss.EnterpriseLinux	zenoss	Zenoss	1.1.0	Yes
ZenPacks.zenoss.EnterpriseReports	zenoss	Zenoss	2.1.2	Yes
ZenPacks.zenoss.EnterpriseSecurity	zenoss	Zenoss	1.0.0	Yes
ZenPacks.zenoss.EnterpriseSkin	zenoss	Zenoss	2.0.1	Yes
ZenPacks.zenoss.FtpMonitor	zenoss	Zenoss	1.0.2	Yes

Figure 13.1. Loaded ZenPacks

From the action menu on this page, you can create, install, and delete ZenPacks.

#### Note

Alternatively, you can view loaded ZenPacks from the command line:

```
zenpack --list
```

### 13.3. Creating ZenPacks

Read the following information and procedures to learn more about why you might want to create a ZenPack, and how to:

- Create a ZenPack
- Add a database object to a ZenPack
- View database objects in a ZenPack
- Remove a database object from a ZenPack
- Add other items to a ZenPack

### 13.3.1. Why Create a ZenPack?

Suppose you have developed a monitoring template for a new piece of hardware. You have 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 also have created a new device class for this hardware. You can create a ZenPack to easily distribute your template and device class to other administrators. This ZenPack can be entirely created from within the user interface.

As another example, suppose 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 the system. 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 the system not necessary for the previous example, but this functionality can be distributed as a ZenPack.

### 13.3.2. Create a ZenPack

Use the following instructions and guidelines to create a ZenPack.

#### Note

You must be logged in as an administrator to create a ZenPack.

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select ZenPacks.
3. From  (Action menu), select Create a ZenPack.

The Create a ZenPack dialog appears.

4. Enter the name of the ZenPack, which must be in the format:

*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.

5. Click **OK**.

The system creates the ZenPack object in the database and a new directory in the file system `$ZENHOME/ZenPacks/YourZenPackID`.

### 13.3.3. Add a Database Object to a ZenPack

To add a database object (such as a device, service, or event class, event mapping, user or event command, device organizer, or monitoring template) to a ZenPack:

1. Navigate to the object in the interface.
2. From the Action menu, select Add to ZenPack.

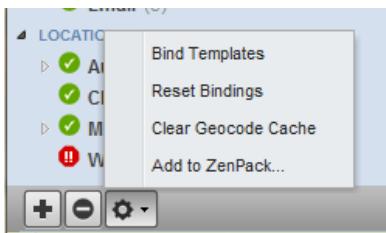


Figure 13.2. Add to ZenPack

The Add to ZenPack dialog appears.

3. Select a ZenPack from the list of installed ZenPacks, and then click **Submit**.

### 13.3.4. View Database Objects in a ZenPack

To view the objects that are part of a ZenPack:

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select ZenPacks.
3. Click the name of a ZenPack in the list.

The ZenPack Provides area of the page lists objects that are part of the ZenPack.

### 13.3.5. Remove a Database Object from a ZenPack

To remove a database object from a ZenPack:

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select ZenPacks.
3. Click the name of a ZenPack in the list.
4. Select an object in the ZenPack Provides area of the page.
5. From the Action menu, select Delete from ZenPack.

### 13.3.6. Adding Other Items to ZenPacks

ZenPacks can contain items that are not ZEO database items, such as:

- Daemons
- Data source types
- Skins

You can add these to a ZenPack by placing them in the appropriate subdirectory in the ZenPack's directory. See the Core ZenPacks at <http://community.zenoss.org/community/zenpacks> for examples of how to incorporate such items into your ZenPack.

## 13.4. Packaging and Distributing ZenPacks

Follow these steps to create an installable ZenPack .egg file:

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select ZenPacks.

3. Click the name of a ZenPack in the list.
4. From  (Action menu) located at the bottom left of the page, select Export ZenPack.

The Export ZenPack dialog appears.

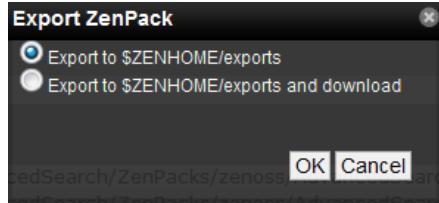


Figure 13.3. Export ZenPack

5. Select one of the export options:
  - **Export to \$ZENHOME/exports** - Exports the ZenPack to a file named `ZenPackID.egg` in the `$ZENHOME/exports` directory on the Zenoss server.
  - **Export to \$ZENHOME/exports** - Additionally downloads the exported file to your browser. Other administrators can then install this exported `.egg` file.
6. Click **OK**.

