Zenoss™
Open Enterprise Management

Zenoss Administration

Version 2.3

Zenoss, Inc.
www.zenoss.com

Zenoss Administration for Version 2.3

1. About Zenoss 1
1. Zenoss Overview 1
1.1. Configuration Model 2
1.2. Availability Monitoring 2
1.3. Event Management 2
1.4. Performance Monitoring 3
2. Detailed Architecture 4
1. Zenoss Detailed Architecture 4
2. User Layer 4
3. Data Layer 5
4. Collection and Control Services Layer 5
4.1. Automated Modeling Daemons 5
4.2. Availability Modeling Daemons 5
4.3. Event Collection Daemons 6
4.4. Performance Monitoring Daemons 6
4.5. Automated Response Daemons 6
3. Zenoss Interface and Navigation 7
1. About the Zenoss Dashboard 7
1.1. Navigation Menu 8
1.1.1. Hiding the Menu 9
1.1.2. Pinning the Menu 9
1.2. Breadcrumbs 9
1.3. Device/IP Search 9
1.4. User Information Area 10
1.5. Update Details 10
1.6. Portlets 10
1.6.1. Customizing Portlets 12
2. Customizing the Zenoss Dashboard 12
2.1. Selecting Portlets 13
2.2. Arranging Portlets 13
2.3. Changing the Dashboard Column Layout 14
3. Zenoss Network Map 14
3.1. Viewing Device and Network Details 15
3.2. Loading Link Data 15
3.3. Filtering By Device Type 15
3.4. Adjusting Viewable Hops 16
3.5. Adjusting the Network Map 16
3.6. Viewing Device or Network Details 16
4. Zenoss Menus 16
4.1. Page Menus 16
4.2. Table Menus 16
4. Managing Zenoss Users 18
1. About Zenoss User Accounts 18
2. Creating User Accounts 18
3. Editing User Accounts 19
3.1. Associating Objects in Zenoss with Specific Users 20
4. User Groups 22
5. Roles 23
5. Device Access Control Lists 24
1. About Device Access Control Lists in Zenoss 24
2. Key Elements 24
2.1. Permissions and Roles 24
2.2. Administered Objects 24
2.3. Users and Groups 24
2.4. Assigning Administered Object Access 24
2.5. Portlet Access Control 24
3. Setup and Configuration Examples 25
3.1. Restricted User with ZenUser Role 25

3.2. Restricted User with ZenManager Role 25
3.3. Adding Device Organizers 25
3.4. Restricted User Organizer Management 25
3.5. Viewing Events 25
4. Detailed Restricted Screen Functionality 26
4.1. Dashboard 26
4.2. Device List 26
4.3. Device Organizers 26
4.4. Reporting 26
6. Email and Pager Settings 27
1. About Email and Pager Settings 27
2. Setting SMTP and SNPP Information 27
7. Organizers and Path Navigation in Zenoss 29
1. About Organizers and Path Navigation 29
2. Classes 29
2.1. Setting zProperties at the Class Level 30
2.2. Defining and Applying Templates at the Class Level 31
2.3. Creating New Classes 32
3. Systems 32
3.1. Adding, Moving and Nesting Systems 32
3.1.1. Moving the Sub-System 33
4. Groups 34
4.1. Adding Groups 34
4.1.1. Moving Groups 34
5. Locations 35
5.1. Google Maps Geographic View of Network Health 35
5.1.1. Overview 35
5.1.2. API Key 35
5.1.3. Setting an Address for a Location 36
5.1.4. Clearing the Google Maps Cache 36
5.1.5. Network Links 36
5.1.6. Google Maps Example 37
5.2. Adding, Moving, and Nesting Locations 37
5.2.1. Moving Sub-locations 37
6. Inheritance 38
7. Multiple Mix-In Inheritance and Performance (RRD) Template Binding 38
7.1. Template Binding Example 1 39
7.2. Template Binding Example 2 39
8. zProperties 40
1. About zProperties 40
2. Event zProperties 40
3. Device zProperties 40
4. Service zProperties 43
5. Process zProperties 43
6. Network zProperties 43
7. Manufacturer zProperties 44
9. Device Inventory and Configuration 45
1. What is Zenoss Inventory and Configuration? 45
2. How Does Zenoss Model Devices? 45
3. The ZenModeler Daemon 45
4. Add a Device 45
5. Add a Device with Context 48
6. Discover Devices 48
6.1. Classifying Discovered Devices 50
6.2. Adding Information to a Device Record 50
7. Device List 50
7.1. Managing Multiple Devices from the Device List 51
8. Individual Device Tabs 51

8.1. Status Tab 51
8.2. OS (Operating Systems) Tab 53
8.2.1. File System Monitoring 54
8.3. Hardware Tab 54
8.4. Software Tab 55
8.5. Events Tab 56
8.6. Performance (Perf) Tab 56
8.7. Edit Tab 57
8.8. Device Custom Properties 58
8.9. Device zProperties 59
8.10. Device Templates 59
8.11. Device Administration 60
8.12. Device Collector Plugins 60
8.13. Device Modifications 61
9. Searching for Devices 61
10. Editing Device Configuration 62
11. Managing Devices 63
11.1. Remodeling a Device 64
11.2. Changing the Device Class of a Device 64
11.3. Resetting Device Manage IP 64
11.4. Renaming a Device 65
11.5. Locking Device Configuration 65
11.6. Resetting the Device Community 66
11.7. Pushing Configuration Changes Back to the Zenoss System 66
11.8. Clearing Heartbeats 66
11.9. Deleting Devices From the System 67
12. Modeling Devices Using SNMP 67
12.1. Testing to See if a Device is Running SNMP 67
12.2. Modeling Remote Windows Devices Using SNMP 67
12.3. Modeling Remote Linux Devices Using SNMP 67
12.4. Modeling Cisco Devices Using SNMP 68
13. Modeling Using SSH/COMMAND 68
13.1. Using Device Class to Monitor Devices Using SSH 68
14. Modeling Devices Using PortScan 68
14.1. Using the /Server/Scan Device Class to Monitor with Portscan 69
15. Modeling Plugins 69
15.1. Viewing Modeling Plugins for a Device 69
16. Debugging the Modeling Process 69
17. Adding Custom Links to a Device Status Page 70
18. Dumping and Loading Devices Using XML Lists 70
10. Monitoring VMware Devices 71
1. Monitoring VMware Servers with Zenoss 71
1.1. ZenVMware ZenPack 71
1.2. Setting Up VMware Monitoring 71
1.3. Viewing VMware Devices 73
1.4. Viewing Guest Virtual Machines 73
1.5. VMware Events 75
1.6. Migration Events 78
1.7. Custom Data Sources 78
11. Event Monitoring 80
1. About Event Monitoring In Zenoss 80
2. Understanding Zenoss Events 80
2.1. Life Cycle of a Zenoss Event 80
2.2. Zenoss Event Life Cycle 81
2.3. Suppressing Duplicate Events 81
2.4. Begin-End Correlation 82
3. Zenoss Event Console 82
3.1. Sorting and Filtering Events 83

3.2. Viewing Event Details 83
3.3. Selecting Events 84
3.4. Managing Events 84
4. Creating Test Events 85
4.1. Sending Events from the Command Line 85
5. Event Classes 86
5.1. Creating an Event Class 86
6. Event Manager Settings 86
6.1. Accessing Event Manager Settings 87
6.2. Changing Event Database Connection Information 87
6.3. Changing Event Manager Cache Settings 87
6.4. Changing Event Manager Maintenance Settings 88
7. Acknowledging Events 88
8. Moving Events Manually To History 88
9. Clearing Event History 89
10. Event Class Mapping 89
11. Applying Event and Device Context Using Event zProperties 92
12. Mapping Events Through the Zenoss Interface 93
13. Using Mappings to Correlate Events 94
14. Event Commands 94
14.1. Commands Tab 94
14.2. Example: Create an Event Command 94
14.3. Example: Test the Event Command 95
15. SNMP TRAPs and Event Transforms 95
15.1. Mapping SNMP TRAPs 95
15.2. Example: Sending Test Traps 97
15.3. Transforming Events with Event Mappings 98
15.4. Event Transforms Based on Event Class 98
16. Creating Custom Event Views 98
17. Capturing Email Messages as Zenoss Events 100
17.1. ZenMail 100
17.2. ZenPop 100
17.3. Translating Message Elements to the Event 100
12. Availability Monitoring 102
1. Monitoring Topology with ZenPing 102
1.1. Controlling the Ping Cycle Time 102
1.2. Using the Predefined /Ping Device Class 102
2. Monitoring TCP Services 102
2.1. Zenstatus 102
2.2. Adding a Service To Monitor 103
2.3. Monitoring Status Service Status Information 103
2.4. Editing Service Information 104
2.5. Configuring Service zProperties 105
2.6. Using the Predefined /Server/Scan Device Class 106
2.7. Monitoring a Service Using a Service Class 106
3. Monitoring Processes 108
3.1. Adding Processes to Monitor 109
3.2. Configuring Process zProperties 111
13. Performance Monitoring 112
1. Performance Monitoring 112
2. Performance Templates 112
2.1. Templates Page 112
3. Template Binding 113
4. Data Sources 114
4.1. Adding a Data Source 114
5. Data Points 115
6. Thresholds 117
7. Performance Graphs 119

7.1. Graph Points 120
7.1.1. Re-sequencing Graph Points 120
7.1.2. DataPoint Graph Points 120
7.1.3. Threshold Graph Points 121
7.1.4. Custom Graph Points 122
7.2. Custom Graph Definition 122
7.3. Graph Commands 122
8. Changing Graph Display Order 122
14. Monitoring Devices Remotely via SSH 123
1. Monitoring Devices Remotely via SSH 123
1.1. Installing Zenoss Plugins on the Remote Machine 123
1.1.1. Zenoss Plugin Installation Technique: RPM 123
1.1.2. Zenoss Plugin Installation Technique: setuptools 123
1.1.3. Testing the Plugin Installation 123
1.1.4. Troubleshooting Plugin Installation 124
1.1.5. Changing Zenoss to Monitor Devices Remotely Using SSH 124
1.1.6. Using the Predefined /Server/Cmd Device Class 126
15. Monitoring Using ZenCommand 127
1. About ZenCommands 127
2. Example: Writing a ZenCommand (check_http example) 127
3. Example: Collect Data from A ZenCommand 128
4. Plugin Format for ZenCommands 129
5. Testing ZenCommands 130
16. Monitoring Windows Devices 131
1. Device Preparation for Windows Devices 131
2. Setting Windows zProperties 131
3. Testing WMI on a Windows Server 132
4. Optional Windows Configuration 132
5. Modeling Services on Windows Devices 132
6. Collecting Windows Eventlog Events 132
7. Monitoring Windows Performance with SNMP Informant 133
8. Running Commands on Windows Servers Using Winexe 133
17. Windows Performance Monitoring (Zenwinperf) 135
1. About Windows Performance Monitoring 135
2. Zenwinperf Daemon 135
2.1. ZenwinPerf zProperties 135
2.2. How to Create a WinPerf Data Source 135
2.3. Windows Performance Counters 136
3. /Device/Server/Windows/WMI Device Class 136
3.1. Device Templates 136
3.1.1. Device Template 136
3.1.2. File System Template 137
3.1.3. ethernetCsmacd Template 137
18. SNMP Monitoring 138
1. SNMP from the Command Line 138
19. Alerting Rules 139
1. Creating and Using Alerts 139
1.1. Setting SMTP Settings For Alerts 139
2. Creating a New Alerting Rule 140
2.1. Define and Enable This Alert 143
2.2. 1.1.1 Create the Content of the Alert Message 144
2.3. Create a Schedule for Sending the Alert 144
3. Escalation of Messaging in Zenoss 147
3.1. Creating an Alerting Hierarchy 147
4. Adding Delay and Schedules to Alerting Rules 147
20. UI Commands 149
1. About UI Commands 149
2. Defining UI Commands 149

2.1. UI Command Example: Echo Command 150
3. Running Commands from the Zenoss Web UI 151
4. Auto-Running Commands Based on Events 151
21. Production States and Maintenance Windows 154
1. About Production States and Maintenance Windows 154
2. Production States 154
2.1. Defining Production States for Devices 154
3. Maintenance Windows 154
3.1. Creating and Using Maintenance Windows 155
22. Reporting 156
1. About Zenoss Reporting 156
2. Organizing Reports 156
3. Navigating and Sorting Report Results 156
4. Exporting Reports 157
4.1. Advanced: Add An Export Button to a Report 157
5. Reports Included With Zenoss 157
5.1. Device Reports 157
5.2. Event Reports 158
5.3. Performance Reports 158
5.4. User Reports 159
6. Graph Reports 159
6.1. Creating a Graph Report 160
6.2. Adding Graphs 160
6.3. Customizing Graph Text 161
6.4. Organizing Graphs 162
7. MultiGraph Reports 162
7.1. Creating A MultiGraph Report 163
7.2. Collections 164
7.3. Graph Definitions 165
7.4. Graph Groups 166
7.5. Graph Order 167
8. Creating Custom Reports 167
8.1. Creating Custom Reports Using the ZMI 167
8.2. Create A Custom Device Report Example 167
9. Using Reports to Help Troubleshoot Zenoss Daemons 169
10. Advanced Zenoss Reports 169
23. General Zenoss Administration 170
1. Working with Zenoss from the Command Line 170
2. Minimal Zenoss - ZEO and Zope 170
3. Checking the Version of Zenoss 170
4. Checking for Zenoss Updates 171
5. Starting and Stopping the Zenoss Daemons 171
6. Zenoss Daemon Commands and Options 171
6.1. Configuring Zenoss Daemons 172
6.2. General Options for Zenoss Daemons 172
6.3. zenhub Options 172
6.4. zenmodeler Options 172
6.5. zenperfsnmp Options 173
6.6. zenperfxmlrpc Options 173
6.7. zenprocess Options 174
6.8. zenping Options 174
6.9. zensyslog Options 174
6.10. zenstatus Options 174
6.11. zenactions Options 175
6.12. zentrap Options 175
6.13. zencommand Options 175
7. Troubleshooting Zenoss Daemons 175
8. Automatic Startup in Linux Environments 176

9. Using Zenoss with a Remote MySQL Instance 176
10. Loading and Registering MIBs with Zenoss 176
10.1. Resolving MIB Dependencies 176
24. Backup, Recovery and Maintenance 177
1. Backup and Restore 177
1.1. Backup Details 177
1.2. Backups Tab 178
1.3. Remote Backups 178
1.4. Restore Details 178
1.5. Periodic Backups 179
1.5.1. Pack ZEO Database 179
1.5.2. Log Rotate Script 179
1.5.3. Backing up the MySQL Event Backend 179
25. ZenPacks 180
1. Introduction to ZenPacks 180
1.1. Installing ZenPacks 180
1.1.1. Installing via the Command Line 180
1.1.2. Installing via the User Interface 181
1.1.3. Installing All Core ZenPacks via RPM 181
1.2. Creating a ZenPack 181
1.2.1. Packaging and Distributing Your ZenPack 182
1.3. Removing a ZenPack 182
2. Zenoss Core ZenPacks 182
2.1. ZenJMX ZenPack 182
2.1.1. About ZenJMX 182
2.1.2. JMX Background 182
2.1.3. ZenJMX Capabilities 183
2.1.4. Single Value Attribute Calls 183
2.1.5. Complex-Value Attribute Calls 184
2.1.6. Operation Calls 184
2.1.7. ZenJMX Behavior 187
2.1.8. Running the ZenJMX Daemon 187
2.1.9. Defining Custom JMX Data Sources 187
2.1.10. Attribute Path Input Value 187
2.1.11. Enabling Remote JMX Access 188
2.1.12. Interrogating an JMX Agent via JConsole 190
2.1.13. Installing ZenJMX 193
2.2. ApacheMonitor ZenPack 193
2.3. DellMonitor ZenPack 194
2.4. HPMonitor ZenPack 195
2.5. MySqlMonitor ZenPack 195
2.6. NtpMonitor ZenPack 196
2.6.1. Components 196
2.6.2. DataSource Class Options 196
2.6.3. Example 196
2.7. LDAPMonitor ZenPack 196
2.7.1. Components 197
2.7.2. DataSource Class Options 197
2.7.3. Additional Configuration 197
2.7.4. Example 197
2.8. RPCMonitor ZenPack 198
2.8.1. Components 198
2.8.2. DataSource Class Options 198
2.8.3. Additional Configuration 198
2.8.4. Example 198
2.9. IRCMonitor ZenPack 199
2.9.1. Components 199
2.9.2. DataSource Class Options 199

2.9.3. Example 199
2.10. JabberMonitor ZenPack 199
2.10.1. Components 199
2.10.2. DataSource Class Options 200
2.10.3. Example 200
2.11. DigMonitor ZenPack 200
2.11.1. Components 200
2.11.2. DataSource Class Options 201
2.11.3. Example 201
2.12. FTPMonitor ZenPack 201
2.12.1. Components 201
2.12.2. Datasource Class Options 202
2.12.3. Example 202
2.13. NNTPMonitor ZenPack 202
2.13.1. Components 202
2.13.2. DataSource Class Options 203
2.13.3. Example 203
26. Zenoss Global Dashboard (ZenGlobe) 204
1. About Zenoss Global Dashboard (ZenGlobe) 204
2. Additional Requirements 204
3. Installation 204
3.1. Setting Up Users on Remote Zenoss Instances 205
4. Setup and Configuration 205
5. Using the Zenoss Global Dashboard 205
6. Viewing a Zenoss Instance from the Zenoss Global Dashboard 205
7. Logging out of the Zenoss Global Dashboard 206
27. Zenoss High Availability 207
1. Overview 207
2. Conventions 207
3. Prerequisites 208
4. Installation of Packages 208
5. Stopping Services 208
6. Setting up DRBD for File System Replication 208
7. Relocating Shared Data to the Replicated Disk 209
8. Setting up Heartbeat for Resource Migration 210
9. Setting up Heartbeat Resources 210
10. Troubleshooting 211
11. References 212
28. Distributed Collector 213
1. What is Distributed Collector? 213
1.1. Restrictions and Requirements 213
1.2. Navigating Existing Collectors and Hubs 213
2. Typical Usage Scenarios for Distributed Monitoring 214
2.1. Scalability on the Local Device 214
2.2. Scalability Over Multiple Remote Devices 214
2.3. Collectors and or Hubs Set Up through a Firewall 214
2.4. Collection Over Multiple Overlapping IP Spaces 214
3. Deploying Hubs 214
4. Deploying Collectors 216
4.1. Deleting Collectors 217
4.2. Updating a Hub or Collector 217
5. Adding Devices to Collectors 217
5.1. Moving Devices Between Collectors 218
5.1.1. Moving VMware Devices Between Collectors 219
5.2. Installation Notes 219
5.3. Debugging 219
5.4. Firewall Notes 220
5.5. Platform Notes 220

29. Monitoring Virtual Host Machines 221
1. About Monitoring Virtual Host Monitoring 221
2. Using Zenoss Virtual Host Monitor 221
30. Predictive Thresholding (ZenHoltWinters) 222
1. About Predictive Thresholding 222
2. Using Predictive Thresholding 222
3. Customizing the Thresholds 222
31. Authentication Enterprise ZenPacks 223
1. Active Directory Authentication 223
1.1. About Active Directory Authentication 223
1.2. Active Directory Monitored Metrics 223
2. LDAP Authentication in Zenoss 224
2.1. About LDAP Authentication 224
2.2. Setting Up LDAP Authentication 224
32. Transaction Monitoring ZenPacks 225
1. Synthetic Transactions (ZenWebTx) 225
1.1. About Synthetic Transactions (ZenwebTx) 225
1.2. Overview 225
1.3. Creating a WebTx Data Source 226
1.4. Creating Twill Commands 227
1.5. TestGen4Web 227
1.6. Creating Twill Commands Manually 228
1.7. Event Generation 228
1.8. Data Points 228
2. Mail Transactions Monitoring (ZenMailTx) 229
2.1. About Mail Transaction Monitoring 229
2.2. Creating a ZenMailTX Data Source 229
2.2.1. Testing Data Source Settings 229
2.3. zenmailtx daemon 229
3. SQL Transaction Monitoring (ZenSQLTx) 230
3.1. About SQL Transaction Monitoring 230
3.2. Creating a SQL Data Source 230
33. Application Monitoring Enterprise ZenPacks 231
1. MS SQL Monitoring (MSSQLServer) 231
1.1. About MS SQL Transaction Monitoring 231
1.2. Installed Objects 231
1.2.1. Device Class 231
1.2.2. Event Class 231
1.2.3. WinServices 231
1.2.4. IpServices 231
1.2.5. Performance Templates 231
1.2.6. Datasources 231
1.2.7. Thresholds 232
1.3. Using the MSSQLServer ZenPack 232
2. MS Exchange Monitoring (MSExchange) 232
2.1. About MS Exchange Monitoring 232
2.2. Installed Objects 232
2.2.1. Device Classes 232
2.2.2. Event Classes 232
2.2.3. Event Class Keys 232
2.2.4. Alerting Rules 233
2.2.5. WinServices 233
2.2.6. IP Services 233
2.2.7. Performance Templates 233
2.2.8. Datasources 233
2.2.9. Thresholds 234
2.3. Performance Counters and Thresholds 234
2.4. Reports, Views & Dashboards 234

2.5. Using the the MSExchange ZenPack 234
3. IIS Monitoring 234
3.1. About IIS Monitoring 234
3.2. Setting up IIS Monitoring 235
4. JBoss Application Server Monitoring 235
4.1. About JBoss Application Server Monitoring 235
4.2. Overview 235
4.3. Collected Metrics for JBoss 235
4.4. Setting Up the JBoss ZenPack 236
5. WebLogic Application Server Monitoring 236
5.1. About WebLogic Application Server Monitoring 236
5.2. Weblogic Collected Metrics 236
5.2.1. Overall Application Server Vitals 236
5.2.2. Entity EJB, Message Driven Bean (MDB), and Session EJB Subsystem Metrics ... 237
5.2.3. Data Pool (JDBC) metrics 237
5.2.4. Queue (JMS) Metrics 237
6. Tomcat Application Server Monitoring 237
6.1. About Tomcat Application Server Monitoring 237
6.2. Collected Metrics for Tomcat 238
6.3. Tomcat Monitoring Setup 238
34. Integration Enterprise ZenPacks 239
1. Remedy Integration in Zenoss 239
1.1. About Remedy Integration 239
1.2. Setting Up Remedy Integration 239
2. RANCID Integration 239
2.1. About RANCID Integration 239
2.2. Setting Up RANCID Integration 240
A. Net-SNMP and Zenoss 241
1. Net-SNMP Configuration Settings 241
1.1. Community Information 241
1.2. System Contact Information 241
1.3. Extra Information 241
B. Event Database Dictionary 242
1. Event Database Dictionary 242
C. TALES Expressions 243
1. About TALES Expressions 243
2. TALES Device Attributes 243
3. TALES Event Attributes 245
D. Device Preparation 246
1. Net-SNMP 246
2. SNMP V3 Support 246
3. Forwarding Syslog Messages from UNIX/Linux Devices 247
4. Forwarding Syslog Messages from a Cisco IOS Router 247
4.1. Other Cisco Syslog Configurations 247
5. Forwarding Syslog Messages from a Cisco CatOS Switch 248
6. Forwarding Syslog Messages using Syslog-ng 248
E. ZenWebTx Twill Commands Reference 249
1. About Twill Commands 249
2. Browsing 249
3. Assertions 249
4. Display 250
5. Forms 250
6. Cookies 250
7. Debugging 251
8. Other Commands 251
9. Details on Form Handling 251
10. ZenWebTx Extensions to Twill 252
10.1. twilltiming 253

10.2. twillextract 253
10.3. twillxpathextract 254
10.4. ignorescripts 254
F. Basic Troubleshooting in Zenoss 255
1. How To Troubleshoot Zenoss Daemons 255
1.1. Identify the Problem 255
1.1.1. Check the logs 255
1.1.2. Get more information 255
1.1.3. Find out what the program is **_really_** doing 255
1.2. Narrow the Problem 256
1.3. Look for Conflicts 256
1.4. Reproduce the Problem 256
1.5. Getting Help Solving the Problem 256
1.5.1. Search the Zenoss Forums 256
1.5.2. Report the Problem to Zenoss 256

Chapter 1. About Zenoss

1. Zenoss Overview

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

The power of Zenoss starts with its in-depth Inventory and Configuration Management Database (CMDB). Zenoss 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 key systems, down to the level of *resource components* (interfaces, services, and processes, and installed software).

With the model built, you then use Zenoss' integrated availability and monitoring features to monitor and report on all aspects of your IT infrastructure. Zenoss also provides events and fault management features that tie into the CMDB. These features help drive operational efficiency and productivity by automating many of the notification, alerts, escalation, and remediation tasks you perform each day.

Zenoss Architecture

At a high level, Zenoss comprises these major sub-systems:

- Configuration model
- Availability monitoring
- Event management
- Performance monitoring
- Discovery and collection

Figure 1.1. High-Level View of Zenoss

1.1. Configuration Model

At the heart of Zenoss is the configuration model (also known as the standard model), which drives all monitoring elements of Zenoss. The model is a detailed description of each device that Zenoss manages, and its relationship to other devices, business units, and any groups you define.

The model is populated through:

- Discovery -- Through the *discovery* process, Zenoss uses one of the available transports to discover services and interfaces on a device.
- Manual addition -- Through the Zenoss Web interface, or its external APIs, you can add and manipulate devices.

Discovery locking allows Zenoss to tightly integrate discovered information with information you add manually.

Zenoss brings together a wide range of monitoring and management information. This information is available through a standard Web browser. In fact, all aspects of the system are accessed though the Web -- there is no need to edit configuration files.

1.2. Availability Monitoring

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

1.3. Event Management

Zenoss event management consolidates status information from all parts of the Zenoss system, as well as any monitored external systems. When a Zenoss monitoring daemon detects a failure or threshold breach, the system

generates events. Zenoss also incorporates event imports from other parts of the IT infrastructure, including syslog and SNMP traps. As it receives events, Zenoss runs them through a set of rules that augment the data. It then integrates all of this information into the configuration model.