## 13.5. Removing ZenPacks

### Warning

Removing a ZenPack can have unexpected consequences. For example, removing a ZenPack that installs a device class removes the device class and all devices in that class.

Before removing a ZenPack, you should:

- Delete any data source of a type provided by the ZenPack
- Perform a backup of your system data. (See the section titled "Backup and Restore" in *Zenoss Administration* for information on backing up your system data.)

## 13.6. Where to Find More Information

Further information about ZenPack development is available in the *Zenoss Developer's Guide*.

Discussions about ZenPack development and implementation take place on the zenoss-dev and zenoss-zipacks forums:

- <http://community.zenoss.org/community/forums>
- [http://community.zenoss.org/community/developers/zenpack\\_development](http://community.zenoss.org/community/developers/zenpack_development)
- <http://zenpacks.zenoss.org>

---

# Chapter 14. General Administration and Settings

Read the following sections to learn more about performing general administration tasks, including:

- Sending alerts to email recipients or pagers
- Adjusting event manager settings
- Setting portlet permissions
- Performing backup and recovery tasks
- Viewing, starting, and stopping jobs
- Tuning and maintaining the system

## 14.1. Email and Pager Alerts

You can send alerts 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 operating system does not include a mail server, you must install one or specify a separate SMTP server in the system 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 settings.

### 14.1.1. Editing SMTP and SNPP Information

Follow these steps to edit SMTP and SNPP settings.

#### Note

You must be logged in to a user account with management privileges.

1. From the navigation bar, select Advanced.

The Settings page appears.

The screenshot shows the 'Settings' page with the following configuration values:

State at time: 2010/04/28 16:58:36	
Instance Identifier	Zenoss
SMTP Host	localhost
SMTP Port (usually 25)	25
SMTP Username (blank for none)	
SMTP Password (blank for none)	
From Address for Emails	test@test.com
Use TLS?	<input type="checkbox"/>
Page Command	\$ZENHOME/bin/zensnpp
Dashboard Production State Threshold	1000
Dashboard Priority Threshold	2

Figure 14.1. Settings

2. Change the following SMTP settings as needed:

Field	Description
SMTP Host	Set to the value of your corporate email server.
SMTP Port	Specify the port number (usually port 25).
SMTP Username	Leave this field blank.
SMTP Password	Leave this field blank.
From Address for Email	Enter a value if you want email to come from a specific email address.
Use TLS?	Select this option if you use transport layer security for your email alerts.

Table 14.1. SMTP Options

3. If you want to send pages, enter a value in the Page Command field. The pageCommand variable enables the system to execute pageCommand when a page is sent, and writes the message to the standard input of the subshell. The command prints any error messages to standard output. This enables a wider ranging of paging customization.

The standard page command is:

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

This uses ZENHOME to send the page from the localhost; however, you can use any page command customizations you prefer. In this case \$RECIPIENT is the paging address for the user, as set in the settings for each user.

## 14.2. Event Manager Settings

You can adjust settings for the Event Manager, including:

- MySQL event database connection
- Event cache timeouts and counts
- Maintenance settings

### 14.2.1. Editing Event Manager Settings

To edit event manager settings, select Events > Event Manager from the navigation bar. The event manager edit page appears.

Edit	Connection Information	
Fields	Backend Type	mysql
History Fields	User Name	<input type="text" value="root"/>
Commands	Password	<input type="password"/>
Modifications	Database	<input type="text" value="events"/>
	Hostname	<input type="text" value="127.0.0.1"/>
	Port	<input type="text" value="3306"/>
Cache		
	Cache Timeout	<input type="text" value="20"/>
	Cache Clear Count	<input type="text" value="20"/>
	History Cache Timeout	<input type="text" value="300"/>
	History Cache Clear Count	<input type="text" value="20"/>
Maintenance		
	Event Aging Threshold (hours)	<input type="text" value="4"/> Error ▾ 0 7 3
	Don't Age This Severity and Above	
	Delete Historical Events Older Than (days)	
	Default Availability Report (days)	
	Default Syslog Priority	
<a href="#" style="color: inherit; text-decoration: none;">Save Changes</a> <a href="#" style="color: inherit; text-decoration: none;">Save</a>		

Figure 14.2. Event Manager (Edit)

### 14.2.2. Changing Event Database Connection Information

To edit event database connection settings, make changes to one or more fields in the Connection Information area:

- **Backend Type** - Specifies the database type (MySQL). You cannot edit this value.
- **User Name** - Enter a user name for the MySQL database.
- **Password** - Enter the password for User Name.
- **Database** - Specify the database to use.
- **Hostname** - Specify the IP address of the host.
- **Port** - Specify the port to use when accessing the event database.