1.4. Performance Monitoring

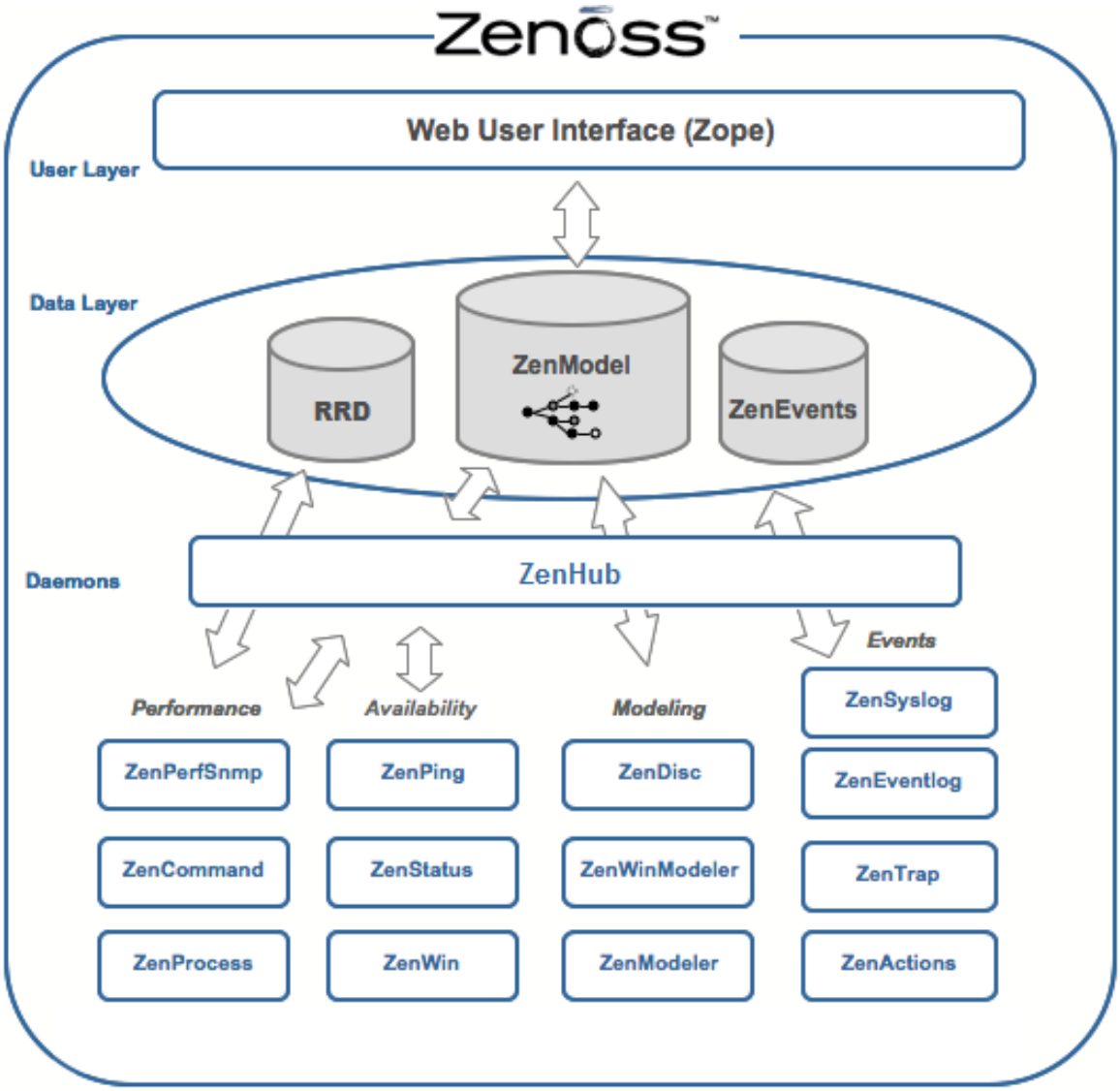
Zenoss performance monitoring tracks important IT resource information and tracks these changes over time. It is critical to know how much disk space is available, what the CPU load is, or how long a Web page takes to download. Zenoss can collect information by using SNMP, custom scripts (ZenCommands)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

Chapter 2. Detailed Architecture

1. Zenoss Detailed Architecture

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

Figure 2.1. Zenoss Detailed Architecture



The diagram illustrates three distinct layers of the Zenoss system:

- User layer
- Data layer
- Collection and control layer

2. User Layer

The User layer is manifested as a Web portal. This layer consists of the Zenoss Graphical User Interface (GUI), which allows you to access and manage key Zenoss components and features. From here, you can:

- Watch the status of your enterprise, using the Zenoss 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 GUI.

3. Data Layer

Zenoss system information is stored in the Data layer. This level represents the "heart" of Zenoss, consisting of these components:

- ZenRD - Gathers and stores performance data (through RRDtool).
- ZenModel - Serves as the configuration management interface, holding device data in the Zeo back-end object database).
- ZenEvents - Stores MySQL events.
- ZenHub - Brokers information between the data layer and the collection daemons.

4. Collection and Control Services Layer

The services that collect and feed data to the Data layer are provided by the daemons associated with the Collection and Control Services Layer. These daemons are divided among these distinct areas:

- Automated Modeling
- Availability Monitoring
- Event Collection
- Performance Monitoring
- Automated Response

The daemons in each category are detailed in the following sections.

4.1. Automated Modeling Daemons

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

4.2. Availability Modeling Daemons

Daemon	Description
Zenping	Zenping is the ping status monitoring (ICMP) for Zenoss. Zenping does the high-performance asynchronous testing of the ICMP status.

Daemon	Description
Zenstatus	Zenstatus performs active TCP connection testing of remote daemons.
Zenprocess	Zenprocess enables process monitoring using SNMP host resources MIB.

4.3. Event Collection Daemons

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

4.4. Performance Monitoring Daemons

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

4.5. Automated Response Daemons

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

Chapter 3. Zenoss Interface and Navigation

1. About the Zenoss Dashboard

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

Figure 3.1. Zenoss Dashboard

The Dashboard can show:

- Important error-level device events
- Geographical high-level view
- "Troubled" devices

Key Dashboard and interface areas include:

- Navigation menu
- Breadcrumbs
- Device/IP search
- User information area
- Portlets

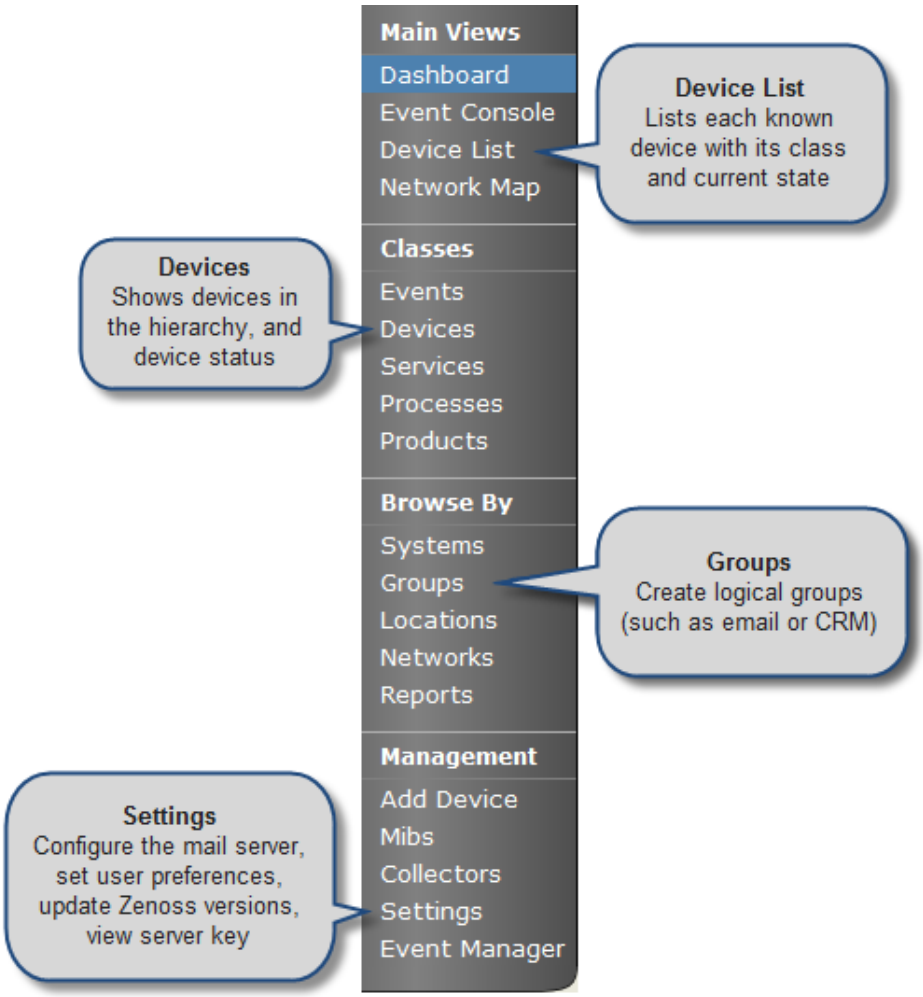
1.1. Navigation Menu

The Navigation menu lets you access most of the Zenoss' features. The menu is divided among several functional areas:

- Main Views, which includes these selections:
  - Dashboard - Returns you to the primary view.
  - Event Console - Shows a list of all current events in the event database.
  - Device List - Shows a list of all devices in Zenoss.
  - Network Map - Shows a graphical representation of the devices in your network.
- Classes, which includes these selections:
  - Events - Links to the event management area, where you can monitor event status, events, history, zProperties, and event transforms. You also can track changes made to events.
  - Devices - Lets you manage sub-devices and a summary of events by severity. Allows you to view events sorted by severity, followed by device name, history of device events, PerfConfig, zProperties for devices, and recent device changes.
  - Services - Allows you to show service classes, administer commands on a service basis, access zProperties, and track changes made to services monitoring.
  - Processes - Lets you create new process groupings and add processes to monitor.
  - Products - Shows a list of all manufacturers of devices in the Zenoss database.
- Browse By, which lets you see data based on any of the local groupings Zenoss enables you to create. Selections include:
  - Systems - Lets you see network status, categorized into the system groupings you create.
  - Groups - Provides access to the same data as when browsing by Systems, with the exception of performance data.
  - Locations - Allows you to see data related to devices grouped by physical locations.
  - Networks - Shows devices and sub-networks, based on IP address groupings.
  - Reports - Lets you view and define reports in Zenoss.
- Management, which includes:
  - Add Device - Add devices to Zenoss.
  - MIBs - Add and manage MIBs in Zenoss.
  - Collectors - Add collectors to Zenoss for better performance and scaling when the device count is too large for one collector.
  - Settings - Manage other Zenoss settings, such as dashboard production state, state conversions, and administrative roles.
  - Event Manager - Lets you view and manage the event cache.

The following figure illustrates key selections from the Navigation menu.

Figure 3.2. Navigation Menu



1.1.1. Hiding the Menu

Click the triangular indicator at the top of the Navigation menu to hide or display menu selections.

1.1.2. Pinning the Menu

Click the pin icon to "pin" the menu into place, keeping it visible in all views.

1.2. Breadcrumbs

The breadcrumbs area shows your current location in Zenoss. Use this trail to keep track of your location and navigate to previously selected pages in the interface hierarchy.

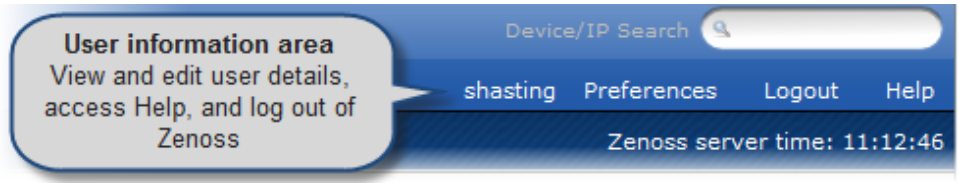
Figure 3.3. Breadcrumbs (Navigation)

1.3. Device/IP Search

Enter all or part of a device name in the Device/IP Search field to locate a device. Alternatively, you can enter an IP address to find the device.

1.4. User Information Area

Figure 3.4. User Information Area



The User information area offers information and selections:

- Login ID - The ID of the user currently logged in to Zenoss appears at the far left of this area.
- Preferences - Click to edit user settings, such as authentication information, roles, and groups. (You also can access user settings from the Navigation menu Settings selection.)

  Note

  From other Preferences tabs, you can manage administered objects, event views, and alerting rules.

- Logout - Click to log out of Zenoss.
- Help - Click to access Zenoss community product documentation, FAQs, and HowTos, at:

  http://www.zenoss.com/community/docs

1.5. Update Details

The date and time that Zenoss was last updated appears below the Preferences link. Every 60 seconds, Zenoss polls for new data and refreshes the data fields. If the poll fails, then Zenoss displays the message:

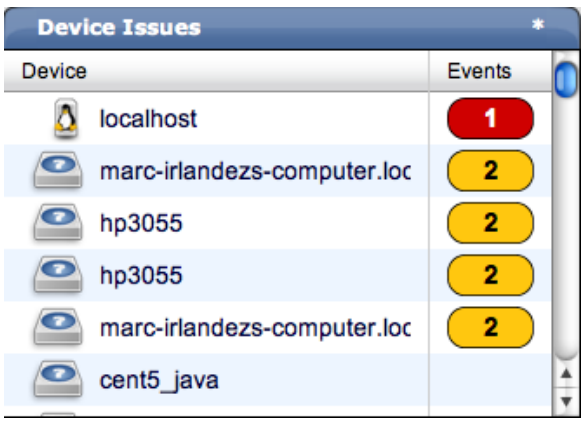
```
Lost Connection to Zenoss
```

1.6. Portlets

The main content of the Dashboard comprises portlets, which provide information about the system and your infrastructure. Portlets that 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 3.5. Device Issues Portlet

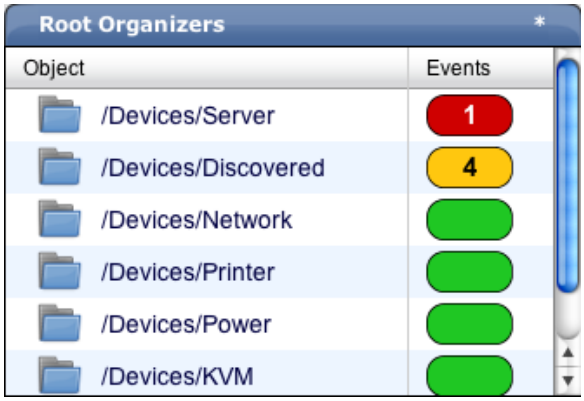


- Google Maps (device locations) - Shows configured locations and configured network connections.

Figure 3.6. Google Maps Portlet

- Zenoss Issues - Contains system self-monitoring information.
- Production States - Shows devices assigned to a particular production state.
- Top Level (Root) Organizers - Lists status for each grouping in your defined Zenoss hierarchy.

Figure 3.7. Top Level Organizers Portlet



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

1.6.1. Customizing Portlets

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

Click * (asterisk), 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.

For this portlet type..	...you can customize:
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. Customizing the Zenoss Dashboard

You can customize the Zenoss Dashboard by:

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

Figure 3.8. Customize Dashboard

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.

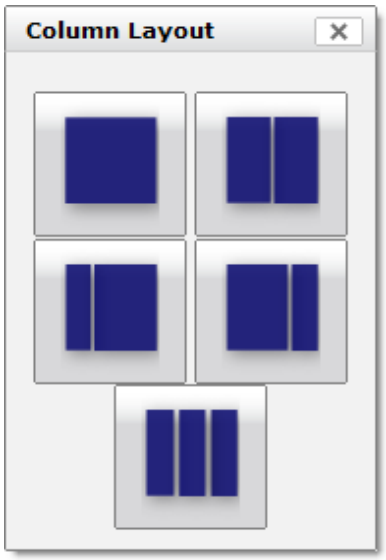
2.2. Arranging Portlets

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

2.3. 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 3.9. 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.

3. Zenoss 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 background color.

Figure 3.10. Zenoss Network Map

3.1. 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.

3.2. 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.

3.3. 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.

3.4. Adjusting Viewable Hops

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

3.5. 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.

3.6. 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.

4. Zenoss Menus

Zenoss offers two types of menus from which you make selections:

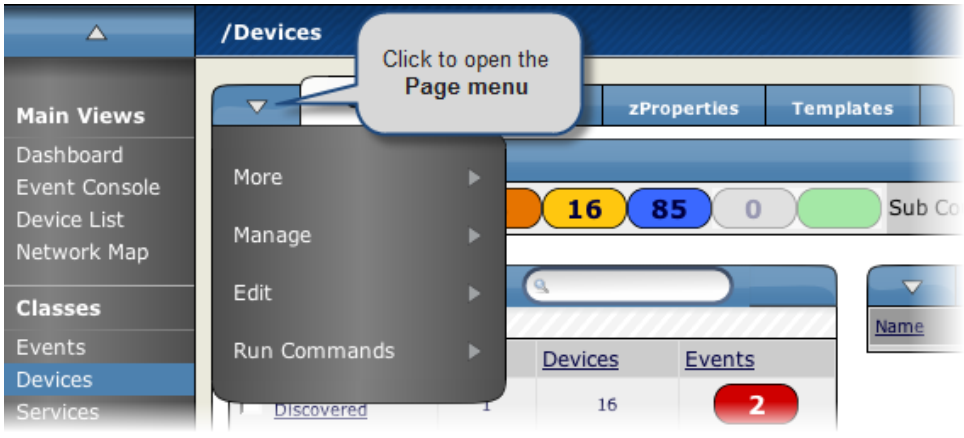
- Page menus
- Table menus

4.1. Page Menus

Page menus extend the tabs that appear at the top of the page. Generally, actions initiated through a page menu affect the object or objects that the page represents. This could be, for example, a device or any group of devices.

As shown in the following figure, the Page menu is expanded next to the Classes tab.

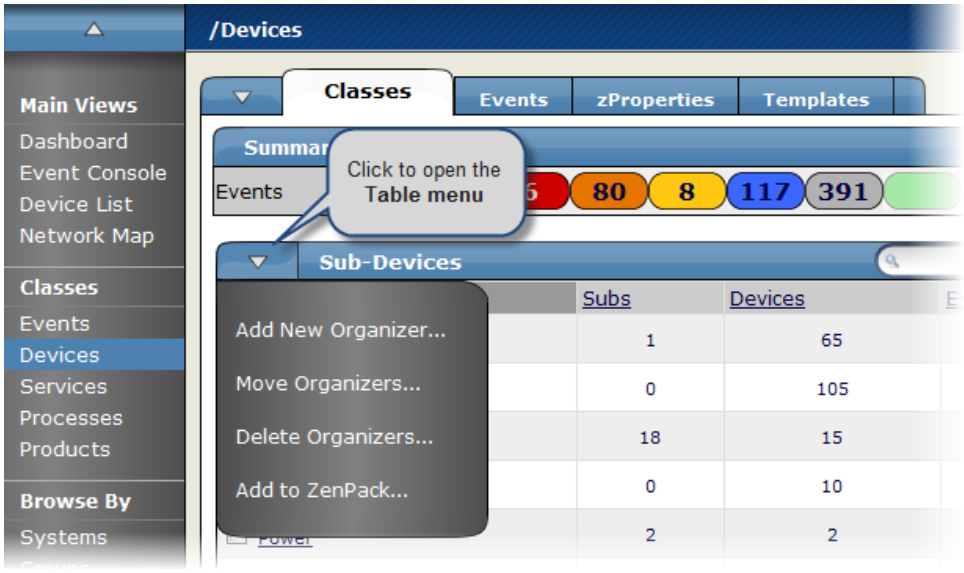
Figure 3.11. Page Menu



4.2. Table Menus

Table menus generally affect objects in a table. Access table menus by clicking the triangle next to a table title on a page. As shown in the following figure, the Sub-Devices table menu is expanded.

Figure 3.12. Table Menu



Chapter 4. Managing Zenoss Users

1. About Zenoss User Accounts

Each Zenoss 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.

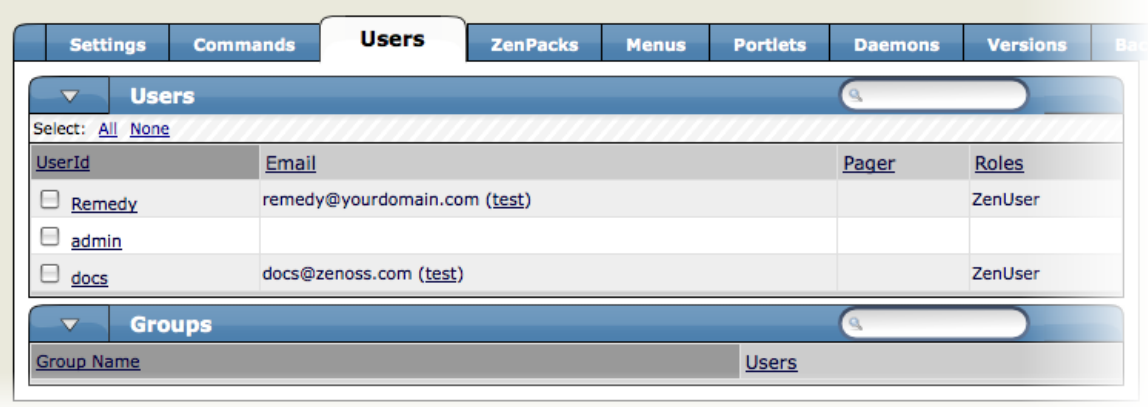
To create and manage user accounts, you must be logged in to the Zenoss admin account, or as a user with manage privileges.

2. Creating User Accounts

To create a user account:

1. From the Zenoss Dashboard Navigation menu, select Settings.
2. Click the Users tab. The Users administration page appears.

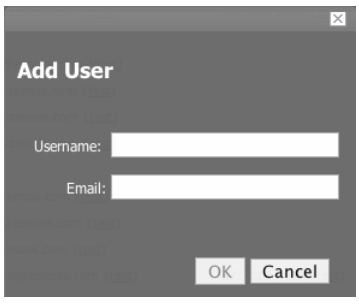
Figure 4.1. User Administration



3. From the Users table menu, select Add User.

   The Add User dialog appears.

Figure 4.2. Add User Dialog



4. In the Username field, enter a unique name for the account.
5. In the Email field, enter the user account email address.

   Note: 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. See the section titled Editing User Accounts in this chapter for more information about setting user preferences.

3. Editing User Accounts

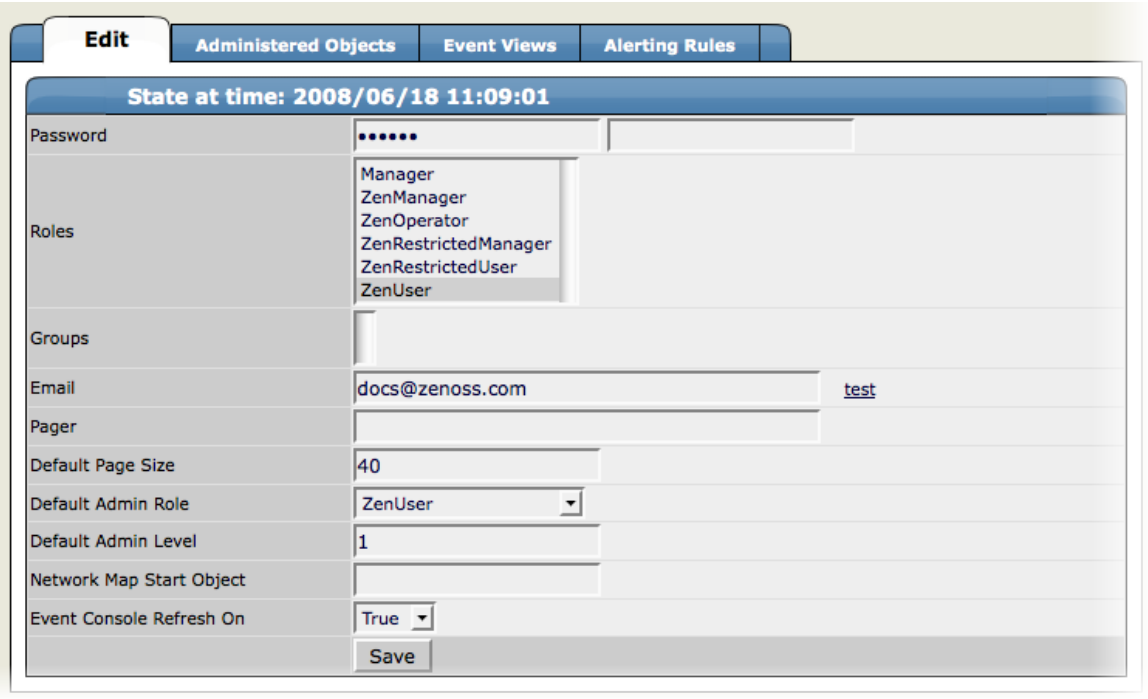
To access and edit user account information:

1. From the navigation menu, select Settings.
2. Click the Users Tab.

   The Users Administration page appears.
3. Click the name of the user account you want to edit.

   The individual User Administration page appears.

Figure 4.3. Edit User



4. Make changes to one or more settings:
   - Password - Assign and confirm a user password.
   - Roles - Assign one or more roles (user privileges) to the user. To edit or assign roles, you must be a Zenoss Admin or be assigned the Manager role.

     For more information about Zenoss user roles, and for a list of available roles and the privileges they provide, see the section titled Roles in this chapter.
   - 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 - 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.
- Event Console Refresh On - Select True or False.

3.1. Associating Objects in Zenoss with Specific Users

You can associate any object in the Zenoss system with a particular user, for monitoring or reporting purposes. Once associated with a user, you can then assign the user a specific role that applies to his privileges with respect to that object.