### 14.2.3. Changing Event Manager Cache Settings

To edit cache settings, make changes to one or more fields in the Cache area:

- **Cache Timeout** - Specify the cache timeout value for the event monitor. The lower the setting, the more responsive the event console will be.
- **Cache Clear Count** - Set the event count value at which the cache will be cleared of stored events.
- **History Cache Timeout** - Sets the timeout value for the History cache. The lower the number, the more responsible the history will be.
- **History Cache Clear Count** - Set the value at which history counts will be cleared.

### 14.2.4. Changing Event Manager Maintenance Settings

To edit maintenance settings, make changes to one or more fields in the Maintenance area:

- **Event Aging Threshold (hours)** - Specify how long the system should wait before aging an event into the history table.

- **Don't Age This Severity and Above** - Select a severity level (Clear, Debug, Info, Warning, Error, or Critical). Events with this severity level and severity levels above this one will not age out and be placed into event history. (These events remain in the event list until acknowledged or moved into history manually.)
- **Delete Historical Events Older Than** (days) - Enter a value in days. Zenoss will automatically purge (delete) events from the event history that are older than this value.
- **Default Availability Report** (days) - Enter the number of days to include in the automatically generated Availability Report. This report shows a graphical summary of availability and status.
- **Default Syslog Priority** - Specify the default severity level for an event to generate an entry into the syslog.

## 14.3. Setting Portlet Permissions

By setting permissions, you determine which users can view and interact with portlets. Permissions settings restrict which Zope Access Control List (ACL) can access each portlet.

Before you can successfully set portlet permissions, you must assign the user a specific Zenoss role. (You assign roles from the user edit page, from Advanced > Settings.) Each user role is mapped to one or more Zope ACL permissions, which allow you to restrict the portlets a permission level can see.

A user's specific portlet permissions are defined in part by Zope ACL permissions, and in part by the role to which he is assigned.

### 14.3.1. User Role to ACL Mapping

The following table shows how user roles map to ACLs.

User Roles	ACL Permission
ZenUser, ZenOperator	ZenCommon, View
ZenManager, Manager	ZenCommon, View, Manage DMD
No Role, Administered Obj	ZenCommon

### 14.3.2. Setting Permissions

To set portlet permissions:

1. Select Advanced from the navigation bar.

The Settings page appears.

2. In the left panel, select Portlets.

The Portlets page appears.

Available Portlets	
Device Issues	Users with ZenCommon permission ▾
Google Maps	Users with View permission ▾
Zenoss Issues	Users with Manage DMD permission ▾
Production States	Users with ZenCommon permission ▾
Site Window	Users with View permission ▾
Top Level Organizers	Users with View permission ▾
Messages	Users with ZenCommon permission ▾
Watch List	Users with ZenCommon permission ▾

**Save**

*Figure 14.3. Portlet Permissions*

3. For one or more portlets in the Available Portlets list, select the permissions you want to apply.
4. Click **Save**.

### 14.3.3. Troubleshooting: Users Cannot See All Portlets

You may mistakenly block users from being able to access some portlets. Often, this happens when a user has been set to see only particular devices. By default, this user will see only portlets set to the ZenCommon permission level. In effect, this blocks three of six portlets.

To remedy this problem, you can:

- Change the permission levels (on the Portlets page) to ZenCommon, or
- Change the user role to a role higher than "No Role."

## 14.4. Backup and Recovery

In some situations, you might want to back up configuration information and data from a Zenoss instance, and then later restore that instance. You might do this periodically, to take regular "snapshots" of your instance to archive; or infrequently, such as to move data from one instance to another, or to restore a setup after performing a fresh installation. Zenoss provides tools that enable you to manage these backup and restore tasks.