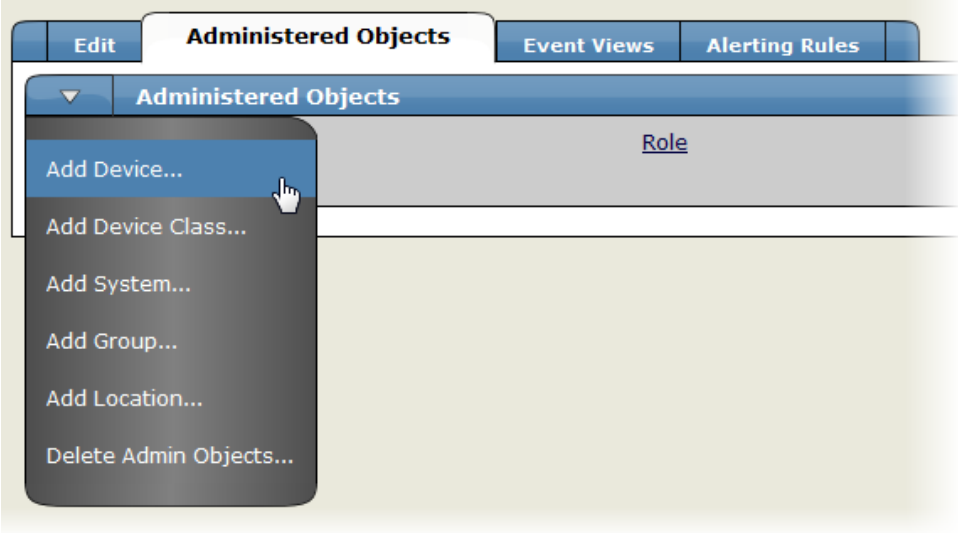
For more information about object-specific roles, see the section titled Roles in this chapter.

To create an object association:

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

   The list of administered objects appears.

Figure 4.4. Administered Objects - Add Object



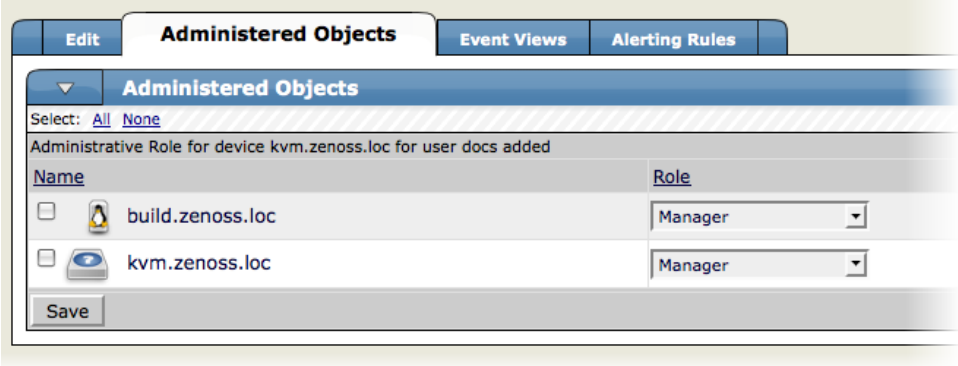
2. Select an object type from the Administered Objects table menu. You can add a Device, System, Group, or Location.

   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.

Figure 4.5. Administered Objects - Object Added



4. Optionally, change the role that is associated for this user on this object.

   ## Note

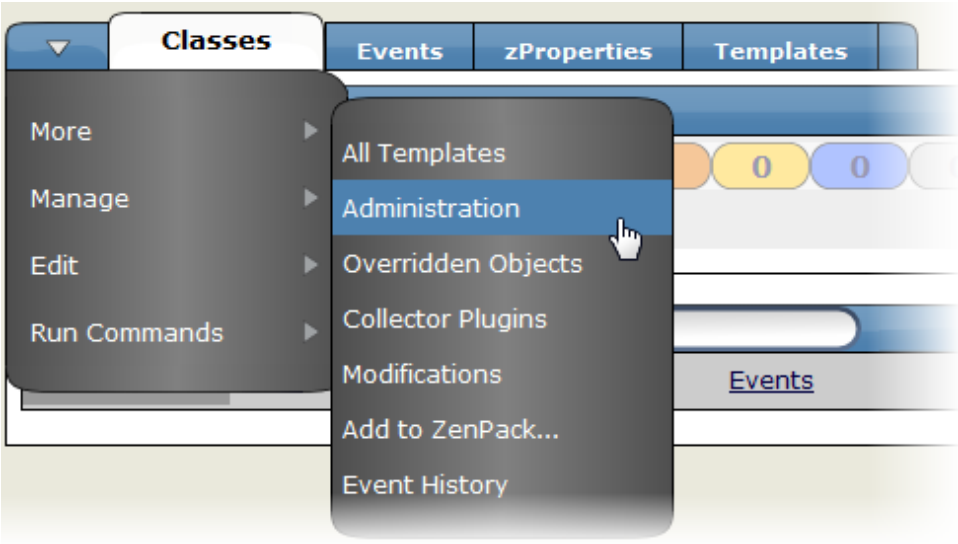
   The default role assigned to the user for an administered object is specified by the Default Admin Role field on the Edit tab.

5. Click Save to save changes.

You can also set associated an object with a user by adding an administrator to the object. To do this:

1. Navigate to the object you want to add the user's list of administered objects.
2. From the page menu, select More, and then select Administration.

Figure 4.6. Administered Objects - Add Administrator



3. In the Administrators area, select Add Administrator from the table menu.

   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.

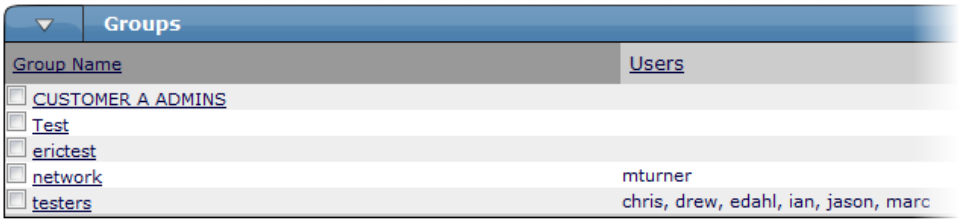
4. User Groups

Zenoss allows you to create user groups. By grouping users, you can aggregate rules and apply them across multiple user accounts.

Viewing User Groups

To view user groups, go to the Users tab of the Settings page. The groups area shows each user group and the users assigned to that group.

Figure 4.7. User Groups (Settings Page)



Creating User Groups

You can create Zenoss user groups to aggregate rules and apply them across multiple user accounts.

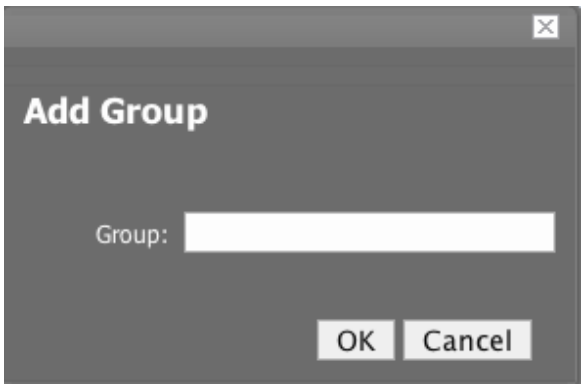
To create a user group:

1. From the Navigation menu, select Settings.
2. Click the Users tab.

   The Users tab appears.
3. From the Groups table menu, select Add New Group.

   The Add New Group dialog appears.

   ### Figure 4.8. Add New User Group Dialog

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 Edit tab for this group appears.

6. From the Users In Group table menu, select Add User.

   The Add User to Group dialog appears.

Figure 4.9. Add User to Group Dialog

7. From the User list of selections, select the 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.

5. Roles

A role is a group of permissions that you can assign to users or groups in Zenoss.

The following table lists available Zenoss roles.

Role	Definition
ZenUser	Provides global read-only access to Zenoss objects.
ZenManager	Provides global read-write access to Zenoss objects.
Manager	Provides global read-write access to Zenoss objects. Additionally provides read-write access to the underlying Zope object database.
ZenOperator	Installed by the ZenOperatorRole ZenPack. This ZenPack is available only to Zenoss Professional and 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 Zenoss, but also allow them to acknowledge events, move events to history, and add log messages to events.

Chapter 5. Device Access Control Lists

1. About Device Access Control Lists in Zenoss

Note

This feature is available only with Zenoss Enterprise.

The Device Access Control List (ACL) Enterprise ZenPack (ZenDeviceACL) adds fine-grained security controls to Zenoss. For example, this control can be used to give limited access to certain departments within a large organization or limit a customer of a service provider to only see 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.

2. Key Elements

Following are key elements of device ACLs.

2.1. Permissions and Roles

Actions within Zenoss are assigned permissions. For instance to access the device edit screen you must have the “Change Device” permission. Permissions are not assigned directly to a user since this would be difficult to manage. 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 tab, it is “global." When creating a restricted user you may not want to give that user any global role.

2.2. Administered Objects

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

2.3. Users and Groups

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

2.4. Assigning Administered Object Access

For each user or group there is a tab called “Administered Objects." The menu has an add item for each type of administered object. Adding an object will pull up a dialog box with live search on the given type of object. After an object has been added you can assign it a role. Roles can be different for each object so a user or group might have ZenUser on a particular device but 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.

2.5. Portlet Access Control

Within Zenoss Core, portlet access can be controlled. This is important for Device ACLs.

3. Setup and Configuration Examples

Refer to the following examples for setup and configuration steps.

3.1. Restricted User with ZenUser Role

1. As admin or any user account with Manager or ZenManager role, create a user named acltest. Set a password for the user.
2. From the user's Edit tab, make sure that no role is assigned.
3. Select the user's "Administered Objects" tab.
4. From, the menu, select the "Add Device…" item and add an existing device to that 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. Click on the "Device List".

   You should see only the device you assigned to acltest.
7. Navigate to the device and notice that the Edit tab is not available. This is because you are in read-only mode for this device.

3.2. Restricted User with ZenManager Role

Following the example above:

1. Change the acltest user's role to "ZenManager" on the device. (You must to do this as a user with ZenManager global rights.)
2. Go back to the acltest user "Administered Objects" tab and set the role on the device to ZenManager.
3. As acltest, navigate back to the device. You now have access to the Edit tab.

3.3. Adding Device Organizers

1. Go to the Groups root and create a group called "RestrictGroup."
2. Go to the acltest user's Administered Objects tab and add the group to the user.
3. Logged in as acltest, notice that the Navigation menu has the Groups item. Group 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

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.

3.5. Viewing Events

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

4. Detailed Restricted Screen Functionality

4.1. Dashboard

By default, the dashboard is configured with only three portlets:

- Object Watch List
- Device Issues
- Production State

These have content that will be restricted to objects for a given user.

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.

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.

4.4. Reporting

Reports are limited to device reports and performance reports.

Chapter 6. Email and Pager Settings

1. About Email and Pager Settings

You can send Zenoss 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 OS does not include a mail server, you must install one or specify a separate SMTP server in the Zenoss 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 Zenoss settings.

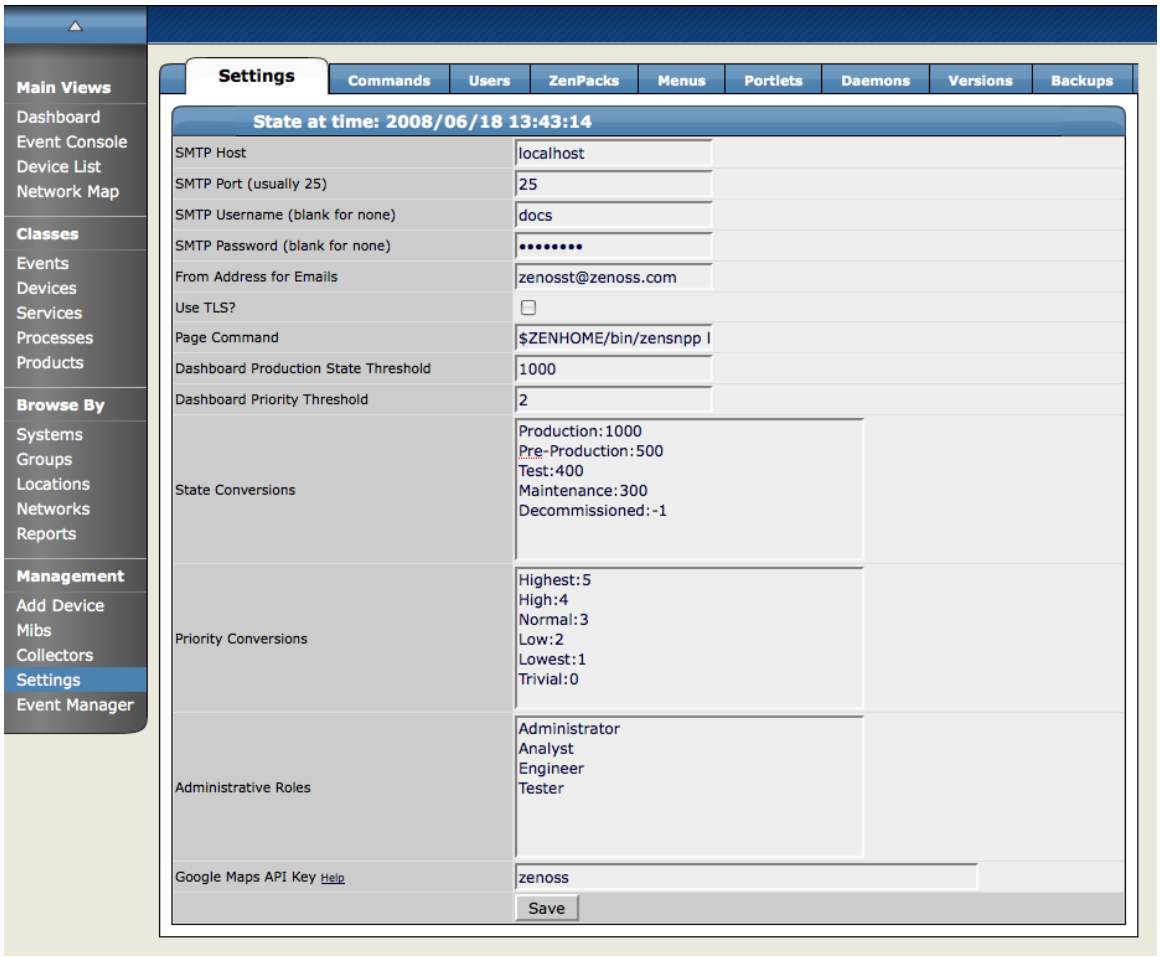
2. Setting SMTP and SNPP Information

To edit Zenoss's SMTP and SNPP settings:

1. While logged in to a user account with management privileges, from the Navigation menu, click Settings.

   The Settings Tab appears.

Figure 6.1. Zenoss Settings- Settings Tab



2. Change the following SMTP settings, as necessary:

Table 6.1.

Field	Description
SMTP Host	Set the SMTP Host value to your corporate email server.
SMTP Port	Usually port 25.
SMTP Username	Leave this field blank.
SMTP Password	Leave this field blank.
From Address for Emails	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.

3. Enter a Page Command as necessary if you are using Zenoss to send pages. The pageCommand variable enables Zenoss to execute the 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 Zenoss Page Command is:

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

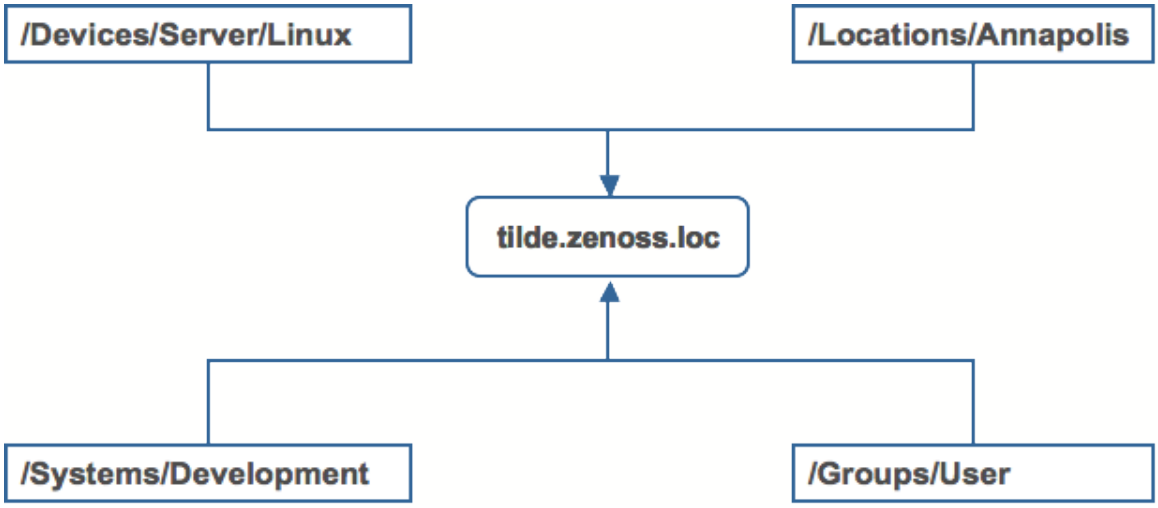
   This uses ZENHOME to send the page from the localhost; however, you can use any page command customizations you want. In this case $RECIPIENT is actually the paging address for the user, as set in the settings for each user.

Chapter 7. Organizers and Path Navigation in Zenoss

1. About Organizers and Path Navigation

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

Figure 7.1. Device Groupings



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

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

2. Classes

The most important organizers in Zenoss are classes, which comprise:

- Device classes
- Event classes
- Service classes
- Product classes

Templates and zProperties 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.

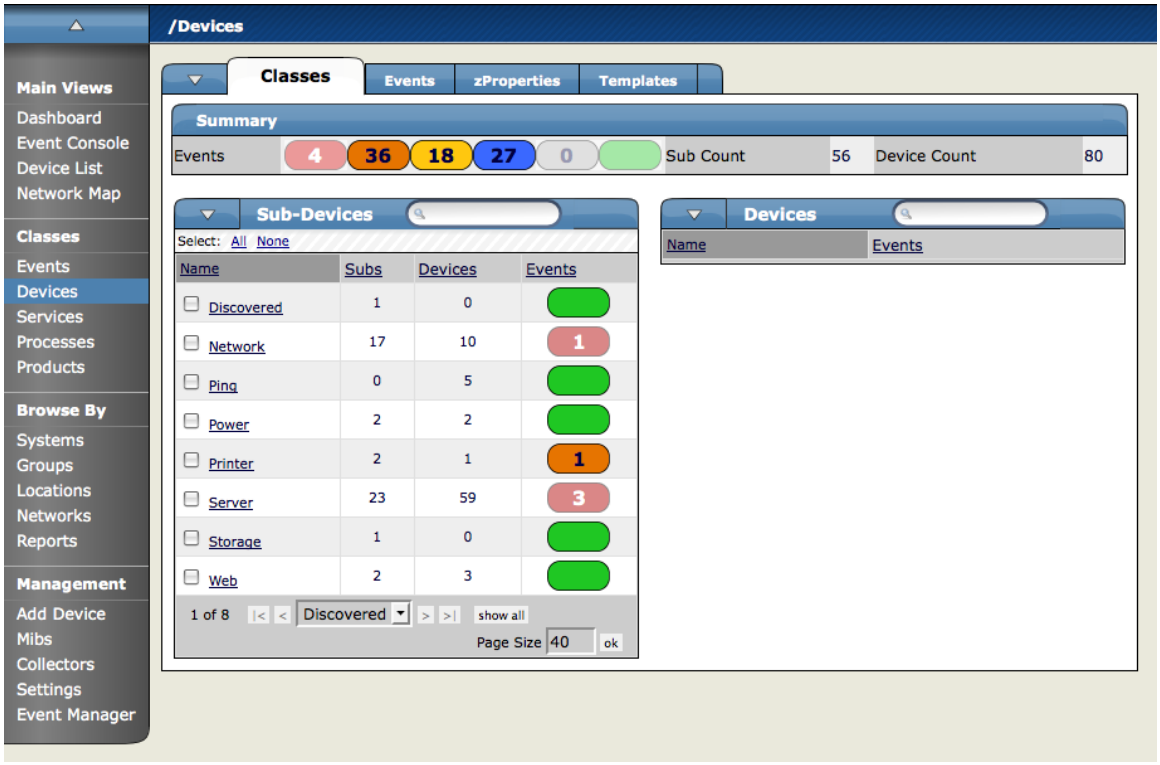
The following procedures use device classes and sub-classes, but the same concepts apply to event classes, service classes, and product classes. When you add a device to Zenoss, you should (after providing the network name or IP address), at a minimum, specify its device class. Templates and zProperties can be set at any level in the device class hierarchy.

Viewing Device Classes

To see all of the devices in a device class, select Devices from the Navigation menu.

The Device Classes tab appears.

Figure 7.2. Device Class Tab



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

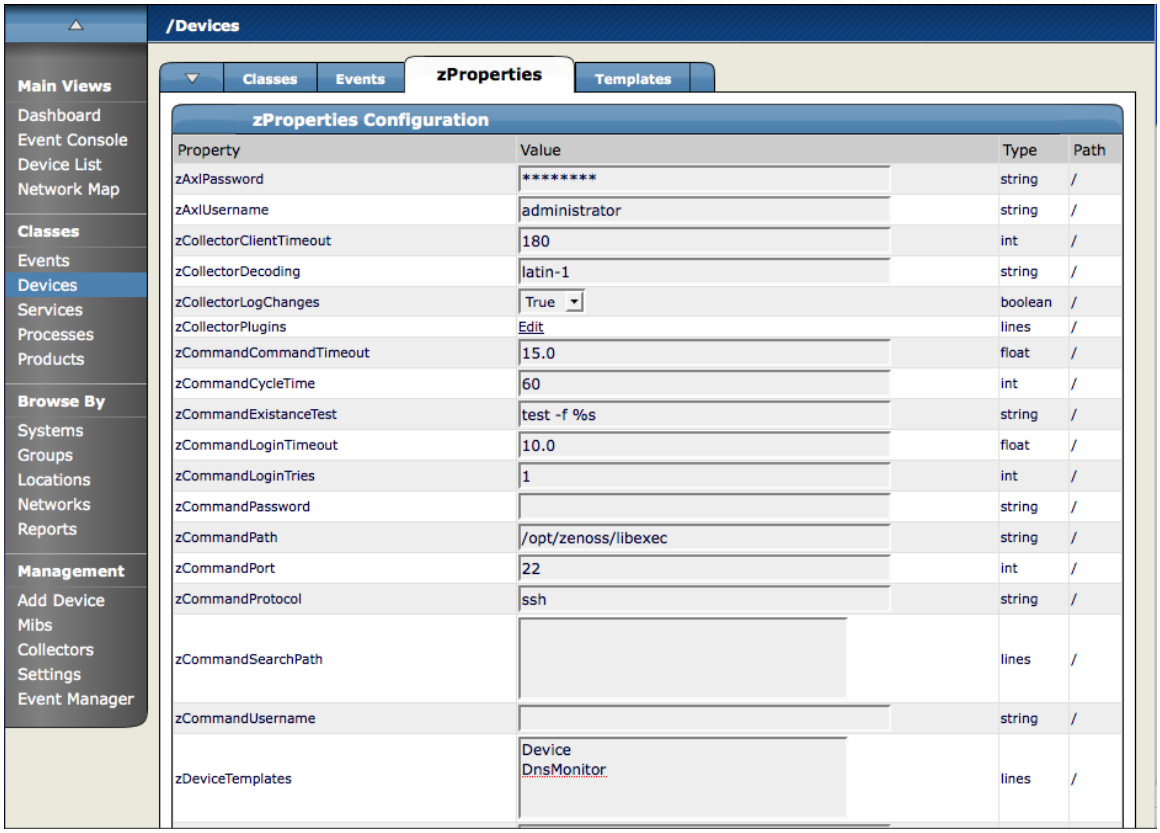
2.1. Setting zProperties at the Class Level

To set zProperties at the Device Class Level:

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

   The zProperties tab for this Device Class appears.

Figure 7.3. Device Class zProperties tab



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

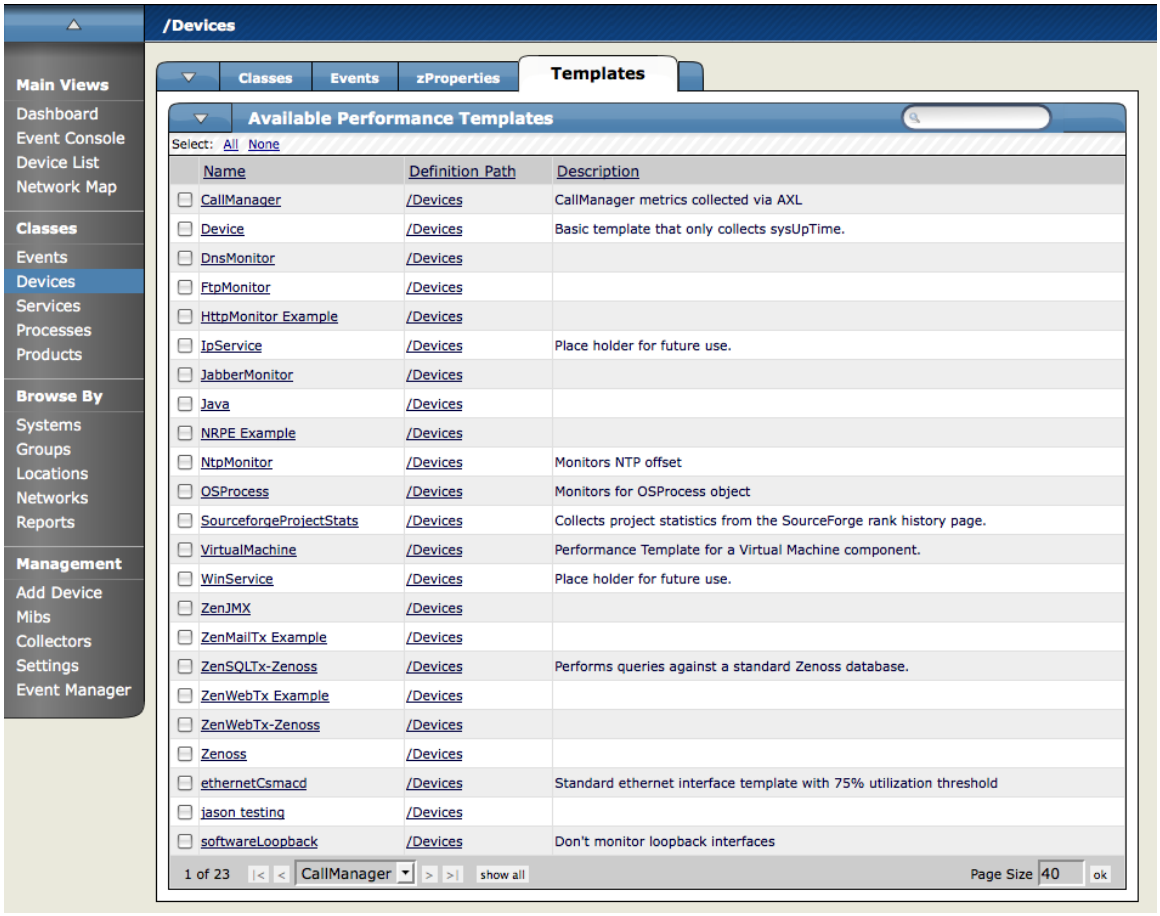
2.2. Defining and Applying Templates at the Class Level

To define Templates at the Device Class Level:

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

   The Templates tab for this Device Class appears.