With backup and restore, the system includes:

- Events database (in MySQL)
- Zope database, which includes all devices, users, and event mappings
- `$ZENHOME/etc` directory, which contains configuration files for the system daemons
- `$ZENHOME/perf` directory, which contains performance data

Suggestions for a successful backup and restore experience:

- If you have the available disk space, tar and zip `$ZENHOME` before starting any backup or restore operation.
- Make sure the system, including all daemons, is stopped before performing a restore operation.
- Avoid using these tools to go from a newer version of Zenoss to an older version.
- If you use these tools to go from an older version to a newer version, you should run `zenmigrate` after the restore operation.

- If restoring to a different Zenoss installation (one that differs from the backup version), make sure file paths in the `$ZENHOME/etc/*.conf` files are appropriate for the new environment after you restore.

The following sections describe backup and restore scripts, as well as options for controlling their behavior.

## 14.4.1. Backup (zenbackup)

The backup script is `$ZENHOME/bin/zenbackup`. Typical use of **zenbackup** looks like:

```
> zenbackup --save-mysql-access --file=BACKUPFILEPATH
```

If the system is running then you can run **zenbackup** without any arguments. A backup file will be placed in `$ZENHOME/backups`.

### 14.4.1.1. Backup Options

The following table lists frequently used **zenbackup** options.

#### Note

Use the **zenbackup --help** command to see a complete list of **zenbackup** options.

Option	Description
<b>--dbname</b>	Specifies the name of the MySQL database the system uses to hold event data. By default this is "Zenoss" but this can be specified when the system is installed. This value can be seen by looking at the Database field on the event manager edit page. If you do not specify <b>--dbname</b> then zenbackup will attempt to retrieve this information from ZEO unless you specify <b>--dont-fetch-args</b> .
<b>--dbuser, --dbpass</b>	These are the MySQL username/password used to access the events database. If you do not specify <b>--dbuser</b> or <b>--dbpass</b> then <b>zenbackup</b> will attempt to retrieve this information from ZEO unless you specify <b>--dont-fetch-args</b> .
<b>--dont-fetch-args</b>	This instructs <b>zenbackup</b> not to attempt to get values for <code>dbname</code> , <code>dbuser</code> and <code>dbpass</code> from ZEO.
<b>--file=Filename</b>	Use <b>--file</b> to specify a location for the backup file. By default it will be named <code>zenoss_Date.tgz</code> and placed in <code>\$ZENHOME/backups</code> .
<b>--stdout</b>	This flag tells <b>zenbackup</b> to send the backup information to standard output instead of to a file. Incompatible with <b>--verbose</b> .
<b>--save-mysql-access</b>	This instructs zenbackup to save <code>dbname</code> , <code>dbuser</code> and <code>dbpass</code> as part of the backup file so that <b>zenrestore</b> will have this information during a restore operation. Use this with caution as it means your backup files will contain a MySQL user name and password.
<b>--no-eventsdb</b>	Do not include the MySQL events database as part of the backup.
<b>-v, --verbose</b>	Print progress messages. Incompatible with <b>--stdout</b> .

#### 14.4.1.2. Create a Backup

To back up your system instance from the interface:

1. From the navigation bar, select Advanced.

The Settings page appears.

2. In the left panel, select Backups.

The Backups page appears.

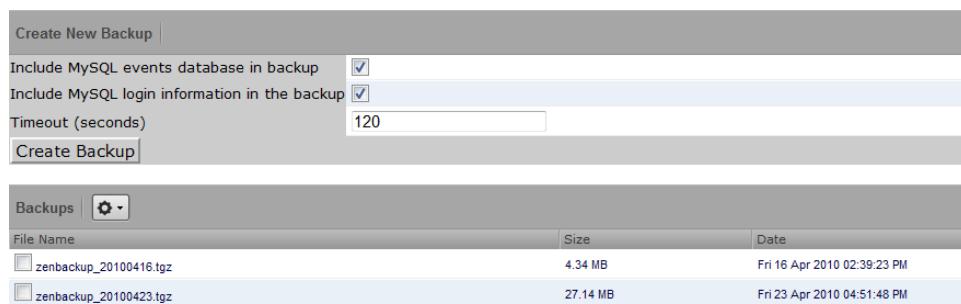


Figure 14.4. Backup

3. In the Create New Backup area, enter information or make selections for the backup. Options available are a subset of those available from the **zenbackup** command line tool.
4. Click **Create Backup**.