Figure 7.4. Device Class Template tab



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

2.3. Creating New Classes

To create a device class:

1. From the class level where you want to add the organizer, navigate to the Classes tab.
2. From the SubClasses table menu, select Add New Organizer, and then enter a name for the organizer.

To add devices to this device class:

1. Navigate to the device list.
2. Select the devices you want to add to the class.
3. From the table menu, select Move to Class, and then select the class to which you want to move the device.

3. Systems

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

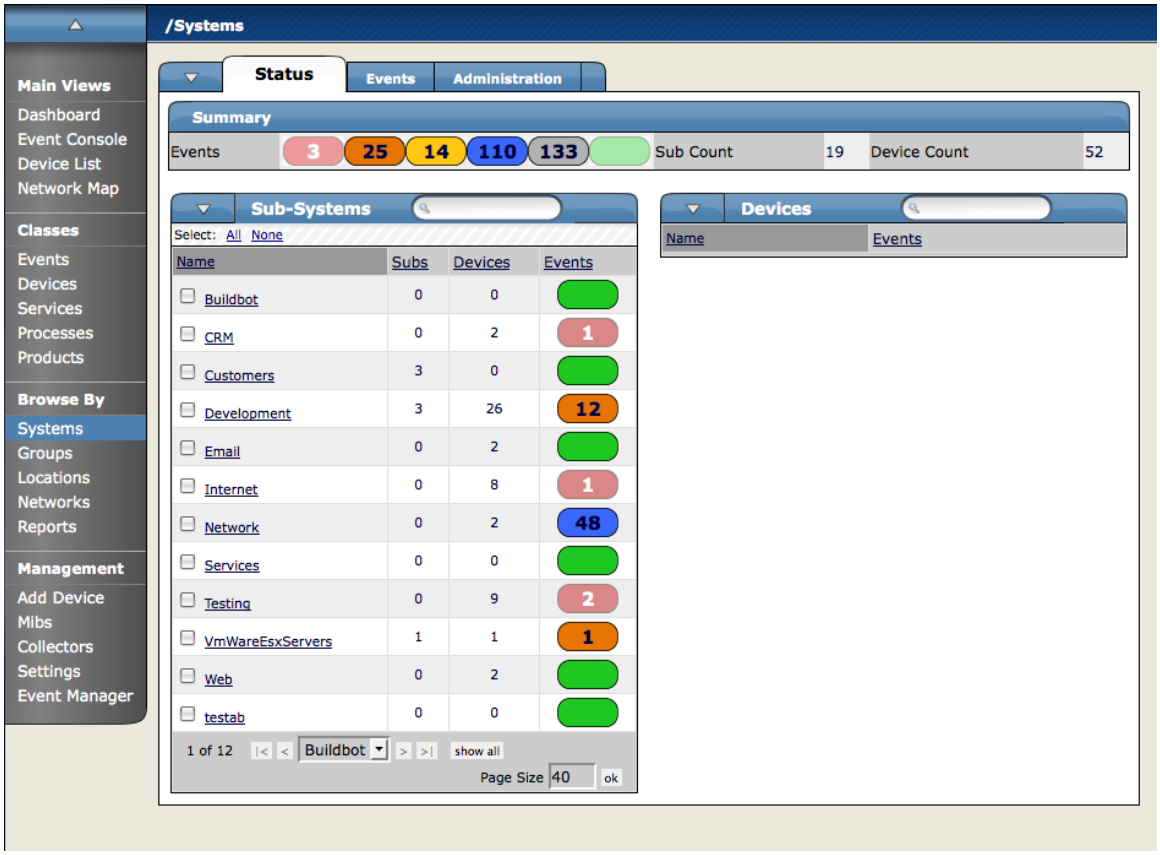
3.1. Adding, Moving and Nesting Systems

To create a new system or sub-system:

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

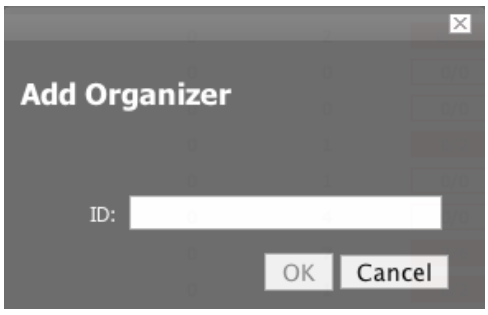
   The Sub-systems Status tab appears.

Figure 7.5. Sub-systems Status Menu



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

Figure 7.6. Add Organizer Dialog



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

4. Click OK.

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

3.1.1. Moving the Sub-System

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

1. Select one or more systems that you want to move, and hen open the Sub-Systems table menu to show the Subsystem options.

2. Select the Move Organizer option.

   The Move Organizer dialog appears.

3. Select the location where you want to move this system.

4. Click Move.

   The system is moved to the selected system. The attributes page for the newly selected system appears.

4. Groups

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

4.1. Adding Groups

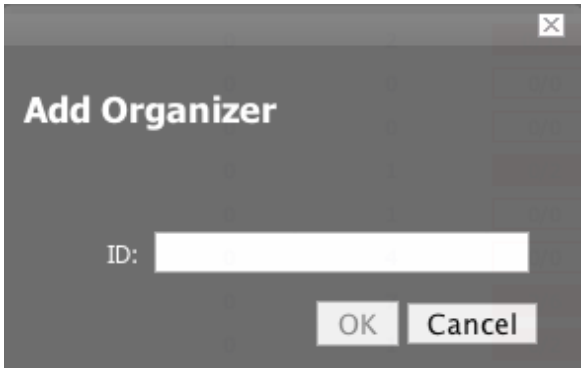
1. From the Navigation menu on the left, under Browse by, select Groups.

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

3. Select the Add New Organizer option.

   The Add Organizer dialog appears.

Figure 7.7. Add Organizer Dialog



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

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

4.1.1. Moving Groups

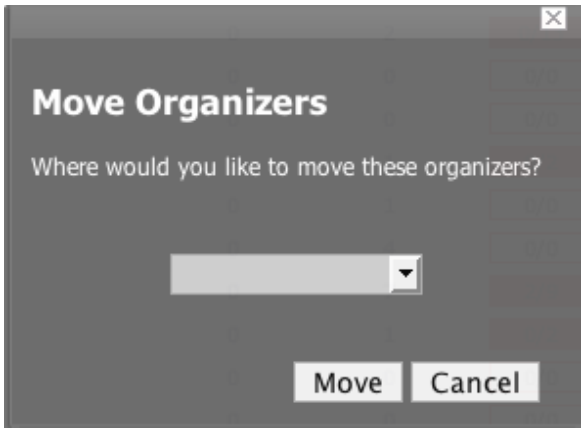
To move the sub-group into another group or sub-group:

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

2. Select the Move Organizer option.

   The Move Organizer dialog appears.

Figure 7.8. Move Organizer



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

4. Click Move.

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

5. Locations

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

5.1. Google Maps Geographic View of Network Health

5.1.1. Overview

Note

The setup instructions in this section apply to Zenoss Core implementations only. If you are a Zenoss Professional or Enterprise version user, please contact Zenoss Client Services for assistance setting up this feature.

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

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

5.1.2. API Key

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

To obtain a free Google Maps API key:

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

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

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

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

5.1.3. Setting an Address for a Location

1. From the left navigation menu, select Locations.

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

   An Edit dialog slides down.

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

4. Click the Save button.

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

5.1.4. Clearing the Google Maps Cache

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

5.1.5. Network Links

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

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

- Any event on the interface at either end of the connection.

5.1.5.1. zDrawMapLinks Property

Calculating network links on the fly is an expensive procedure. If you have a large number of Devices that have been assigned Locations, drawing those links on the map may take a long time. In order to save time, you can tell Zenoss not to attempt to draw links for specific networks (for example, a local network comprising many devices that you know does not span multiple Locations).

1. Navigate to the Network and click the "zProperties" tab.

2. Set the "zDrawMapLinks" property to "False."

3. Click "Save."

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

5.1.6. Google Maps Example

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

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

5.2. Adding, Moving, and Nesting Locations

To create a new Location or Sub-Location:

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

   The Sub-locations Status tab appears.
2. Open the Sub-Locations table menu to show Sub-Locations options.
3. Select the Add New Organizer option.

   The Add Organizer dialog appears.
4. In the ID field, Enter the name for the new Sub-location.
5. Click OK.

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

5.2.1. Moving Sub-locations

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

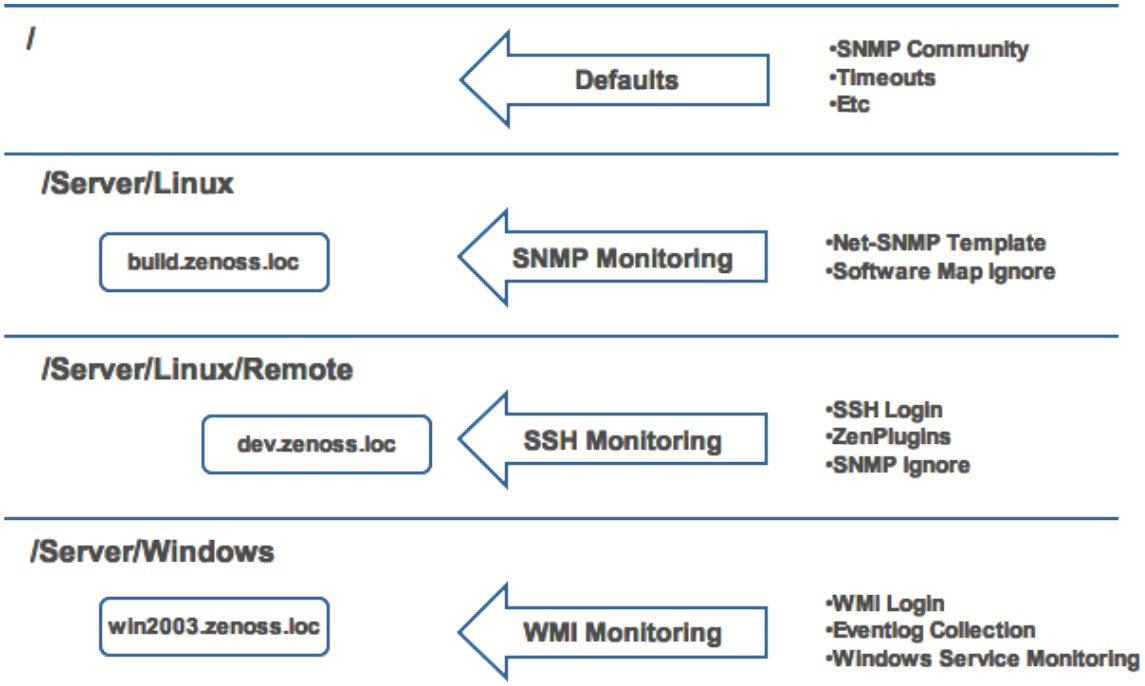
1. Select the location from the location list by clicking the check box next to the locations that you want to move and then from the Sub-Locations table, select the Move Organizer option. The Move Organizer dialog appears.
2. From the pop-up menu, select where you want to move this location.
3. Click Move.

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

6. Inheritance

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

Figure 7.9. Zenoss Device Class Tree and Inheritance



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

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

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

The determination of which templates apply to what objects is called binding. There are two steps to binding: first is determining which template names are appropriate and second is locating the correct templates with those names. The template names used for components such as FileSystems, Interfaces, etc is the same as the components' meta type. For example, file systems use templates name FileSystem. Devices are more complex because users can modify the names of the templates to be used and can specify more than one name. This list of template names

is stored in the zDeviceTemplates zProperty. This zProperty can be edited directly on the zProperties page of any Device or Device Class. It can also be edited through the Bind Templates menu item and dialog available from the Templates page of any Device or Device Class. The second step of template binding is finding the correct template with the given name. As with zProperties, templates can be defined directly on a Device or they can be inherited from one of the Device Classes above it. The first templates found with the correct names searching up the hierarchy are used. Any similarly named templates farther up the hierarchy are ignored.

7.1. Template Binding Example 1

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

7.2. Template Binding Example 2

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

Chapter 8. zProperties

1. About zProperties

zProperties are properties that Zenoss uses to specify various items that control the information in the system. zProperties are very powerful, and can automate many repetitive tasks you typically would perform individually.

zProperties can be configured for:

- All items
- An individual device
- Multiple devices in the device hierarchy

Additionally, zProperties settings can be added to any ZenPacks you create.

zProperties exist for:

- Events
- Devices
- Services
- Networks

2. Event zProperties

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

Table 8.1.

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

3. Device zProperties

To access Device zProperties, from the left navigational menu, select Devices and then click the zProperties tab. You can also select any of the Device Classes and the zProperties tab to set zProperties anywhere in the Device

hierarchy. To set zProperties for an individual Device, navigate to that device, open the page menu, select the More option and then the zProperties tab.

		when collecting SNMP information.
zSnmpCommunity	string	Community to be used when collecting SNMP information. If it is different than what is found by ZenModeler, it will be set on the modeled device.
	zProperties	

Table 8.2.

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

4. Service zProperties

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

Table 8.3.

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

5. Process zProperties

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

Table 8.4.

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

6. Network zProperties

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

Table 8.5.

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

7. Manufacturer zProperties

Table 8.6.

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

Chapter 9. Device Inventory and Configuration

1. What is Zenoss Inventory and Configuration?

Inventory and configuration (also known as modeling) is the process by which Zenoss:

- Populates the device database
- Collects information about the devices in the system

Zenoss models devices through the discovery process, individually or in groups. It accomplishes discovery through the zendisc daemon.

2. How Does Zenoss Model Devices?

To model devices, Zenoss can use:

- SNMP
- SSH
- WMI
- Telnet

Depending on the method, the device model produced contains varying amounts of information.

3. The ZenModeler Daemon

Zenoss employs the ZenModeler daemon to model devices. ZenModeler iterates over the list of devices in the system and attempts to discover sub-components -- such as network interfaces, file systems, process, and IP services -- of each device.

By default, Zenoss remodels each known device every 720 minutes (12 hours). You can change this interval by editing the value of Modeler Cycle Interval (mins) in the collector's configuration.

For larger deployments, modeling frequency may impact performance. In such environments, Zenoss recommends running the modeling process once daily from a cron job.

4. Add a Device

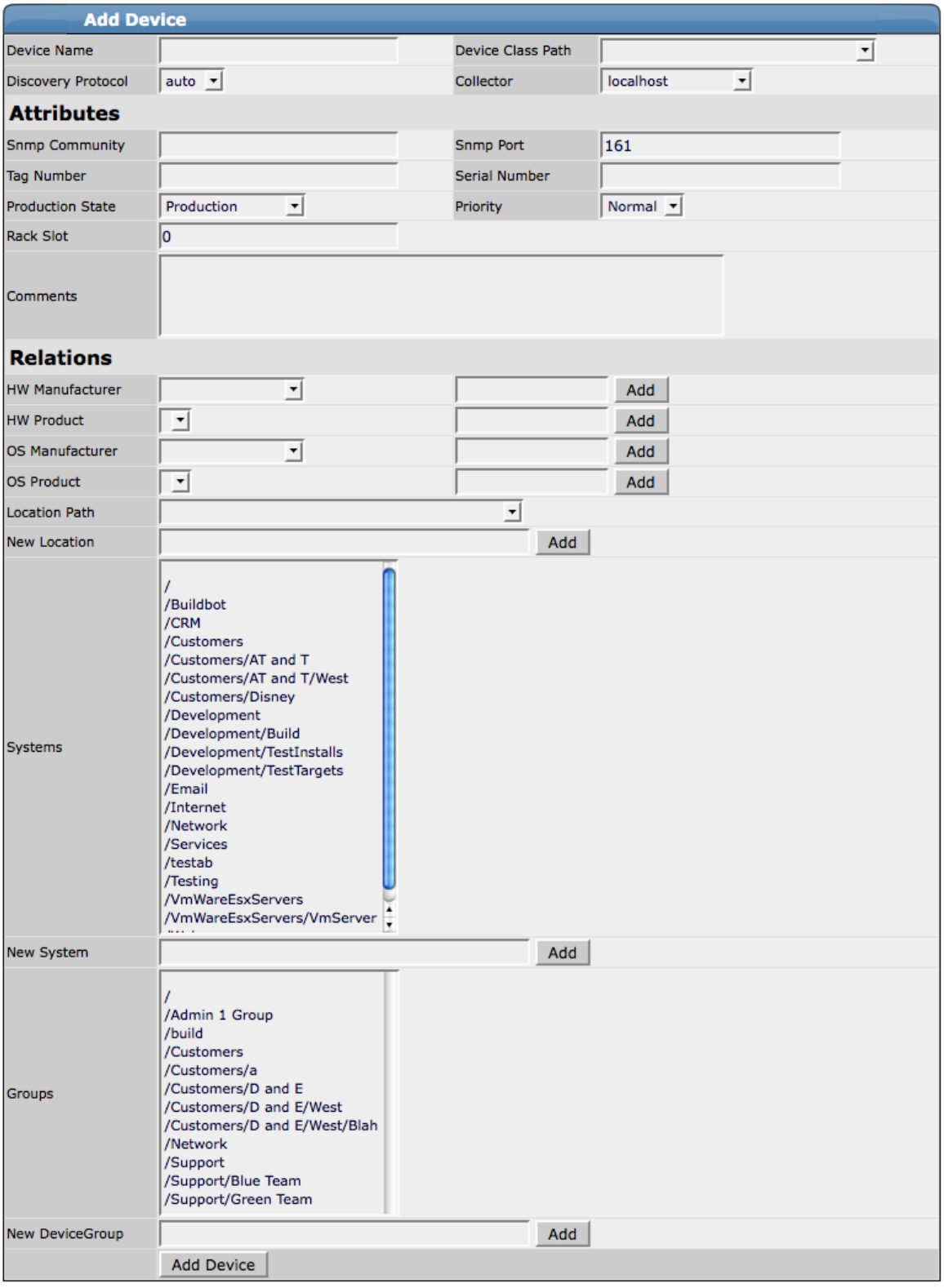
Zenoss adds, models, and monitors the devices you add to the system.

Follow these steps to add a single device to Zenoss.

1. From the Navigation menu, select Add Device.

   The Add Device page appears.

Figure 9.1. Add Device



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

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

- Device Class Path - Select a device class to which this device will belong. For example, if the new device is a Windows server, then choose /Server/Windows.
- Discovery Protocol - Select a discovery protocol (auto or none).

Note

Device Name, Device Class Path, and Discovery Protocol are the only required fields to add the device. Zenoss recommends that you continue without adding more information or making selections, as information you enter or select may conflict with information Zenoss discovers about the device.

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

3. Scroll to the bottom of the page, and then click Add Device.

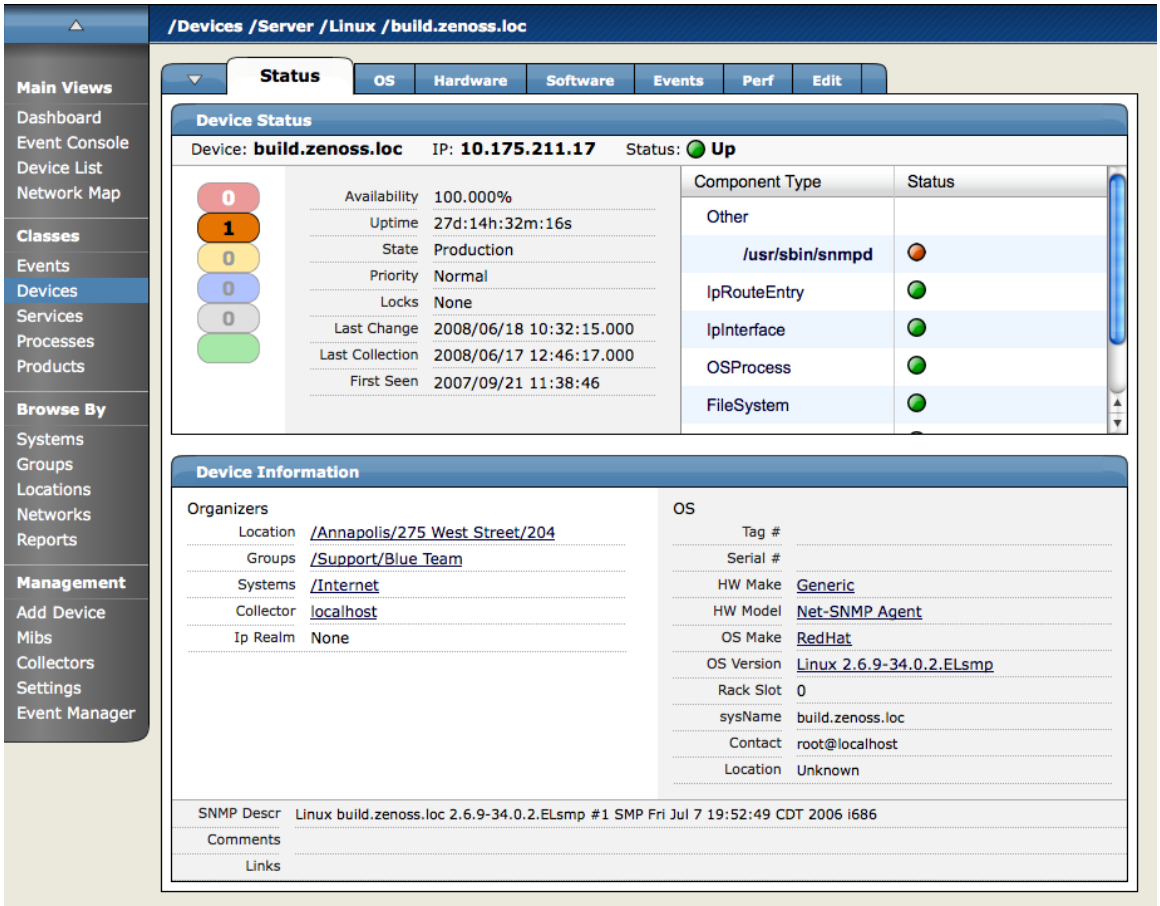
   A Status page appears, showing a log of the operations Zenoss is using to gather information about the device.

4. Scroll to the bottom of the Status page, and then click the link that appears:

   Navigate to device `DeviceName`

   The Main Device page appears, showing the Status Tab.

Figure 9.2. Main Device Page



5. Add a Device with Context

By default, when you add a device, Zenoss places it in the /Discovered class. You can choose instead to add a device with context, which adds the new device to a location you choose in the hierarchy.

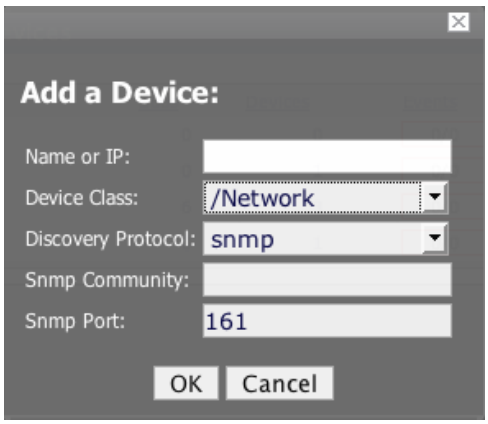
To add a device with context:

1. Navigate to the location in the devices hierarchy where you want to add the device.

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

   The Add Device dialog appears.

Figure 9.3. Add a Device Dialog



3. Complete the Name and Discovery Protocol fields. (For descriptions of valid values for these fields, refer to the section titled Add a Device.) the Device Class value is the class you selected in the devices hierarchy.

4. Click OK.

   The Device is added into the selected device class. The main Device page appears, showing the Status page.

6. Discover Devices

During discovery, Zenoss "SNMP-walks" your selected network and routing tables, models each discovered device, and then adds them to the database. Zenoss accomplishes this through the ZenDisc daemon.

Note

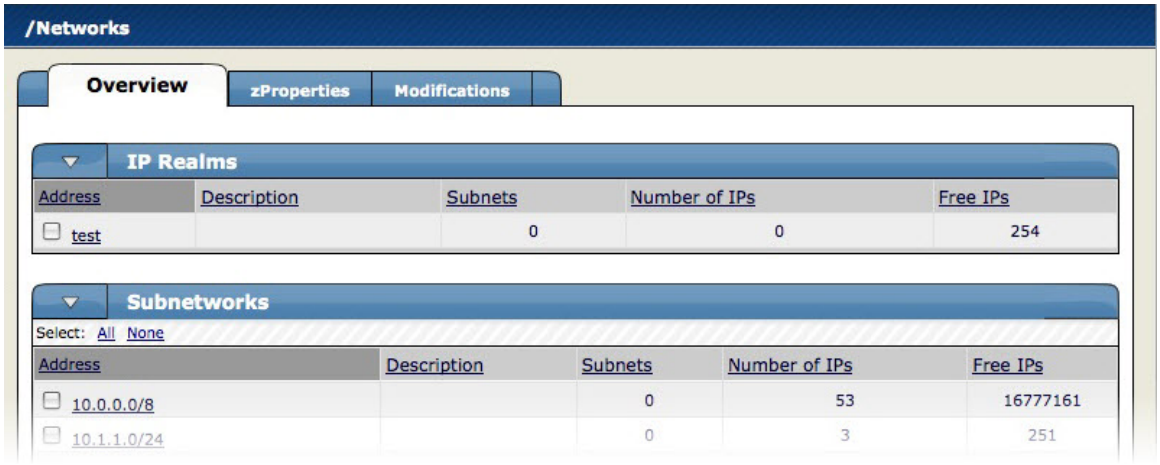
To perform discovery, the machine on which Zenoss is installed must have an SNMP agent running.

To add all of the devices on a network or subnetwork to Zenoss:

1. From the Navigation menu, select Networks.

   The Networks page appears.