#### 14.4.1.3. Delete a Backup

To delete a backup from the interface:

1. From the navigation bar, select Advanced > Settings.
2. In the left panel, select Backups.

The Backups page appears. The Backups area lists all backup files in \$ZENHOME/backups.

3. Select one or more files in the list, and then select Delete Backup from the Action menu.
4. Click **Delete** in the Delete Backup dialog to confirm the action.

#### Note

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.

#### 14.4.1.4. Remote Backups

Keeping backups on your server should help you recover if one of your databases becomes corrupt or your configuration becomes problematic. However, you should keep at least one recent backup file on a different server (ideally at a different physical location) in case a physical disk fails.

#### 14.4.2. Restore (zenrestore)

The restore script is \$ZENHOME/bin/zenrestore. Typical use of **zenrestore** looks like:

```
> zenrestore --file=BACKUPFILEPATH
```

#### 14.4.2.1. Before You Restore (for Versions Earlier Than 2.4.5)

If you are running a version of Zenoss prior to 2.4.5, before you can restore your instance, you must ensure that the same ZenPacks that were installed on the backup system are also installed on the target system.

Make sure that the system 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 dbpass also.

#### 14.4.2.2. Restore Options

The following table lists frequently used zenrestore options.

##### Note

Use the **zenrestore --help** command to see a complete list of **zenrestore** options.

Option	Description
<b>--file</b>	This is a backup file created with zenbackup. You must specify either <b>--file</b> or <b>--dir</b> .
<b>--dir</b>	The path to an unzipped backup file. You must specify either <b>--file</b> or <b>--dir</b> .
<b>--dbname</b>	This is the name of the MySQL database the system uses to hold event data. This database must exist before <b>zenrestore</b> is run. If there are any system tables in the database they will be dropped by <b>zenrestore</b> 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 must set the database name on the event manager edit page.
<b>--dbuser, --dbpass</b>	These are the MySQL username/password used to access the events database. If you do not specify <b>--dbuser</b> or <b>--dbpass</b> then <b>zenrestore</b> will attempt to use values stored in the backup file if <b>--save-mysql-access</b> was used in creating it.
<b>--no-eventsdb</b>	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 <b>--no-eventsdb</b> .
<b>-v, --verbose</b>	Print progress messages.

### 14.5. Working with the Job Manager

The Job Manager runs background tasks, such as discovering a network or adding a device. When you ask the system to perform one of these tasks, it adds a job to the queue. Jobs are run by the `zenjobs` daemon.

##### Note

Not all jobs run in the Job Manager. When running other jobs (in the foreground), do not navigate away from the current page until the job completes.

#### 14.5.1. Viewing Jobs

To access the Job Manager:

- From the navigation bar, select Advanced.

The Settings page appears.

- In the left panel, select Jobs.

The jobs list appears.

The screenshot shows a table titled "Jobs" with a header row containing columns: Status, Job Type, Description, Started, Finished, Duration, and Actions. A single job entry is listed:

Status	Job Type	Description	Started	Finished	Duration	Actions
<span style="color: green;">✓ Succeeded</span>	AutoDiscoveryJob	/opt/zenoss/bin/zendisc run --now --monitor localhost --deviceclass /Discovered --parallel 8 --job fbaa7fe9-b8bb-4f44- b4b9-ced31785bafe --net 10.175.211.0/24	3 hours ago	2 hours ago	16 minutes	<span style="color: green;">Job Log</span> <span style="color: red;">Delete</span>

Figure 14.5. Jobs List

The jobs list shows information about all jobs currently in the system:

- Status** - Shows the current job status. Status options are Pending (waiting for `zenjobs` to begin running), Running, Succeeded, and Failed.
- Job Type** - Provides a short indicator of the job type.
- Description** - Provides a longer description of the job. Generally, this includes the shell command run by the `zenjobs` daemon.
- Started / Finished / Duration** - Provide information about the time period in which the job ran.
- Actions** - Shows actions you can take on the job. These include:
  - Job Log - Click to view the real-time output of a running job or final output from a completed job.
  - Stop - Ask the `zenjobs` daemon to stop running this job.
  - Delete - Remove this job from the system.