Figure 9.4. Networks Page (Overview Tab)

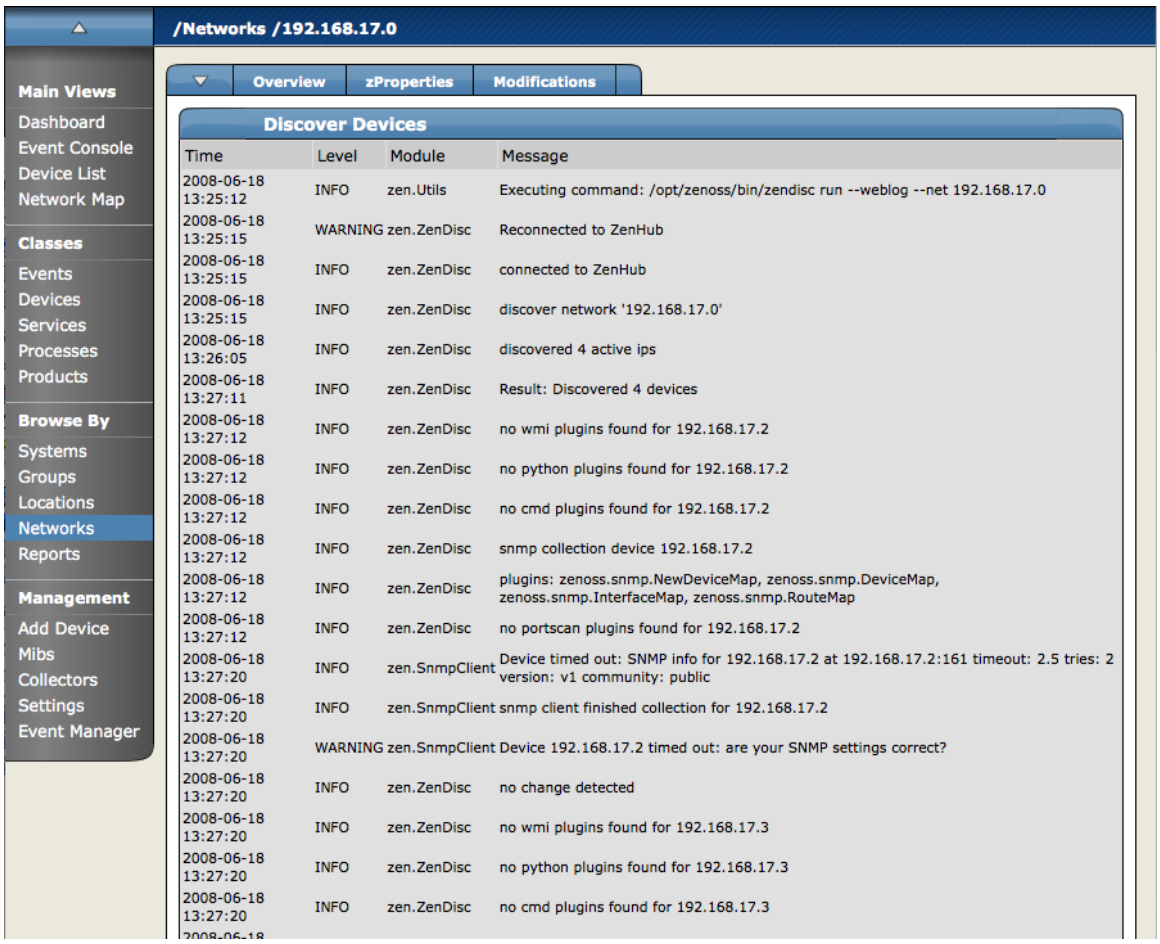


2. Select the network from which you want to add devices.

3. Open the Subnetworks table menu, and then select Discover Devices.

   The Discover Device page appears. This page shows the status of all the device collections in progress.

Figure 9.5. Device Auto-Discovery



Zenoss first models the monitoring machine, and then walks through the routing tables of all routers it locates. Auto-discovery continues while valid SNMP access is found or until a network is discovered in the DMD that has its zAutoDiscover property set to False.

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

6.1. Classifying Discovered Devices

After Zenoss discovers devices, it is important that you move those discovered devices to an appropriate location in the hierarchy. In general, servers are organized by operating system. If Zenoss discovers Windows devices, for example, you might choose to relocate them to /Server/Windows. Moving devices to their correct hierarchy location initiates the monitoring process.

6.2. Adding Information to a Device Record

You may want to add details about a discovered device, such as its Business System or Location. To add information, select the device in the list, and then go to the Edit tab.

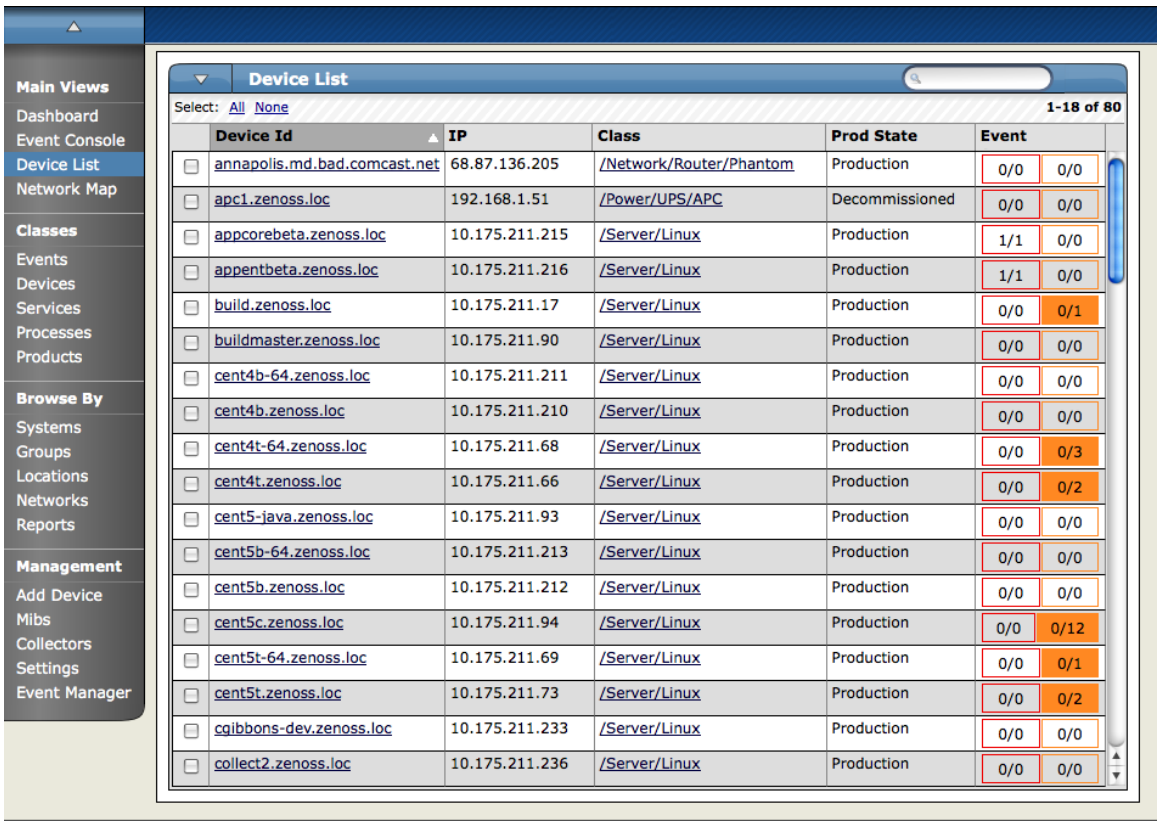
7. Device List

The Device List shows all devices in the system. From here you can:

- Search for devices
- View event summaries
- Perform device management tasks

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

Figure 9.6. Device List



7.1. Managing Multiple Devices from the Device List

Use the Device list page menu to manage multiple devices. Select devices in the list (check the box next to each one you want to select), and then select an option from the page menu.

Available options for managing multiple devices are:

- Move to class – Move devices to new classes.
- Set Groups – Assign devices to groups.
- Set Systems – Assign devices to systems.
- Set Location – Assign devices to locations.
- Set Monitor – Assign monitors for collecting from selected devices.
- Delete devices – Remove devices from the system.
- Lock devices – Provide configuration locks for devices.

8. Individual Device Tabs

Zenoss assigns certain properties and attributes to each discovered device. It categorizes this information and makes it available on the Device page tabs. These are:

- Status
- OS
- Hardware
- Software
- Events
- Perf
- Edit

The following sections provide more information about each Device page tab.

8.1. Status Tab

The Status tab is the default tab that appears when you click a device in the device list.

Viewing Device Status

The Device Status table appears on the Status tab, and provides important device status at a glance. The number of events, grouped by severity, are found on the left side of this table. You can click this "event rainbow" to view the list of events for the device. The left side of the Device Status table also shows device availability, uptime, last collection, and modified time.

Viewing Component Status

On the right side of the Device Status table is the Component Status list. Each item in the list is a type of device component; these are:

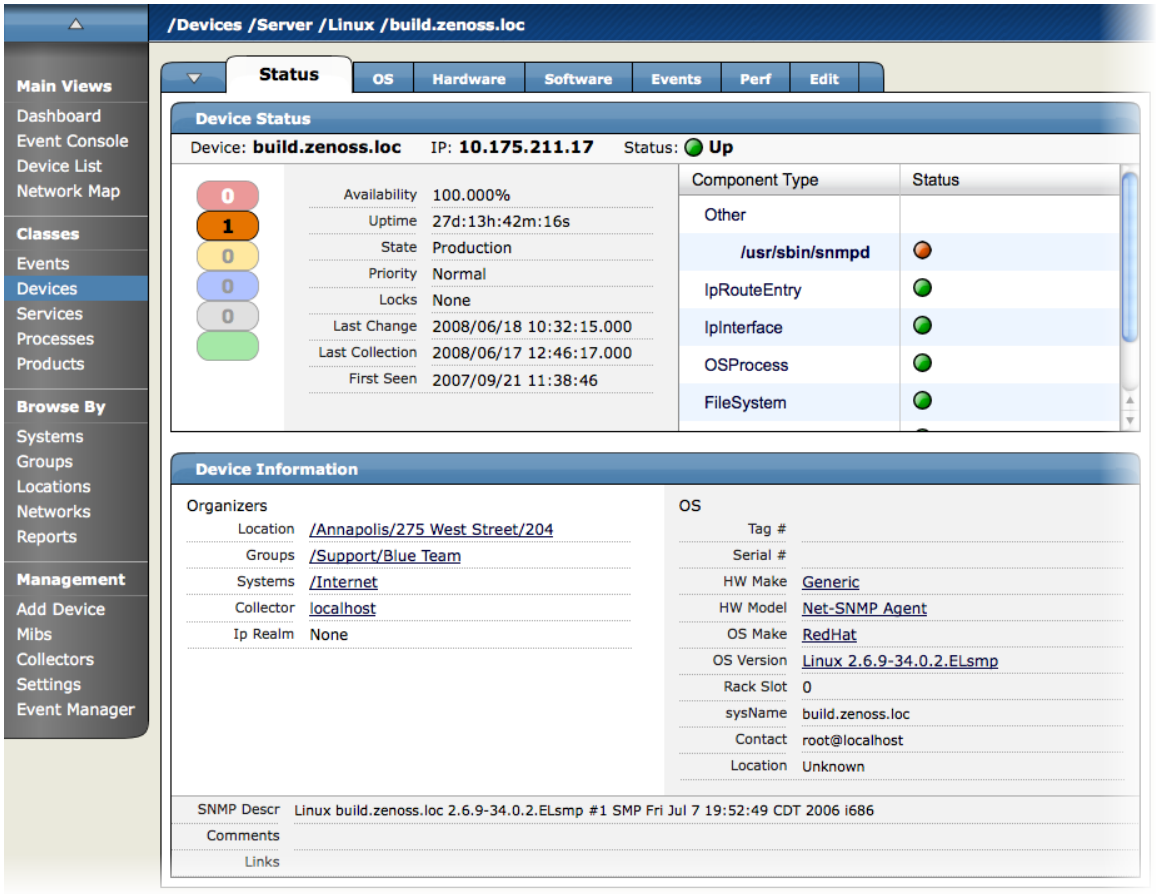
- IPService
- WinService
- IpRouteEntry
- IpInterface
- CPU
- FileSystem
- Other

The status of each device component type is determined by the collective status of the monitored components of the same type. For example, if the IpService status is green, then all monitored IpServices on this device are functioning normally. If there is an event related to a monitored IpService, then the component and highest severity event associated with that component are displayed in the status.

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

Click the component type to go the device OS tab, where you can manage device components.

Figure 9.7. Device Page (Status Tab)



Example: View Device and Component Status

Follow the steps in this example to create events related to a device and then view the impact on device status.

1. Click the OS tab of any device.

2. In the Interfaces area, check to see if there is an eth0 entry. If not, then select Add IpInterface from the Interfaces table menu and add it.

3. Click the eth0 entry and set the value of Monitor to True.

4. Click Save.

5. From the Device Status page, update the table by sending an event linked to a component. To do this:

   a. On the navigation menu, click Events.

   b. Select Add Event from the Event Classes page menu.

   c. Enter values for these fields:

      - Device - Enter the device name.
      - Component - Enter eth0.
      - Severity - Select Critical.

   d. Click OK.

6. Send a second event, using the same "add event" procedure. Enter values for these fields:

   - Device - Enter the device name.
   - Component - Enter "Not a known component."
   - Severity - Select Error.

7. Click OK, and then refresh the Device Status page.

   You should see that the IpInterface type group has a component "eth0." It is marked with a red status, indicating that a critical event has been received.

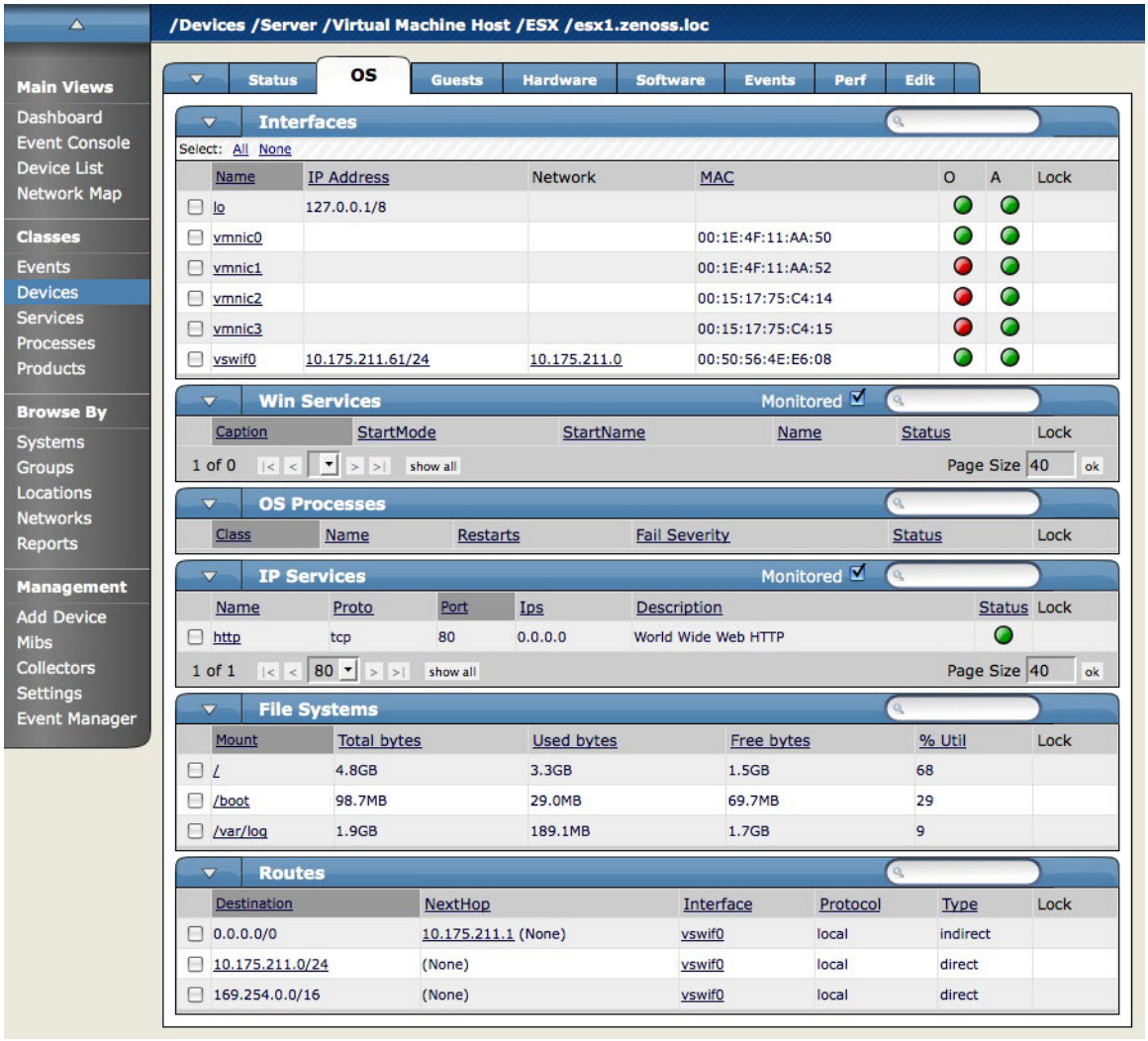
   If you click the status bubble, Zenoss displays the Events page for the device. A new type group (Other) appears with the component type you selected when creating the second event. The status color of this component is orange, representing an Error event.

8.2. OS (Operating Systems) Tab

The OS tab contains logical operating system components such as:

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

Figure 9.8. Device Page (OS Tab)



8.2.1. File System Monitoring

The File Systems area lets you view file system status, if file system monitoring is enabled. To enable file system monitoring, select Monitoring from the File Systems table menu, and then select the Enable option in the dialog.

Note

File systems can be monitored only if the system has a valid HOST-RESOURCES MIB.

Zenoss monitors the amount of blocks used and shows:

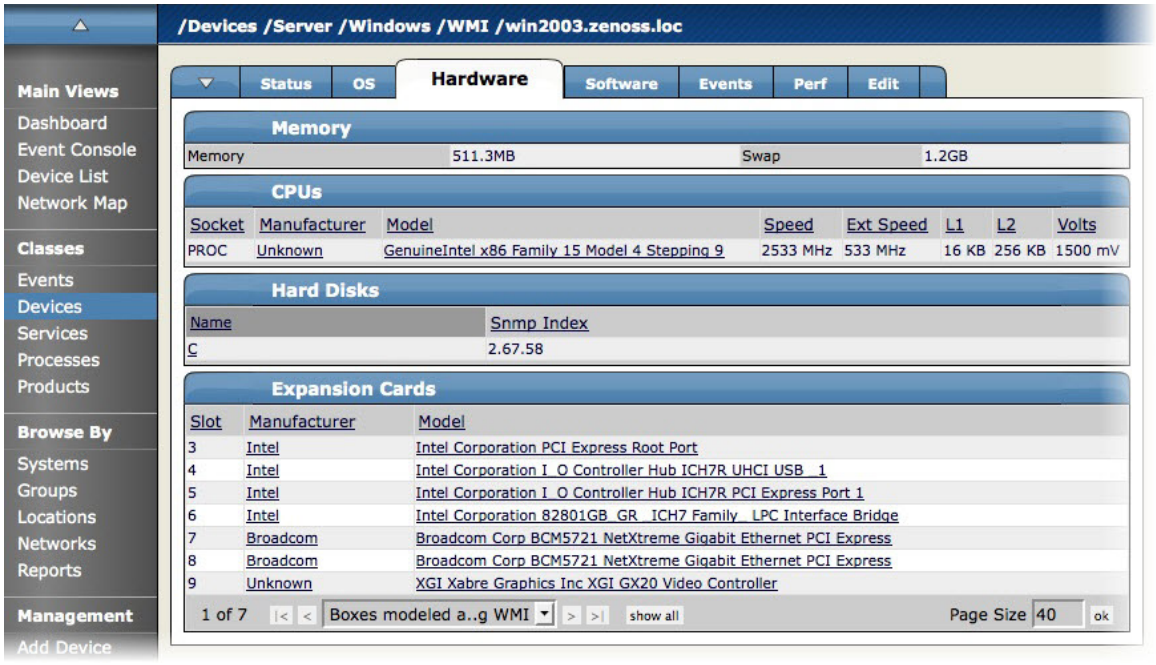
- Total bytes
- Available bytes
- Used bytes
- Percentage used

You can set thresholds to alert when a specific event occurs, such as when the disk reaches 90% utilization.

8.3. Hardware Tab

The Hardware tab gives you information about the device's available and used memory, available and used swap, and information on the CPUs.

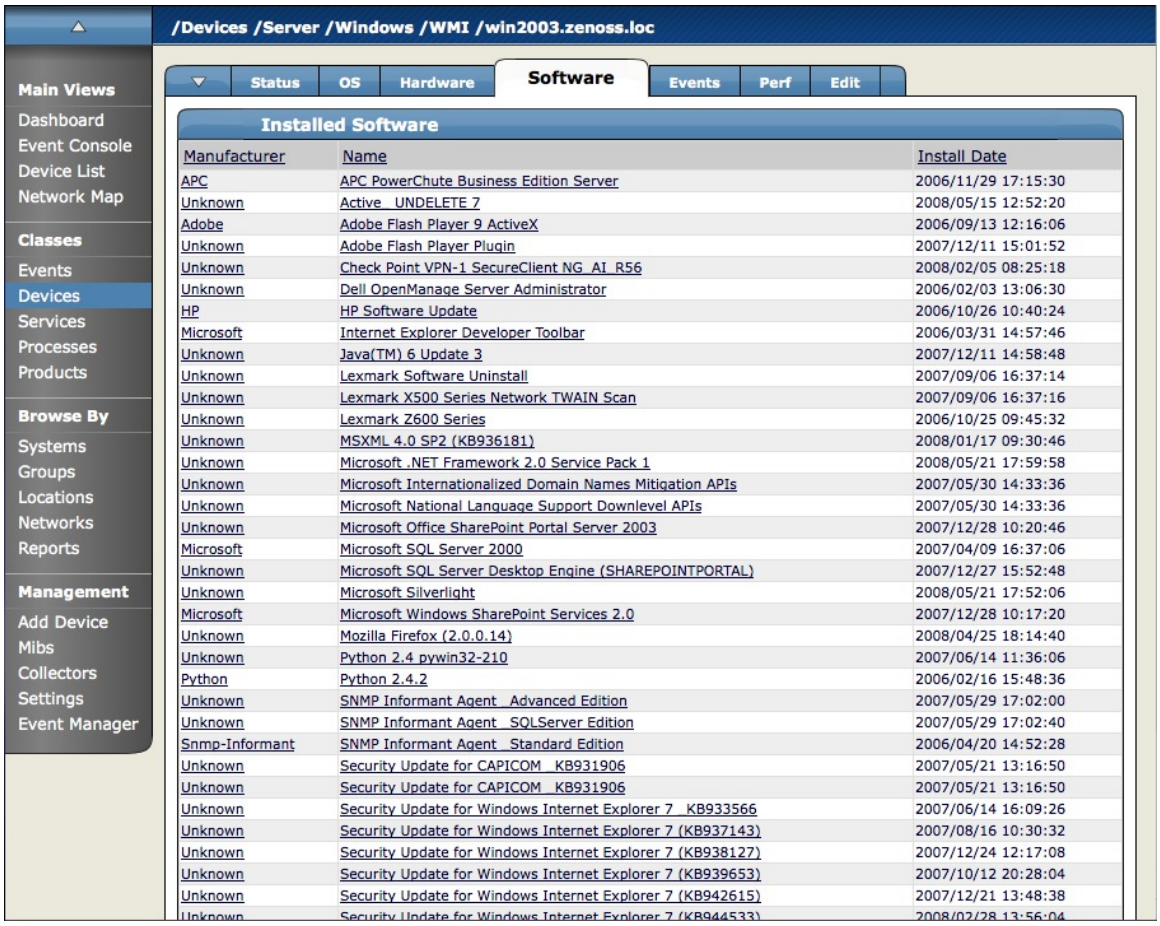
Figure 9.9. Device Page (Hardware Tab)



8.4. Software Tab

The Software tab lists all of the software installed on the device. Click a column head (Manufacturer, Name, or Install Date) to sort the list by the header value.

Figure 9.10. Device Page (Software Tab)

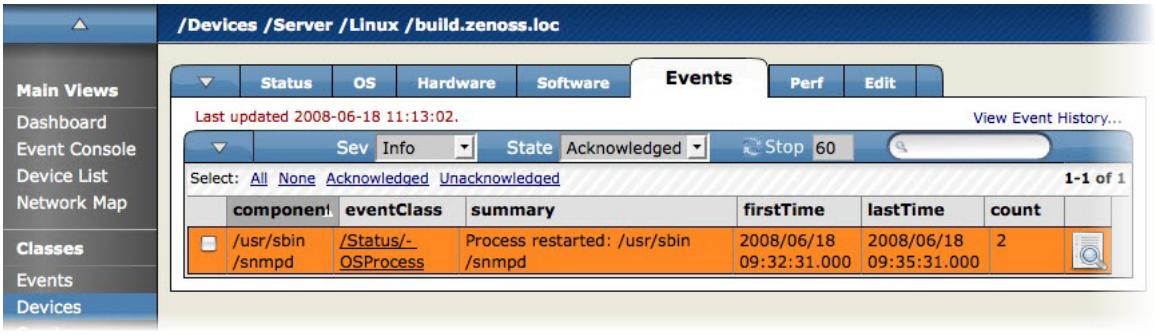


8.5. Events Tab

The Events tab provides events information that is scoped to the device. From here, you can:

- Sort event information by component, event class, or count
- Move, map, and acknowledge events
- Filter events by severity state, or by using a regular expression applied to all events

Figure 9.11. Device Page (Events Tab)



8.6. Performance (Perf) Tab

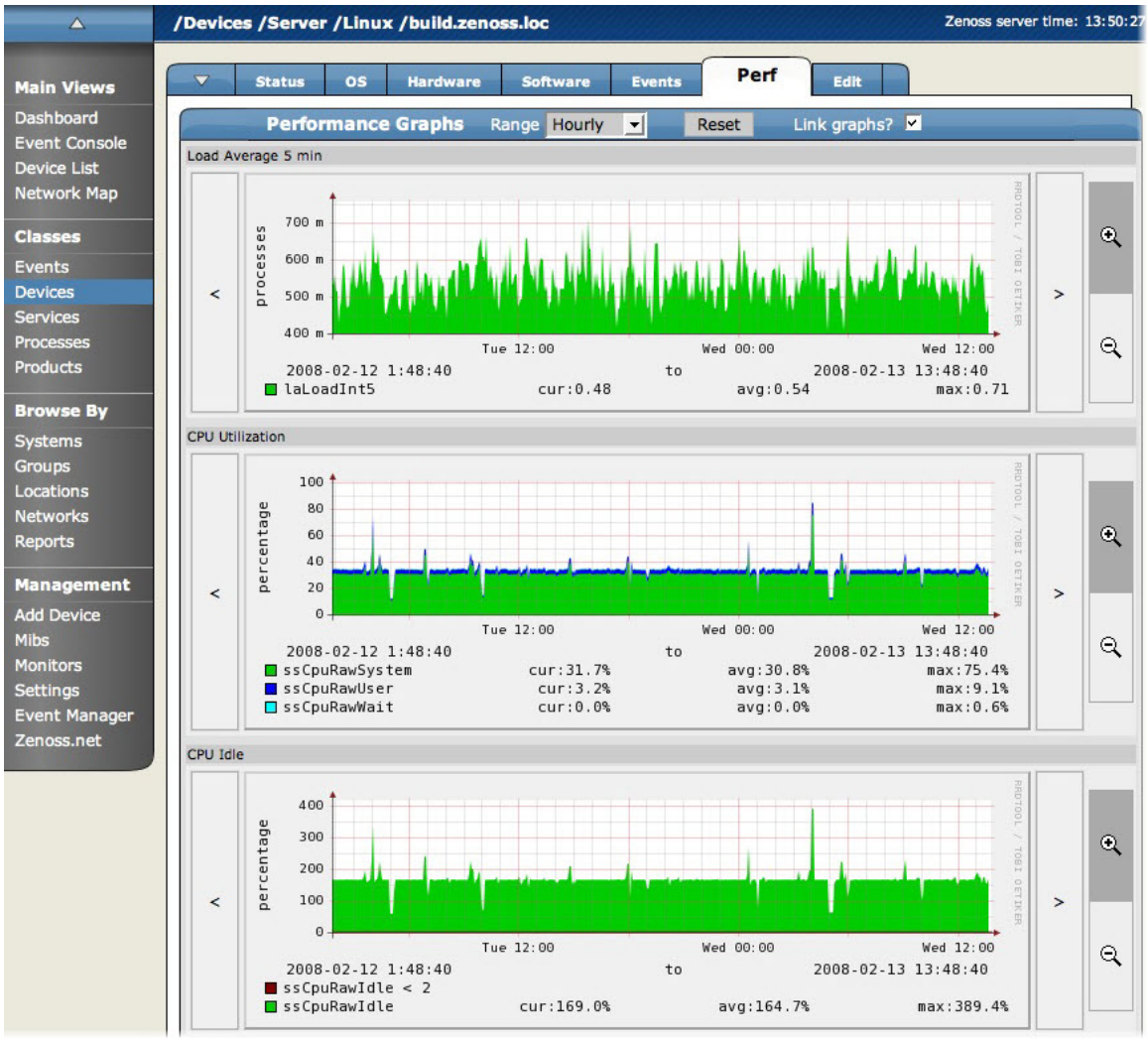
The Perf tab shows performance graphs defined for the currently selected device.

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

From this tab, you can control these performance graph options:

- Range - Select the frequency of graph updates. You can select Hourly, Daily, Weekly, Monthly, or Yearly.
- 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. Uncheck the Link graphs? option to control each graph individually.
- Stop - Turns off automatic refresh of the graphs.

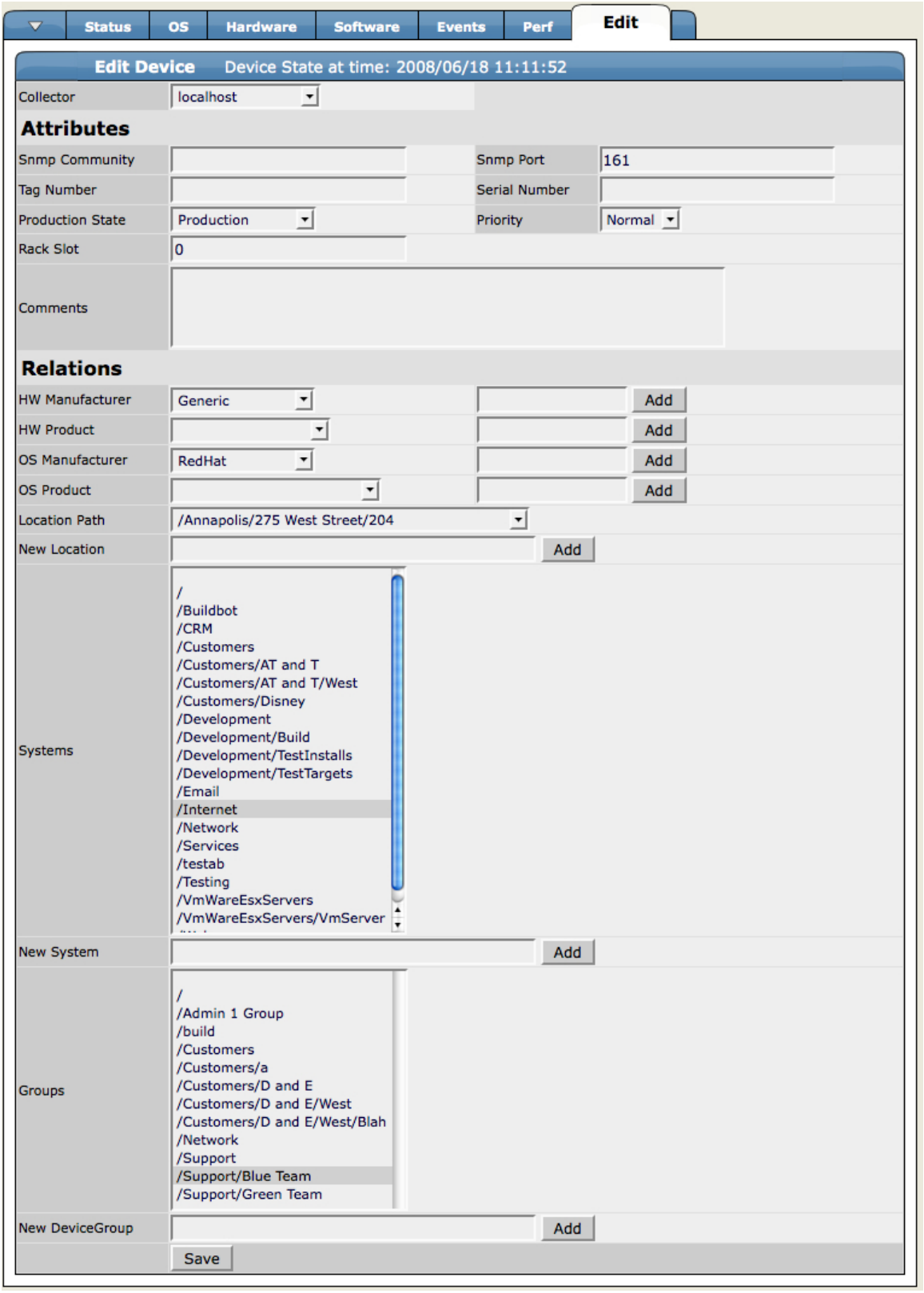
Figure 9.12. Device Page (Performance Tab)



8.7. Edit Tab

Use the Edit tab to change device properties.

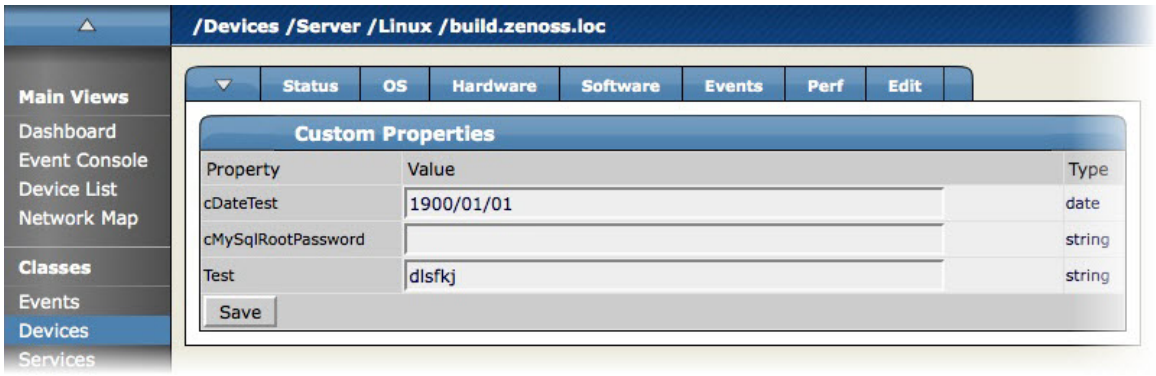
Figure 9.13. Device Page (Edit Tab)



8.8. Device Custom Properties

The Custom Properties page allows you to set values in the custom fields that you define in custom schema. To access this page, open the Device page menu, and then select More > Custom.

Figure 9.14. Device Page (Custom Properties)



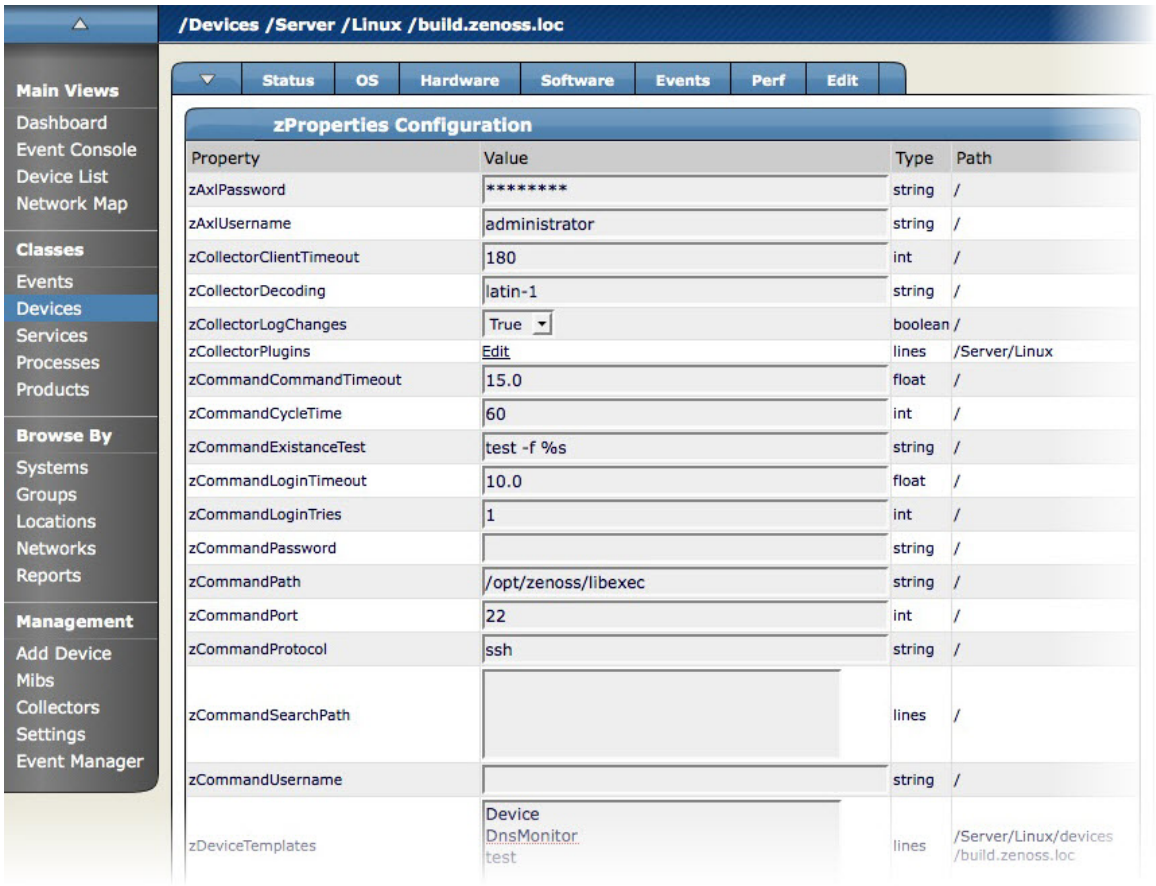
8.9. Device zProperties

From the Device zProperties page you can configure zProperties for all devices or individual devices.

To configure zProperties for all devices, click the zProperties tab from the Device Overview tab. To configure zProperties for an individual device, click the device name in the Device Overview, and then click the zProperties tab for that device.

To access the zProperties, open the device table menu, and then select More > zProperties.

Figure 9.15. Device Page (zProperties)



8.10. Device Templates

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

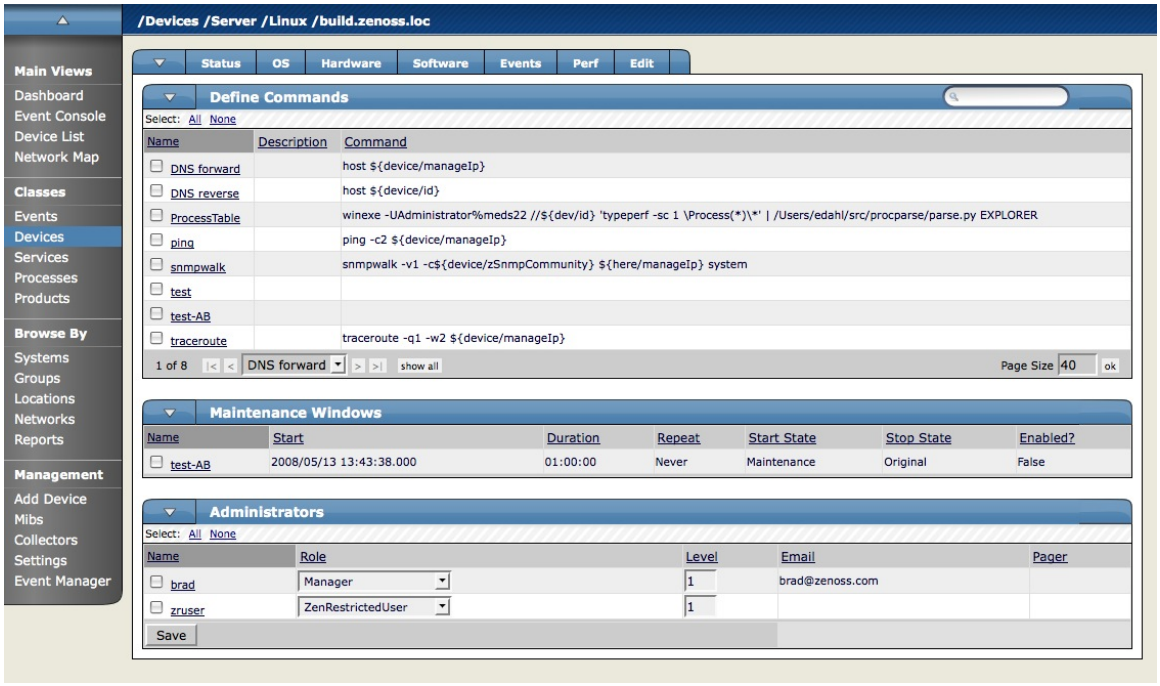
Figure 9.16. Device Page (Templates)

8.11. Device Administration

Use the Device Administration page to define commands, and to specify who holds administration capabilities for the device.

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

Figure 9.17. Device Page (Administration)

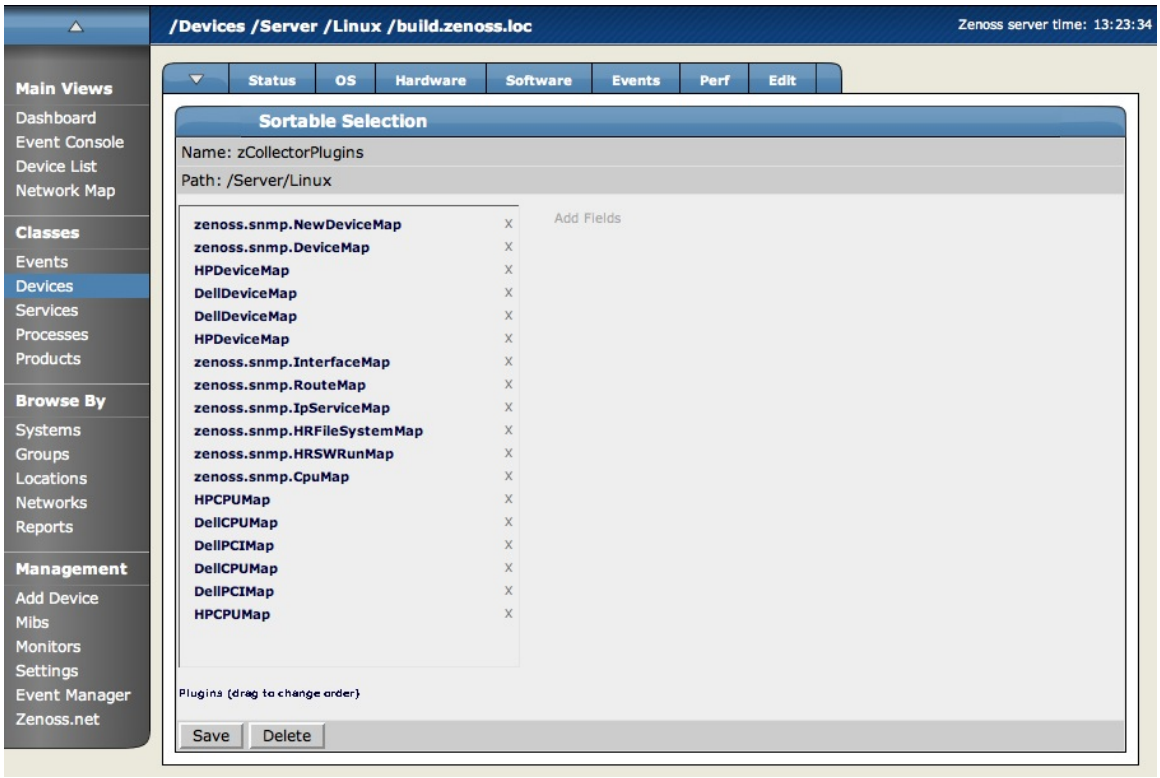


8.12. Device Collector Plugins

The Collector Plugins page lists all of the plugins available for the device.

To access the Collector Plugins page, open the Device page menu, and then select More > Collector Plugins.

Figure 9.18. Device Page (Collector Plugins)

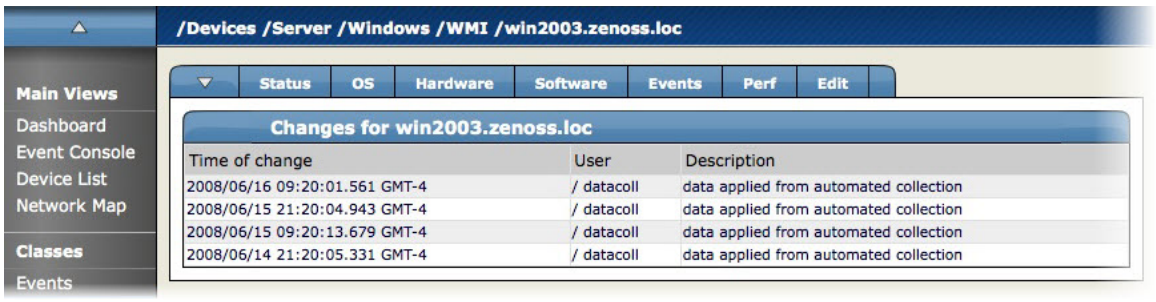


8.13. Device Modifications

The Modifications page logs user changes via the Zope interface.

To access the Modifications page, open the Device page menu, and then select More > Modifications.

Figure 9.19. Device Page (Modifications)

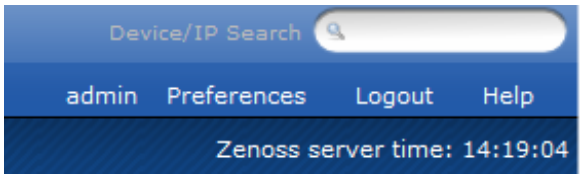


9. Searching for Devices

You can search Zenoss for devices in several ways:

- Device/IP Search - Enter a device name or IP address in this field, located at the top of the Zenoss interface.

Figure 9.20. Device/IP Search Area



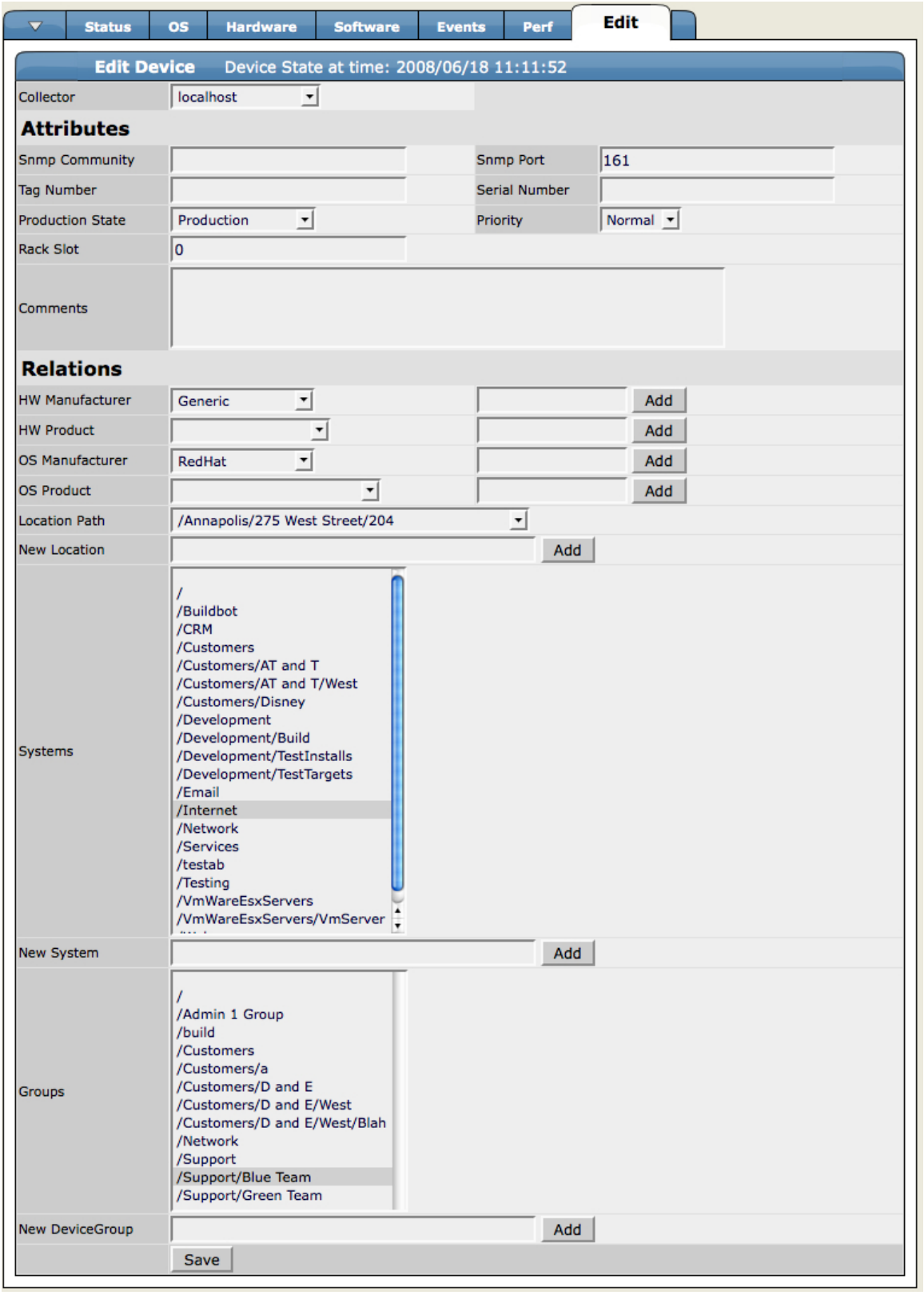
- Browse By - Select an option from the Browse By area of the Navigation menu if you do not know the device name, or if you are not searching for a specific device. You can browse by:
  - Systems - Browse by common device types, such as file server, printers, and infrastructure.
  - Groups - Browse by groups that you set up to organize your devices.
  - Locations - Browse by groups based on location.

10. Editing Device Configuration

To edit configuration selections for a device:

1. Select the device to edit, or use search to locate it.
2. Click the Device page Edit tab.
3. Change or add information to the device configuration, and then click Save. Available configuration options vary by device type.

Figure 9.21. Edit Device Configuration (Linux Server)



11. Managing Devices

To manage device attributes, use the Manage menu selections from the device page menu. These selections allow you to:

- Remodel devices
- Change a device's class
- Reset the manage IP for a device
- Rename a device
- Lock a device's configuration
- Push configuration changes to the system
- Clear heartbeats associated with a device
- Delete one or more devices

Note

You can perform many management tasks according to the inheritance and hierarchies of the various device organizers, such as class, system, group, or location.

11.1. Remodeling a Device

Remodeling forces Zenoss to recollect all configuration information associated with a device.

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Model Device.

   The Device is remodeled and the remodeling status page appears.

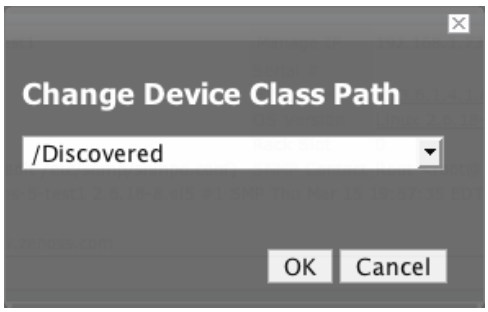
11.2. Changing the Device Class of a Device

To change the class for a device:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Change Class.

   The Device Class Change dialog appears.

Figure 9.22. Change Device Class Dialog



3. Select the new device class for this device.
4. Click OK.

   The device class for the current device is changed.