### 14.5.2. Running the `zenjobs` Daemon

You can stop and start the `zenjobs` daemon from the command line and from Advanced > Settings (Daemons selection). You also can start it (if not already running) from the Job Manager. Click the link that appears at the top right of the jobs list.

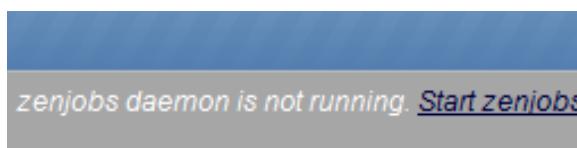


Figure 14.6. Start zenjobs

## 14.6. Maintenance and Performance Tuning

Read the following sections for maintenance and tuning suggestions.

### 14.6.1. Packing the ZEO Database

The ZEO database should 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 -h localhost
```

## 14.6.2. Log Rotate Script

The `logrotate` script must be present and running to rotate the Zope log (`event.log`), ZEO log (`zeo.log`), and access log (`z2.log`).

### 14.6.2.1. Zenoss 2.4.x

At installation, the system adds the following script to the `/etc/logrotate.d/zenoss` file (where `ZENHOME` varies depending on your installation platform):

```
**ZENHOME**/log/event.log **ZENHOME**/log/z2.log **ZENHOME**/log/zeo.log{
    missingok
    weekly
    rotate 2
    copytruncate
}
```

In version 2.4.x, each daemon performs its own log rotation. You can customize rotation parameters by inserting the following directives in each daemon's configuration file, located in the `$ZENHOME/etc` file:

Directive	Description
<code>--maxlogsize=MAXLOGKILOBYTES</code>	Specifies the maximum size of the log file, in kilobytes. By default, the maximum is 10240 KB.
<code>--maxbackuplogs=MAXBACKUPLOGS</code>	Specifies the maximum number of backup log files. By default, the maximum is 3.

If you do not specify a directive, then the system uses the default values.

### 14.6.2.2. Zenoss 2.3.3 and Earlier

The `logrotate` script should be present in your installation, in the `/etc/logrotate.d/zenoss` file. For example, for a CentOS/RHEL, RPM-based installation, the following script should be present:

```
/opt/zenoss/log/*.log /opt/zenoss/log/*/*.log {
missingok
weekly
rotate 2
copytruncate
}
```

---

# Appendix A. Daemon Commands and Options

Zenoss daemons provide the services that collect and feed data to the data layer. These daemons are divided among the following categories:

- Automated modeling
- Availability monitoring
- Event collection
- Performance monitoring
- Automated response

## A.1. Automated Modeling Daemons

Daemon	Description
zendisc	Discovers new network resources. zendisc is a subclass of zenmodeler. It walks the routing table to discover the network topology, and then pings all discovered networks to find active IP addresses and devices.
zenmodeler	Configuration collection and configuration daemon, used for high-performance, automated model population. It uses SNMP, SSH, Telnet, and WMI to collect its information. zenmodeler works against devices that have been loaded into the DMD. It models devices on a periodic schedule (typically every 12 hours).

## A.2. Availability Monitoring Daemons

Daemon	Description
zenping	Performs high-performance, asynchronous testing of ICMP status. Uses the Standard model to perform Layer 3 -aware network topology monitoring.
zenstatus	Performs active TCP connection testing of remote daemons.
zenprocess	Checks for the existence of monitored processes by using SNMP host resources' MIB, logging their CPU and memory utilization. For cases in which more than one process is matched, sums the CPU and memory and tracks the process count.

## A.3. Event Collection Daemons

Daemon	Description
zensyslog	Collects and classifies syslog events. Parses the raw format to find the level and facility, host name, and tag (the freeform message string of the event). Syslog events often have specific, proprietary formats used by vendors; zensyslog tries to parse these by using a series of regular expressions defined in it. Once parsing is complete, the event is sent back to the event system (through zenhub) to be integrated with the model.
zeneventlog	Collects Windows Management Instrumentation (WMI) event log events. Forwards these events to zenhub for further processing.
zentrap	Collects SNMP traps, parses them, resolves OIDs into MIB names, and then forwards them to zenhub for further rules processing.

## A.4. Performance Monitoring Daemons

Daemon	Description
zenperfsnmp	Performs high-performance, asynchronous SNMP performance collection and stores it locally to the collector. Thresholds are tested each time a value is written to disk.
zencommand	Performs XML RPC collection. Allows running of Nagios© and Cactii plug-ins on the local box, or on remote boxes through SSH. Commands must conform to the Nagios or Cactii API specifications.

## A.5. Automated Response Daemons

Daemon	Description
zenactions	Runs background jobs, such as email notification, database aging and maintenance window processing.

---

# Appendix B. SNMP Device Preparation

This appendix provides information about SNMP support and lists Net-SNMP configuration settings that are required by the system.

## B.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
```

## B.2. SNMP V3 Support

Zenoss provides support for SNMPv3 data collection.

The following configuration properties control the authentication and privacy of these requests:

- zSnmpAuthType - Use "MD5" or "SHA" signatures to authenticate SNMP requests.
- zSnmpAuthPassword - Shared private key used for authentication. Must be at least 8 characters long.
- zSnmpPrivType - "DES" or "AES" cryptographic algorithms.
- zSnmpPrivKey - Shared private key used for encrypting SNMP requests. Must be at least 8 characters long.
- zSnmpSecurityName - 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 zSnmpAuthType and zSnmpAuthPassword are set, then the message is sent with authentication but no privacy. If neither Priv or Auth values are set, the message is sent with no authentication or privacy. You cannot 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.

Currently, RedHat 5 and Ubuntu 7.10 do not support AES. OpenSuSE 10.2 and the Zenoss Appliance do.

You can determine if your platform supports AES by using 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 supports AES.

#### Note

SNMPv3 Traps are not supported.

## B.3. Community Information

Add these lines to your `snmp.conf` file.

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
```

## B.4. 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
```

## B.5. Extra Information

For more information, see the `snmpd.conf` manual page, and the output of the `snmpd -H` command.

```
trapcommunity public  
trapsink default
```

---

# Appendix C. Using an Existing MySQL Server to Store Events

## C.1. About

You can configure the system to store events in an existing, remote MySQL server. You might want to do this if you think you may generate too many events for a local MySQL server to handle.

## C.2. Procedure

The following steps show how an existing installation can be configured to use a specific MySQL server.

1. Initialize the new database.

As a super-privileged user on the MySQL server, create the events database schema. The `zenevents.sql` and `zenprocs.sql` files are located in `$ZENHOME/Products/ZenEvents/db` on your server. Replace 10.1.2.30 with the IP address of your server, and replace the password with your password.

From the MySQL client, run these commands:

```
CREATE DATABASE events;
\_. zenevents.sql
\_. zenprocs.sql
GRANT ALL PRIVILEGES ON events.* to zenoss@10.1.2.30 identified by 'password';
FLUSH PRIVILEGES;
```

2. Configure the system to use the new database.

- In the Web interface, navigate to Event Manager. (You must be assigned the Manager role to do this.)
- In the Connection Information section, adjust field values as needed. Replace 10.1.2.40 with the IP address of your MySQL server.
  - User Name = zenoss
  - Password = password
  - Database = events
  - Hostname = 10.1.2.40
  - Port = 3306
- Click **Save**.
- Restart the system.

### Note

The installation `--help` option lists command-line options for setting up a remote MySQL server.

---

# Appendix D. Syslog Device Preparation

## D.1. Forwarding Syslog Messages from UNIX/Linux Devices

Zenoss has its own syslog server, zensyslog. Managed devices should point their syslog daemons to the system. 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 (such as vi).
3. Enter `*.debug` and press the Tab key. Then enter the host name or IP address of the server. For example:

```
*.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
```

## D.2. Forwarding Syslog Messages from a Cisco IOS Router

Here are some links to Cisco commands to turn on syslog. Typically, it is 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
```

### D.2.1. 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 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
```

## D.3. 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 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
```

## D.4. 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 server). For example:

```
log { source(src); destination(zenoss); };
```

---

# Appendix E. TALES Expressions

## E.1. About Tales Expressions

Use TALES syntax to retrieve values and call methods on Zenoss objects. Several fields accept TALES syntax; these include:

- Command templates
- User commands
- Event commands
- zLinks

Commands (those associated with devices and those associated with events) can use TALES expressions to incorporate data from the related devices 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.

For additional documentation on TALES syntax, see the TALES section of the Zope Page Templates Reference:

[http://www.zope.org/Documentation/Books/ZopeBook/2\\_6Edition/AppendixC.stx](http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/AppendixC.stx)

Depending on context, you may have access to a device, an event, or both. Following is a list of the attributes and methods you may want to use on device and event objects. The syntax for accessing device attributes and methods is \${dev/attributename}. 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 ${device/manageIp}
```