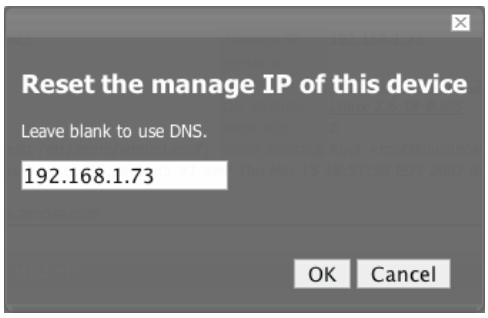
11.3. Resetting Device Manage IP

To reset the manage IP address of a device in the system:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Reset IP.

   The Rest IP dialog appears.

Figure 9.23. Reset IP Dialog



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

   The IP address for the device is reset.

11.4. Renaming a Device

To rename a device in Zenoss:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Rename Device.

Figure 9.24. Rename Device Dialog

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

   The device is renamed in the system.

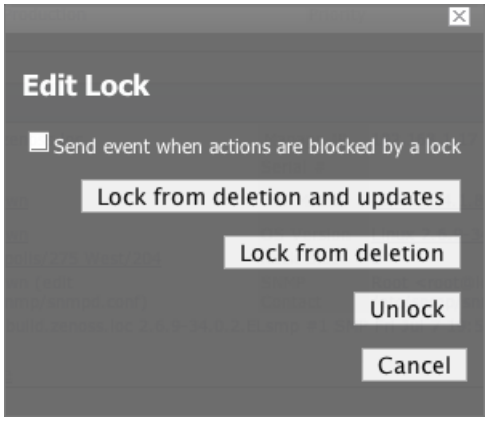
11.5. Locking Device Configuration

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. Use the Configuration Lock Editing Dialog to lock (and unlock) device configuration.

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Lock.

   The Lock Configuration Confirmation Dialog appears.

Figure 9.25. Edit (Configuration) Lock Dialog



3. To send events when actions are blocked by a lock action, select the "Send event..." option.
4. Select the type of lock you want to implement, or select Unlock to unlock the device configuration currently set.

   The lock or unlock action is implemented on the device.

11.6. Resetting the Device Community

To change the device community string:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Reset Community.

   The community for the device is reset.

11.7. Pushing Configuration Changes Back to the Zenoss System

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

To push the changes:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Push Changes.

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

11.8. Clearing Heartbeats

To clear the heartbeats associated with a device:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Clear Heartbeats.

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

The Edit tab for the ZenEventManager appears. Optionally make changes, and then click Save.

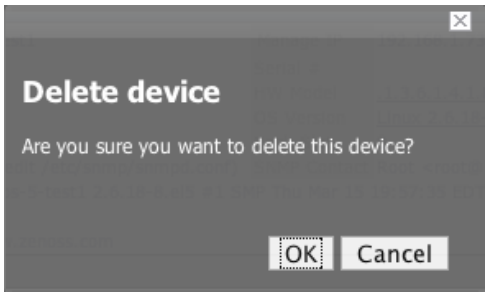
11.9. Deleting Devices From the System

To delete a device from the system:

1. Navigate to the device.
2. From the Device page menu, select Manage, and then select Delete Device.

   The Delete Device Confirmation dialog appears.

   Figure 9.26. Delete Device Confirmation Dialog

3. Click OK.

   The Device is removed from the system.

12. Modeling Devices Using SNMP

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

12.1. Testing to See if a Device is Running SNMP

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

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

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

12.2. Modeling Remote Windows Devices Using SNMP

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

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

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

12.3. Modeling Remote Linux Devices Using SNMP

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

12.4. Modeling Cisco Devices Using SNMP

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

13. Modeling Using SSH/COMMAND

In some cases a system may not offer an SNMP Agent that Zenoss can securely access. This often is the case when Zenoss is used to monitor production systems that only provide only SSH access. Additionally, when SNMP modeling is not available (or does not provide a complete set of information), command plugins can be used to model a device.

Note

Modeling command plugins differ from the performance command plugins used elsewhere in Zenoss.

Each built-in Zenoss modeling command plugin is differentiated by the platform on which it runs. To determine the platform for the device you want to model, run the **uname** command in a shell on the device.

The compatibility options for the Zenoss modeling plugins are currently Linux or Darwin. The modeling command plugins can be extended for platforms not listed above; however, the procedure for creating new modeling command plugins is beyond the scope of this guide.

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

1. Go to the Add Device page.
2. Set discover to a value of None.
3. After adding the device, navigate to the device and view its zProperties tab.
4. If necessary, set zCommandUsername and zCommandPassword to the user name and password of the device (or set up authentication by using RSA/DSA keys.)
5. Click Edit next to zCollectorPlugins.
6. Click Add Fields for a complete list of command plugins.
7. Make sure zenoss.cmd.uname is positioned first on the left side.
8. Click the X next to plugins you wish to remove, and then drag remaining plugins to the left.
9. Remodel the device.

13.1. Using Device Class to Monitor Devices Using SSH

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

14. Modeling Devices Using PortScan

You can model IP services by doing a port scan. To model a device using a port scan:

1. Go to the zProperties tab of a device.

2. Change the zTransportPreference to “portscan."

3. Remodel the device.

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

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

15. Modeling Plugins

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

- DeviceMap – Collects basic information about a device, such as its OS type and hardware model.
- InterfaceMap – Collects the list of interfaces on a device.
- RouteMap – Collects the routing table from the device.
- IpServicesMap – Collects the IP services running on the device.
- FileSystemMap – Collects the list of file systems on a device.

15.1. Viewing Modeling Plugins for a Device

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

1. Navigate to the device.

2. From the page menu, select More, and then select Collector Plugins.

   The Collector Plugins tab appears, showing all of the collector plugins for the device.

16. Debugging the Modeling Process

You can run the Zenoss 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 collection maps Zenoss uses. For example, the following command runs only the interface plugin against the build.zenoss.loc device:

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

If the command returns any stack traces, Zenoss recommends that you forward the following details to Support for assistance:

- Command you ran
- Stack trace or stack traces returned
- Version of your Zenoss instance

17. Adding Custom Links to a Device Status Page

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

To add links:

1. Select a device from the device list.

   ### Note

   You also can create zLinks at the device class level to apply to all devices in the class.

2. From the device page menu, select More, and then select zProperties.
3. Scroll down to the zLinks zProperty.
4. Enter the link in the zLinks text field in the Value column.

   You can use any URL expressions including http, telnet, file, and mailto. You also can include other types of http markup, including image references. (For example, you might want to include performance graphs from other pages.)

5. Click Save.

   The zLink is saved and appears on the Status tab for the selected device.

18. Dumping and Loading Devices Using XML Lists

Zenoss allows you to export a list of your devices to an XML file 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 Zenoss instance (or reload them into the same instance), while in the Zenoss instance where you want the devices to be discovered, run the command:

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

Zenoss attempts to discover each of the devices in the XML file.

Chapter 10. Monitoring VMware Devices

1. Monitoring VMware Servers with Zenoss

Note

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

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

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

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

Note

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

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

1.1. ZenVMware ZenPack

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

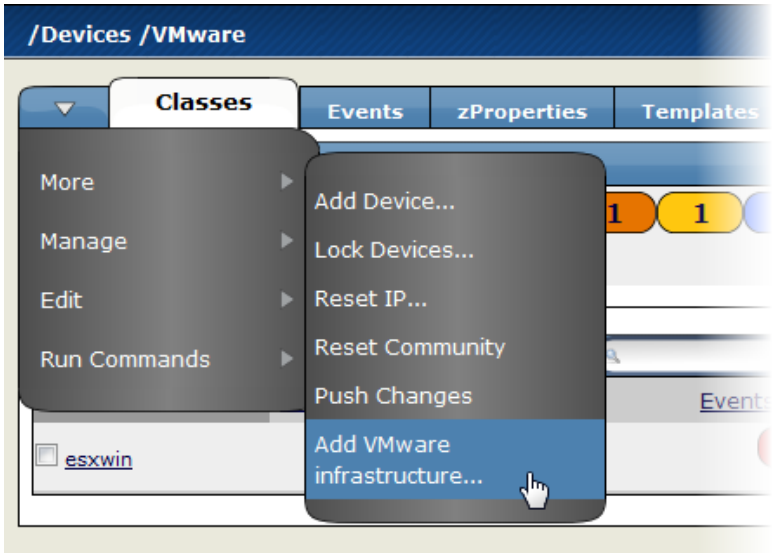
1.2. Setting Up VMware Monitoring

Follow these steps to begin monitoring your VMware servers.

1. From the Zenoss Navigation Menu, click Devices.

   The list of Devices appears.

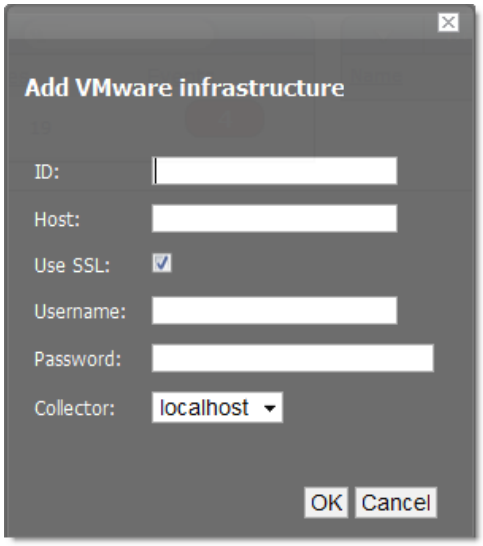
Figure 10.1. Add VMware Infrastructure



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

   The Add VMware infrastructure dialog appears.

Figure 10.2. Add VMware Infrastructure Dialog



4. Enter parameters to connect to the ESX server or vCenter Server that will provide monitoring capabilities.
   - ID - Enter a name for the infrastructure to be monitored.
   - Host - Enter the hostname of the server providing the VI SDK connections. This can be an individual ESX server or the location of a vCenter Server instance.
   - Use SSL - Select this option if the connection should be made by using SSL encryption.
   - Username - Enter the user name used to authenticate.
   - Password - Enter the password used to authenticate.

- Collector - Select the collector to use to retrieve information from the VI SDK endpoint.

5. Click OK.

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

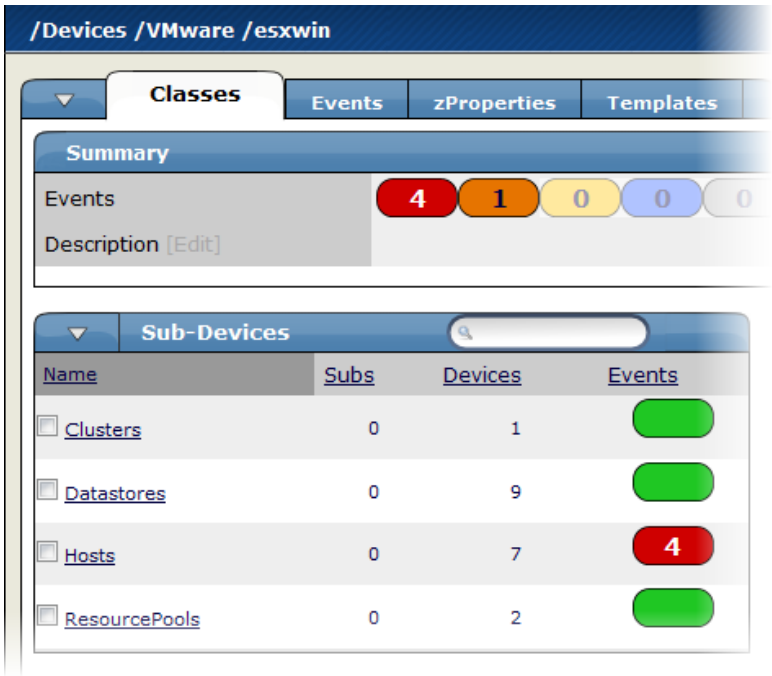
1.3. Viewing VMware Devices

Zenoss represents these VMware entities as devices:

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

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

Figure 10.3. VMware Device Classes



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

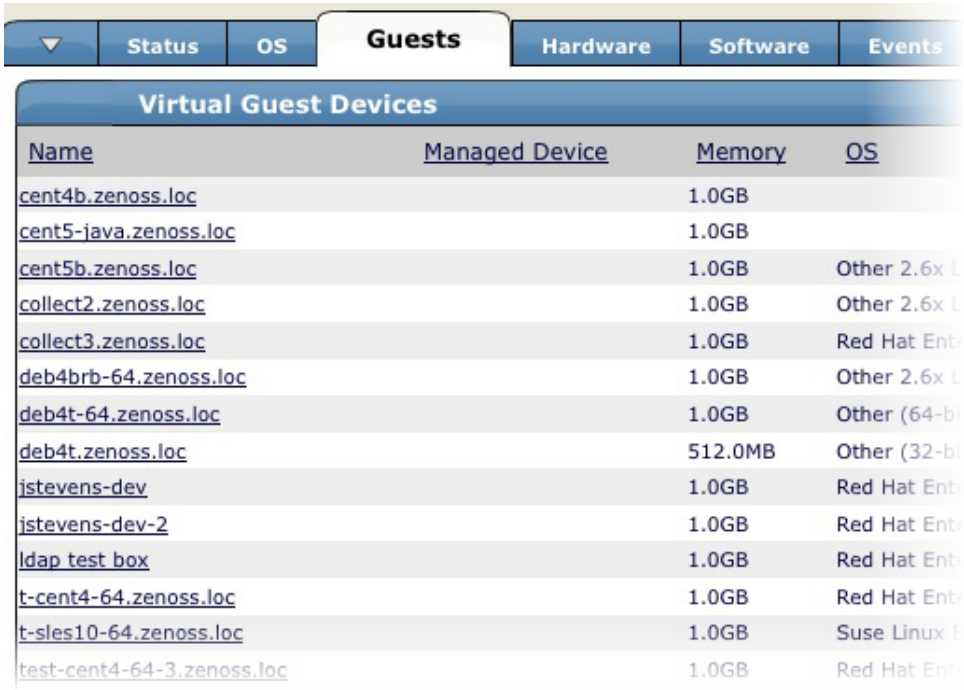
1.4. Viewing Guest Virtual Machines

To view guest VMs on an ESX server:

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

The Virtual Guests Devices list appears.

Figure 10.4. Virtual Guest Devices



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

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

Figure 10.5. Virtual Guest Devices - Managed Device

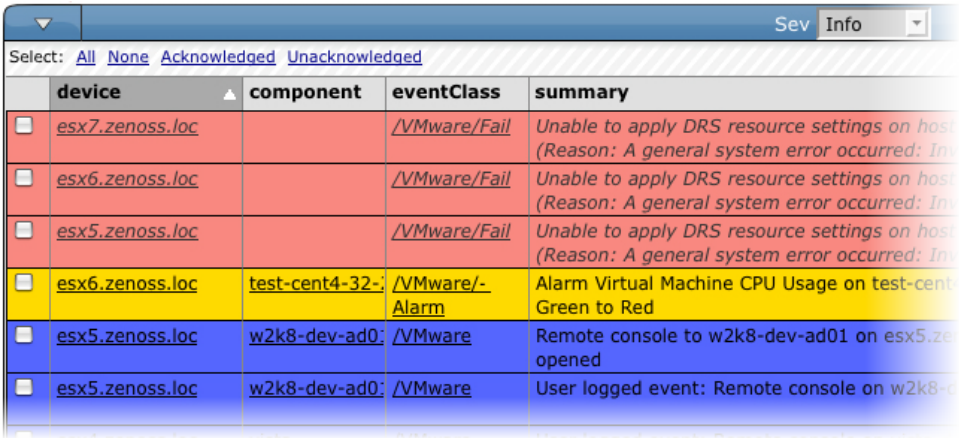
Click the Name link to go to the Guest component status page, which shows the VM's relationships to other VMware entities, and provides access to VMware-specific metrics and events.

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

1.5. VMware Events

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

Figure 10.6. VMware Events (Event Console)

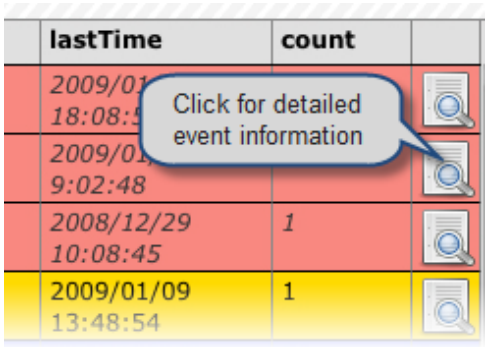


The device column shows the ID of the VMware entity with which the event is associated, unless the event is specific to a guest VM. In that case, the device column shows the ID of the host, and the component column displays the ID of the guest.

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

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

Figure 10.7. VMware Events (Event Console) - View Details



The VMware event type is the value shown for eventGroup.

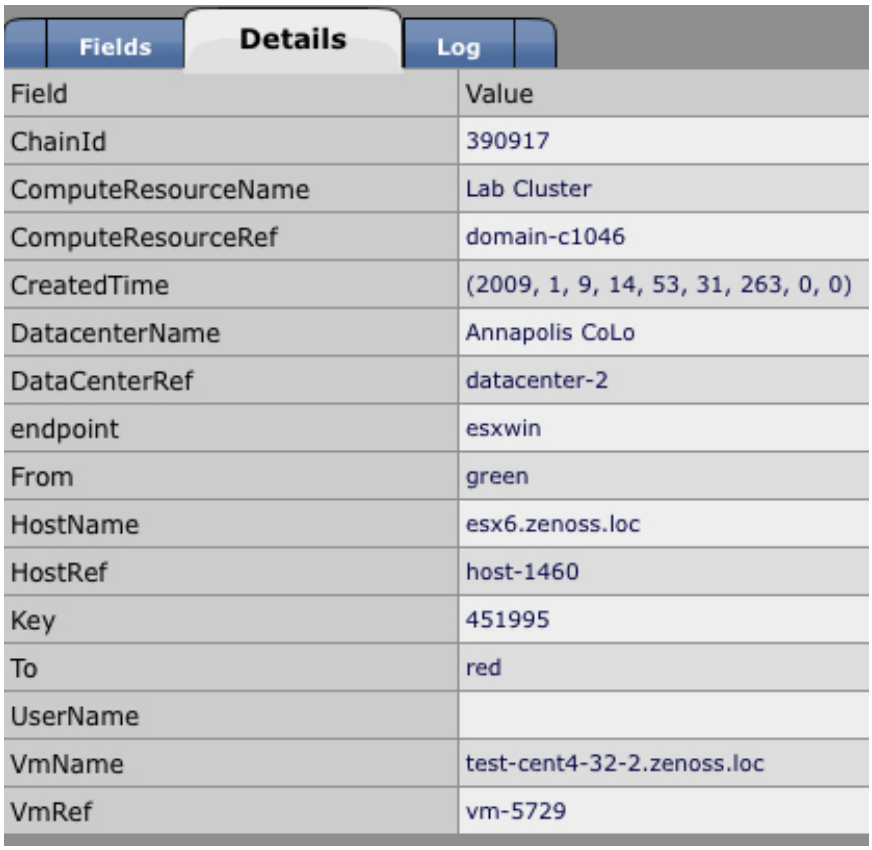
Figure 10.8. Event Details (Fields Tab)

Fields | Details | Log

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

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

Figure 10.9. Event Details (Details Tab)



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

1.6. Migration Events

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

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

1.7. Custom Data Sources

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

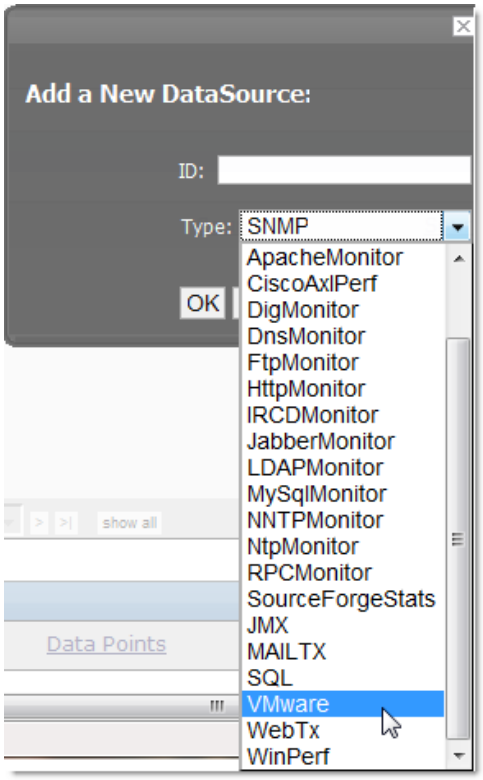
To create a custom data source:

1. Select the template to which you want to add the data source.

2. From the DataSources list of options, select Add Datasource.

   The Add DataSource dialog appears.

Figure 10.10. Add Data Source



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

     http://www.vmware.com/support/developer/vc-sdk/visdk25pubs/visdk25programmingguide.pdf
   - Instance - Certain metrics are further specified by an instance name. For example, the metric whose Group/Counter/Rollup Type triplet is Network/Network Data Receive Rate/average requires the name of the actual interface for full specification. In Zenoss, this metric is represented by the data source nicRx on the template VMwareNic. The VMwareNic template is bound to the individual host interfaces, each of whose ID is the interface name. In this case, the instance name is ${here/instanceId}.
5. Click Save to save the new data source.

Chapter 11. Event Monitoring

1. About Event Monitoring In Zenoss

The Zenoss Event Management system collects events from a variety of sources, such as:

- syslog
- Windows event log
- SNMP traps
- XML-RPC

Zenoss processes raw events to integrate them with the Zenoss model. As Zenoss processes an event, it applies a set of rules to determine the event's class. This provides additional information about the event, such as its severity or up-down correlation.

2. Understanding Zenoss Events

Read this section to learn more about events and their life cycle in Zenoss.

2.1. Life Cycle of a Zenoss Event

The Zenoss event life cycle begins with event creation. Events can be created:

- Outside the Zenoss system. Zenoss captures these events using specialized daemons. Examples include syslog and Windows events, and SNMP TRAPs.
- Within Zenoss. Internal events occur at discovery, or during availability or performance checks.

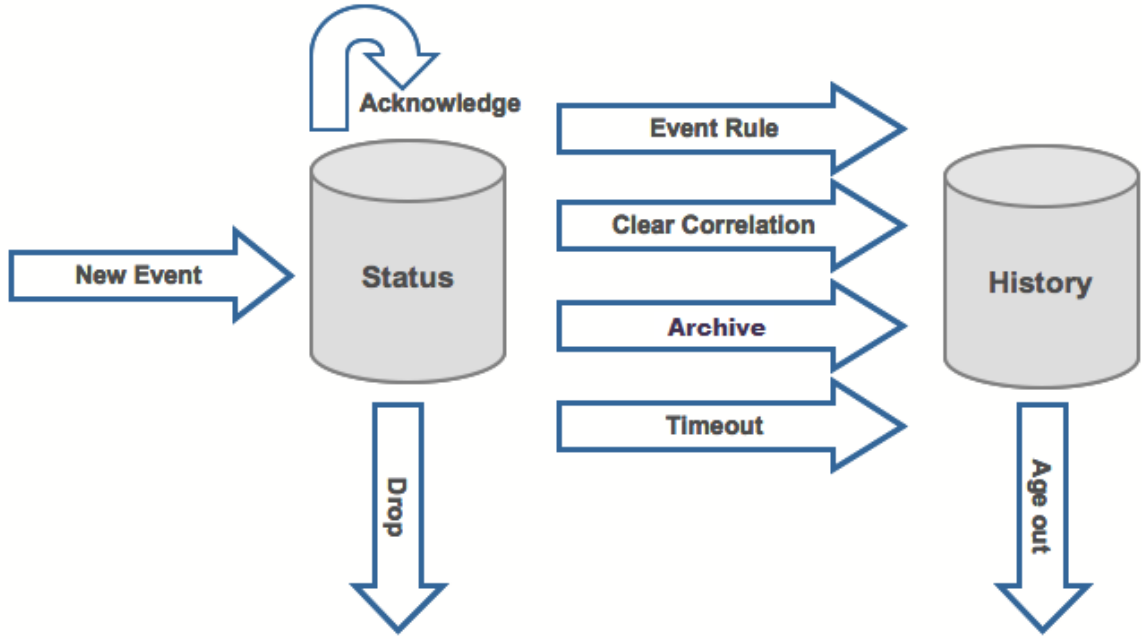
At creation, the event state is set to "New." Once created, the event can be acknowledged, suppressed, or "dropped" by application of an event class rule.

Eventually, the event can be archived into the event history database. It can be placed into the database:

- Manually
- Through auto clear correlation, meaning that a bad event is moved to history by creation of a resolution event
- By an event class rule
- Because of an inactive timeout

2.2. Zenoss Event Life Cycle

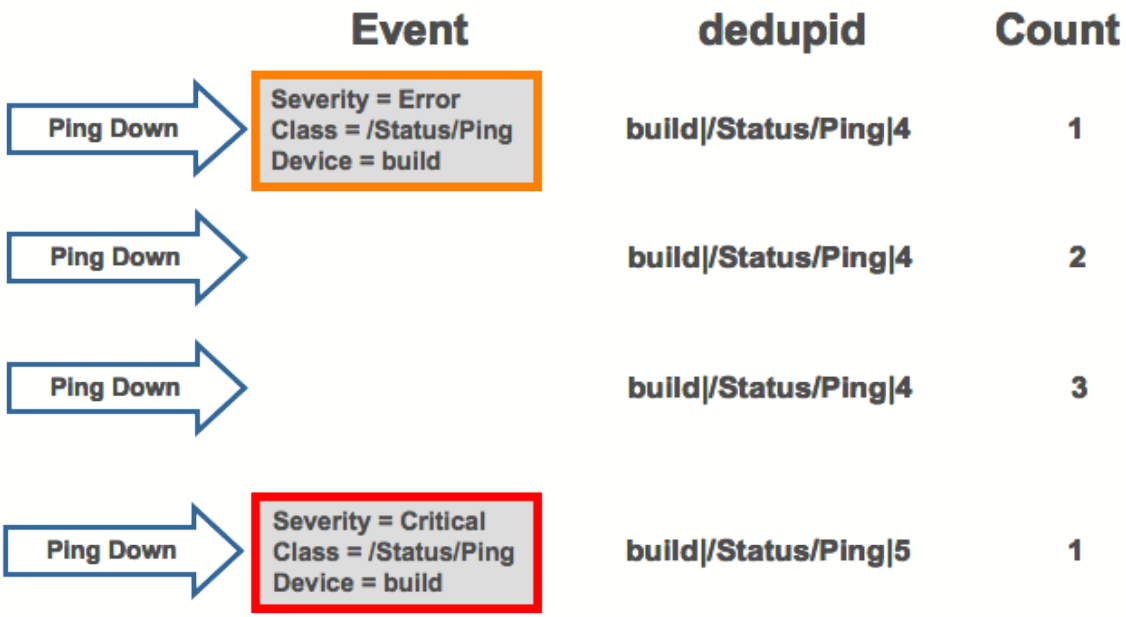
Figure 11.1. Zenoss Event Life Cycle



2.3. Suppressing Duplicate Events

Zenoss suppresses duplicate events to eliminate alert "chatter" that can occur from the time an event occurs and the time it takes to acknowledge or correct it. If a single event is submitted multiple times, Zenoss increments an event counter. Matching is done through an event class. Additional matching event submissions increment the counter but do not generate additional instances of the event in the event list.