### E.1.1. 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} ${device/manageIp} system
```

To use these expressions effectively, you must know which objects, attributes, and methods are available, and in which contexts. Usually there is a device that allows you to access the device in a particular context. Contexts related to a particular event usually have event defined.

## E.2. TALES Device Attributes

The following table lists available device attributes.

Attribute	Description
getId	The primary means of identifying a device within the system

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 Decommissioned. 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 system 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

## E.3. Tales Event Attributes

The following table lists available event attributes.

Attribute	Description
agent	Collector name from which the event came (such as zensyslog or zentrap).
component	Component of the associated device, if applicable. (Examples: eth0, httpd.)
count	Number of times this event has been seen.
dedupid	Key used to correlate duplicate events. By default, this is: device, component, eventClass, eventKey, severity.
device	ID of the associated device, if applicable.
DeviceClass	Device class from device context.
DeviceGroups	Device systems from device context, separated by  .
eventClass	Event class associated with this device. If not specified, may be added by the rule process. If this fails, then will be /Unknown.
eventClassKey	Key by which rules processing begins. Often equal to component.
eventGroup	Logical group of event source (such as syslog, ping, or nteventlog).
eventKey	Primary criteria for mapping events into event classes. Use if a component needs further de-duplication specification.
eventState	State of event. 0 = new, 1 = acknowledged, 2 = suppressed.
evid	Unique ID for the event.
facility	syslog facility, if this is a syslog event.
firstTime	UNIX timestamp when event is received.
ipAddress	IP Address of the associated device, if applicable.
lastTime	Last time this event was seen and its count incremented.
Location	Device location from device context.
manager	Fully qualified domain name of the collector from which this event came.
message	Full message text.
ntevid	nt event ID, if this is an nt eventlog event.
priority	syslog priority, if this is a syslog event.
prodState	prodState of the device context.
severity	One of 0 (Clear), 1 (Debug), 2 (Info), 3 (Warning), 4 (Error) or 5 (Critical).
stateChange	Time the MySQLrecord for this event was last modified.
summary	Text description of the event. Limited to 150 characters.
suppid	ID of the event that suppressed this event.
Systems	Device systems from device context, separated by  .

## Configuration Properties and Custom Properties

Configuration properties and custom properties also are available for devices, and use the same syntax as shown in the previous sections.

---

# Glossary

alert	Email or page sent as a result of an event.
Configuration property	Property defined on a device or event class. Configuration properties control a large part of how monitoring is performed. Configuration relies on inheritance.
data point	Data returned from a data source. In many cases, there is only one data point for a data source (such as in SNMP); but there may also be many data points for a data source (such as when a command results in the output of several variables).
data source	Method used by the system to collect monitoring information. Example data sources include SNMP OIDs, SSH commands, and perfmon paths.
device	Primary monitoring object. Generally, a device is the combination of hardware and an operating system.
device class	Special type of organizer used to manage how the system models and monitors devices (through configuration properties and monitoring templates).
device component	Object contained by a device. Components include interfaces, OS processes, file systems, CPUs, and hard drives.
discovery	Process by which the system gathers detailed information about devices in the infrastructure. Results of discovery are used to populate the model.
event	Manifestation of important occurrence within the system. Events are generated internally (such as when a threshold is exceeded) or externally (such as through a syslog message or SNMP trap).
event class	Categorization system used to organize event rules.
event rules	Controls how events are manipulated as they enter the system (for example, changing the severity of an event). Configuration properties configure event rules.
graph	Displays one or more data points, thresholds, or both.
managed resource	Servers, networks, and other devices in the IT environment.
model	Representation of the IT infrastructure. The model tells the system "what is out there" and how to monitor it.
monitoring template	Description of what to monitor on a device or device component. Monitoring templates comprise four main elements: data sources, data points, thresholds, and graphs.
organizer	Hierarchical system used to describe locations and groups. Also includes special organizers, which are classes that control system configuration.
resource component	Interfaces, services and processes, and installed software in the IT environment.
threshold	Defines a value beyond which a data point should not go. When a threshold is reached, the system generates an event. Typically, threshold events use the /Perf event class.