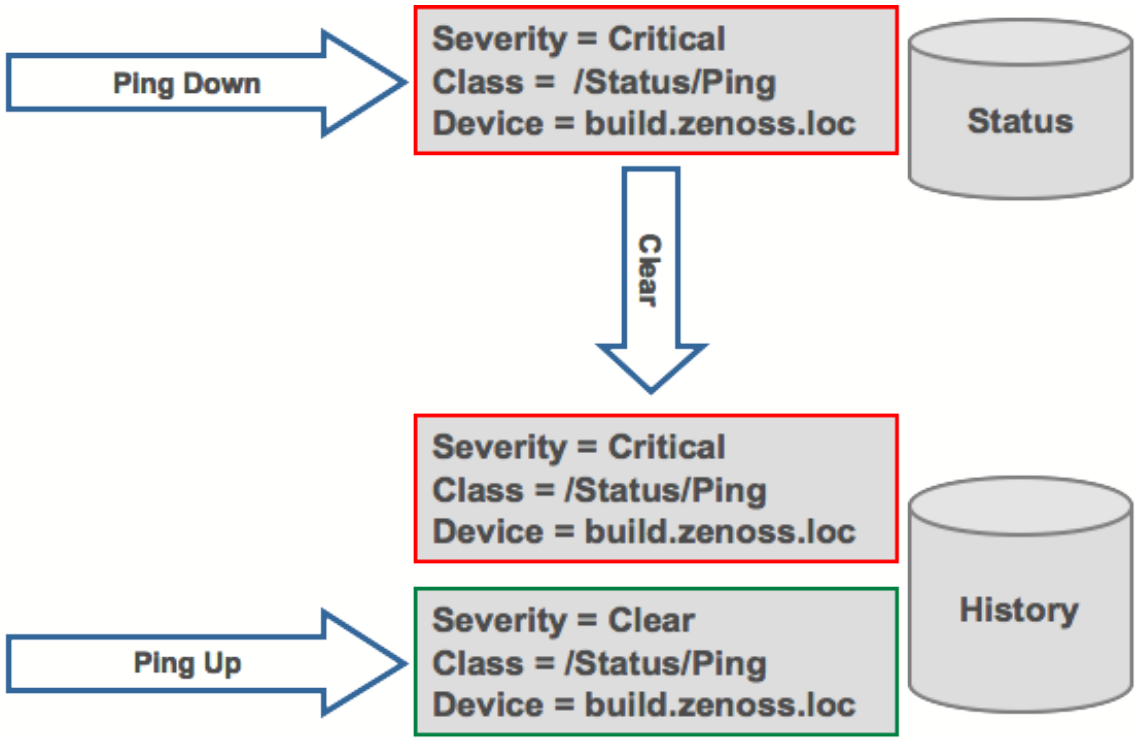
Figure 11.2. Duplicate Event Suppression



2.4. Begin-End Correlation

When an event starts, it enters the event life cycle; when it ends, it leaves the life cycle. If a positive event occurs that corresponds to a negative event already in the life cycle, then the negative event is cleared. Zenoss does this by matching several key data points in the events.

Figure 11.3. Begin-End Correlation

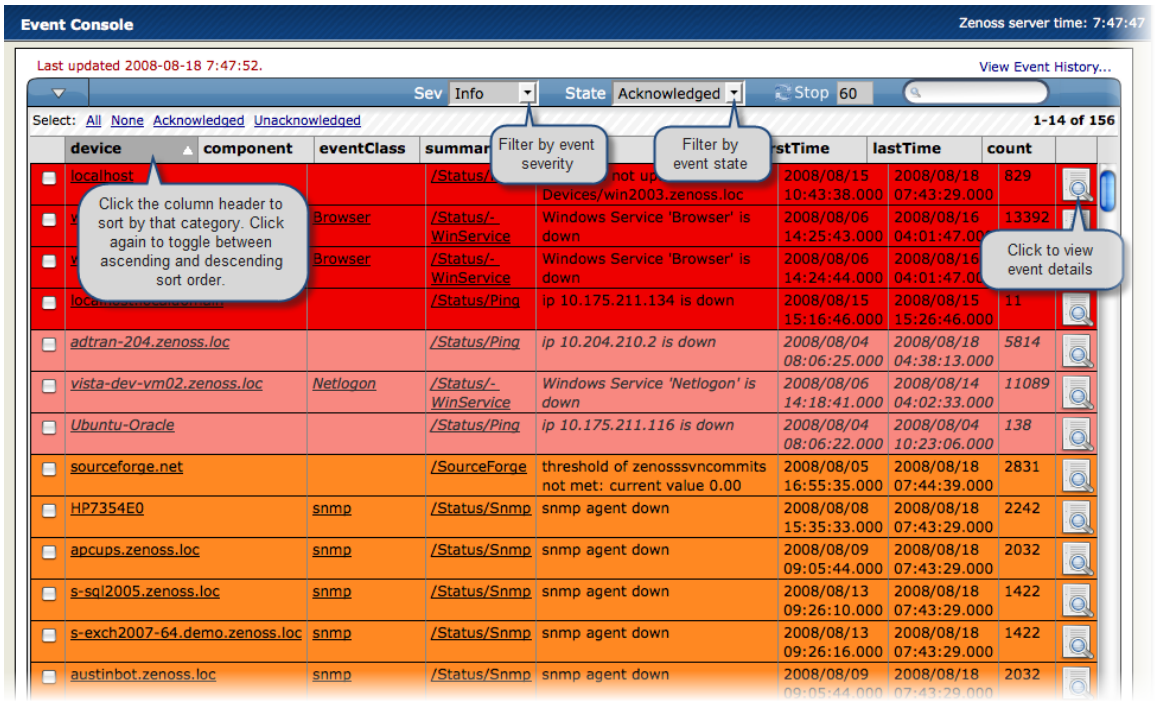


3. Zenoss Event Console

The event console is Zenoss' 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.

Figure 11.4. Zenoss Event Console



3.1. Sorting and Filtering Events

You can sort events that appear in the event console by:

- Device
- Component
- Event class
- Summary
- First time the event was detected (firstTime)
- Last time the event was detected (lastTime)
- Number of times the event was detected (Count)

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

You also can sort events by:

- Severity - Select to filter by Critical, Error, Warning, Info, Debug, or Clear.
- State - Select to filter by new, acknowledged, or suppressed events.

The Filter field lets you filter events. "Filter" is a regular expression on all text seen in the event list.

3.2. Viewing Event Details

You can view details for any event in the system. To view details, click the magnifying glass in the right-most column of an event row. The Event Detail window appears, showing the Fields tab.

Figure 11.5. Event Details

Fields | Details | Log

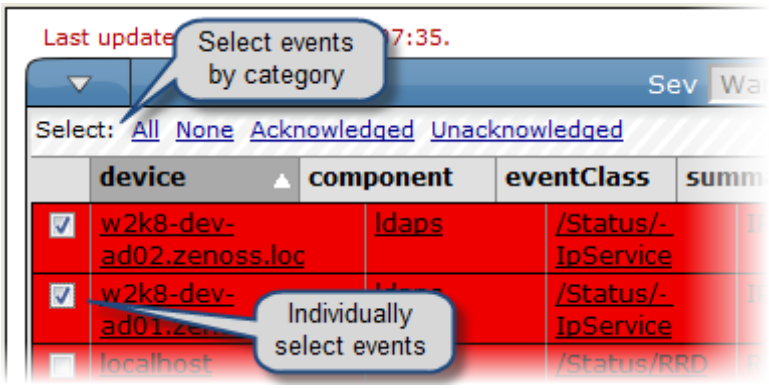
Field	Value
dedupid	localhost|rrd files too old
evid	A908AB295848a594cdfffedc4
device	localhost
component	
eventClass	/Status/RRD
eventKey	
summary	RRD files not updated: Devices/win2003.zenoss.loc/os/interfaces/VMwar
message	RRD files not updated: Devices/colo3560g.zenoss.loc/os/interf
severity	Critical (5)
eventState	New (0)
eventClassKey	
eventGroup	
stateChange	2008/08/18 07:48:29.000
firstTime	2008/08/15 10:43:38.000
lastTime	2008/08/18 07:48:29.000

3.3. Selecting Events

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

- Click the box next to one or more events
- Use the Selection area to select all, acknowledged, or unacknowledged events.

Figure 11.6. Selecting Events



3.4. Managing Events

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

- Acknowledge the event
- Move the event to history
- Map the event to a specific location

You can navigate to the component where an event was triggered. Click a component in the list to go to its Device Status page.

4. Creating Test Events

While most events are generated by the devices in your system, Zenoss also provides you with the ability to generate artificial "test" events about any device in the system. This feature is helpful when you are testing or trying a new setup.

To create a test event:

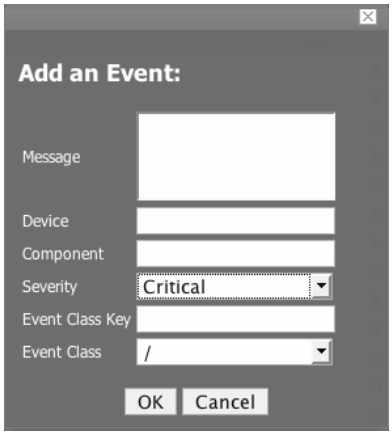
1. Click Events in the Navigation menu.

   The Events page appears.

2. Select Add Event.

   The Add Event dialog appears.

Figure 11.7. Add Event Dialog



3. Enter and select details about the test event and device you want to test, and then click OK.

   The event appears in the event console according to the criteria you set.

4.1. Sending Events from the Command Line

Use the zensendevent tool to send events to Zenoss from the command line. This Python script provides the following options.

Usage:

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

Options:

- -d, --device

  The local fully-qualified domain name

- -p, --component

  Component from which this event is sent.

- -s, --severity

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

- -c, --class

  Event class for this event

- --port

  xmlrpc server port. Default: 8081

- --server

  xmlrpc server. Default: localhost

- --auth

  xmlrpc server auth. Default: admin:zenoss

Example:

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

5. Event Classes

When an event enters the Zenoss event life cycle, Zenoss first integrates the received event by assigning it to an event class. Events are mapped to event classes by event class instances. Event class instances are looked up by a non-unique key (EventClassKey).

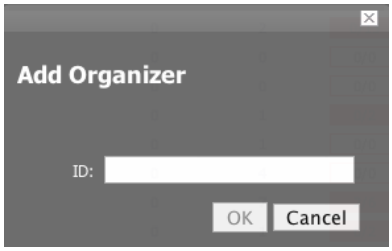
5.1. Creating an Event Class

Follow these steps to add an event class to Zenoss:

1. From the Navigation menu, click Events.
2. From the Classes tab, open the SubClasses table menu, and then select Add Organizer.

   The Add Organizer dialog appears.

   Figure 11.8. Add Organizer Dialog

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

   The Event Class appears in the SubClasses list.

6. Event Manager Settings

You can adjust settings for the Event Manager, including:

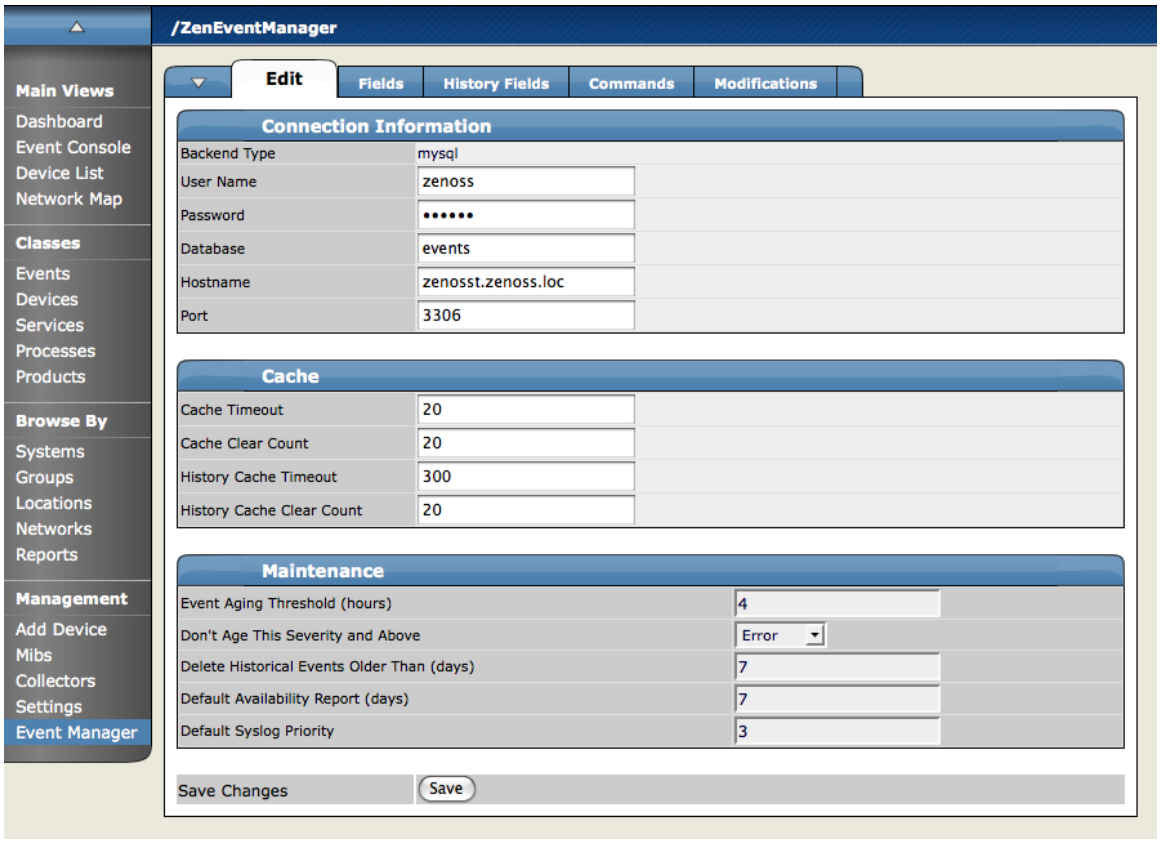
- MySQL event database connection
- Event cache timeouts and counts

- Maintenance settings

6.1. Accessing Event Manager Settings

To access event manager settings, select Event Manager from the Navigation menu. The Event Manager Settings tab appears.

Figure 11.9. Event Manager Edit Tab



6.2. Changing Event Database Connection Information

To edit event database connection settings, make changes to one or more fields in the Connection 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.

6.3. 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.
- Clear Cache 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.

6.4. 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 Zenoss 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 Zenoss availability and status.
- Default Syslog Priority - Specify the default severity level for an event to generate an entry into the syslog.

7. Acknowledging Events

Acknowledging an event tells Zenoss that you are aware the event has occurred, and that the problem is being fixed or under control. When you acknowledge an event, the Dashboard displays that status.

To acknowledge an event:

1. Navigate to the event console.
2. Click the box to the left of one or more events to select them.
3. From the Events page menu, select Acknowledge Events.

Note

The events you acknowledge are rendered in a lighter color to indicate the change. If you return to the dashboard, the events now appear with a status of Acknowledged.

8. Moving Events Manually To History

Instead of deleting events, you can manually send events to the event history to be archived.

Note

Events also can be automatically archived.

To send an event to the event history:

1. Navigate to the event console.
2. Click the box to the left of one or more events to select them.
3. From the Events table menu, select Move to History.

   A confirmation dialog appears.
4. Click OK to confirm and move the selected event or events to history.

9. Clearing Event History

By default, Event History is set to 0. This means that no events in history will be aged-out and deleted.

To change this default setting:

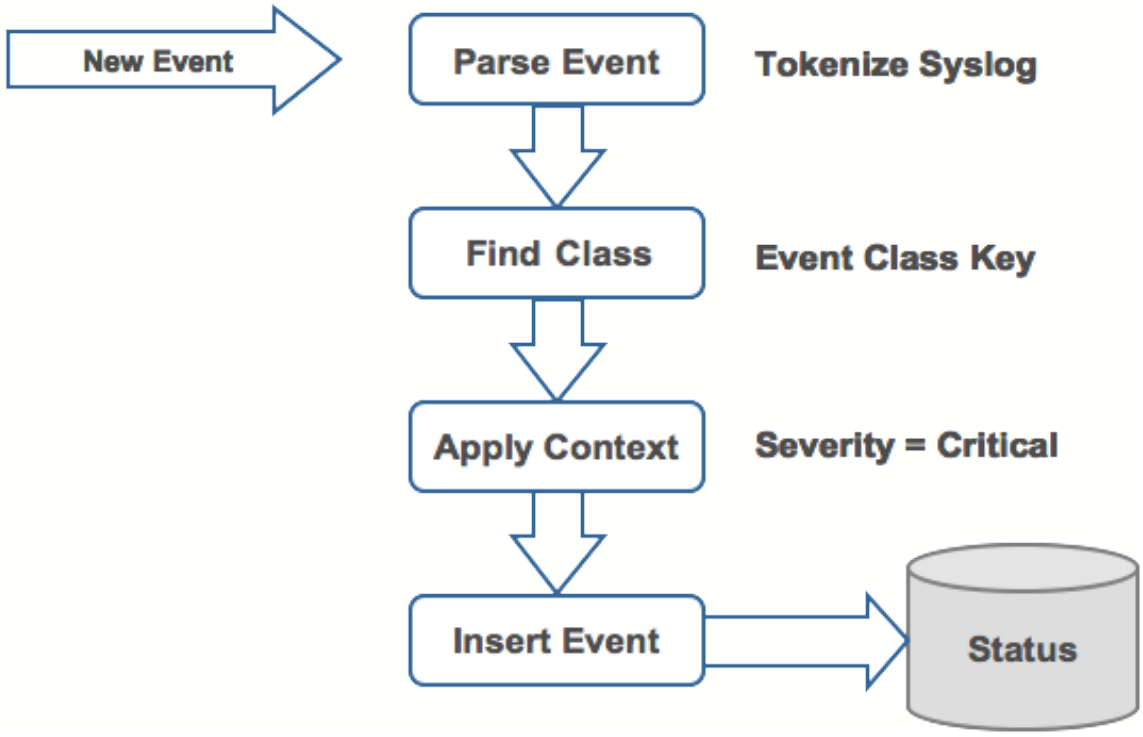
1. From the Navigation menu, select Event Manager.
2. In the Maintenance area, enter a value in the Delete Historical Events Older Than (days) field to set the number of days to wait before deleting items from history.

10. Event Class Mapping

Event class maps are the mechanism by which Zenoss integrates events into the system.

As shown in the following diagram, Zenoss parses an event as it comes into the system. Zenoss then assigns the event to the appropriate event class. Then, as the event class key is found and associated with the event, the context for that event class is applied to the incoming event. Finally, the event is updated based on its classification.

Figure 11.10. Event Class Mapping Process



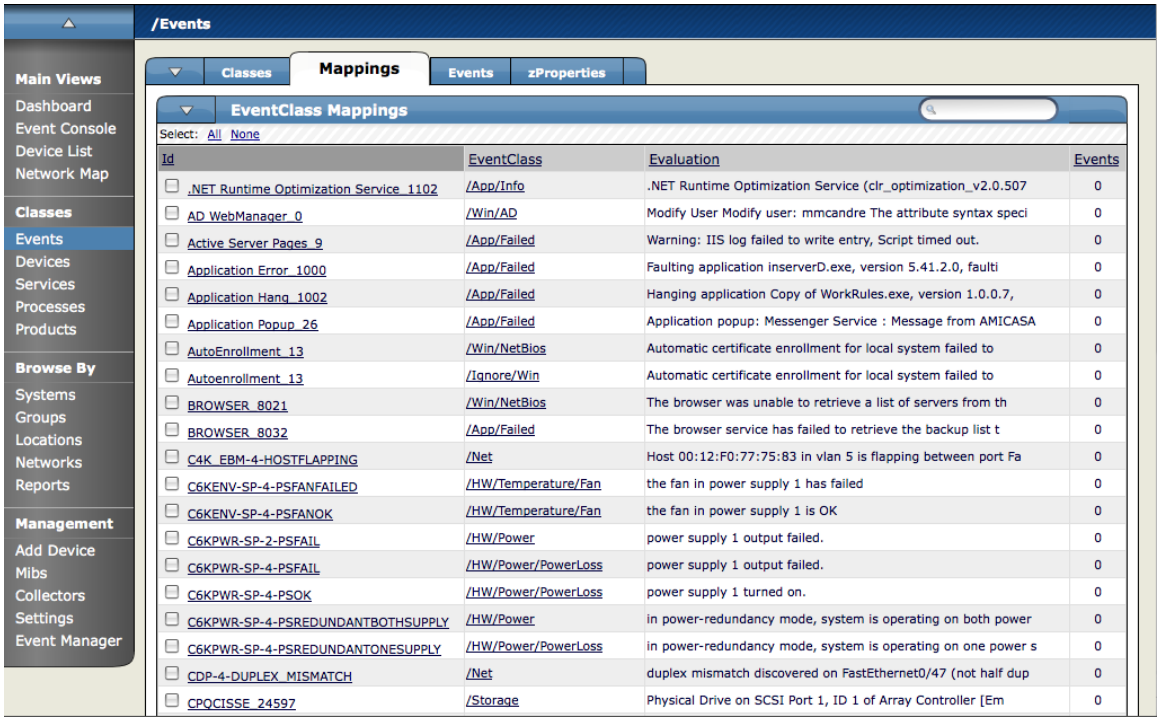
Creating and Updating Event Class Mappings

To create or update event class mappings:

1. From the Navigation menu, select Events.
2. Click the Mappings tab.

   The Mappings page appears.

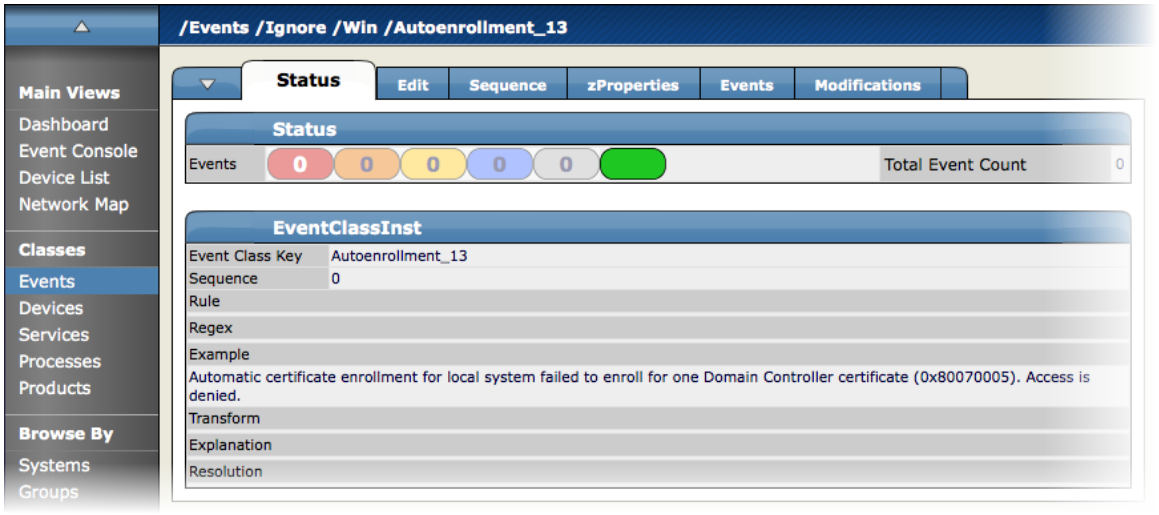
Figure 11.11. Event Mappings Page



3. Click the name of the event class mapping you want to edit.

   The page for the event mapping you selected appears.

Figure 11.12. Individual Event Mapping



4. Click the Edit tab.

   The Edit Event mapping tab appears.

Figure 11.13. Edit Event Mapping Tab

5. Edit these fields to define the event mapping:

   - Name - Enter a name for the event class map.

   - Event Class Key - Specify the key used to map incoming events to event classes. For syslog events, EventClassKey is most commonly the "Tag" or identifier of the syslog event. Often, the syslog tag maps to the process name from which the event came.

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

   - Sequence - If there is more than one match, then you can use this field to define sequencing priorities. The Sequence tab allows all instances for a particular EventClassKey to be ordered.

   - Rule - Enter a Python expression to match to the event. This expression will be evaluated with the variable name "evt" bound to the current event. (Refer to the appendix in this guide titled Event Database Dictionary for a detailed list of fields.)

     An example of a rule using these fields is:

     ```
     evt.priority>4
     ```

   - Regex - Enter regular expressions to match with events. When performing regular expression matches on an event, extraction directives can be used to populate attributes of the event. These directives follow the Python format for named extractions, in the form (?P<keyName>\S+).

For example, when creating a regular expression, the original event text can be added to the example field. When saved, the regular expression will be tested against this text.

- Transform - Enter one or more Python statements. This allows you to modify the event through manipulation of the EVT variable. This section uses TALES expressions. For more information about TALES expressions, see the appendix in this guide titled TALES Expressions.

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

- Resolution - Enter resolution instructions for clearing the event.

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

6. Click Save to save your changes.

11. Applying Event and Device Context Using Event zProperties

Once an event has been mapped to its class, two things happen:

- Event context is applied

- Device context or contexts are applied

EventClass context application is done by looking up the zProperty list zEventProperties.

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

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

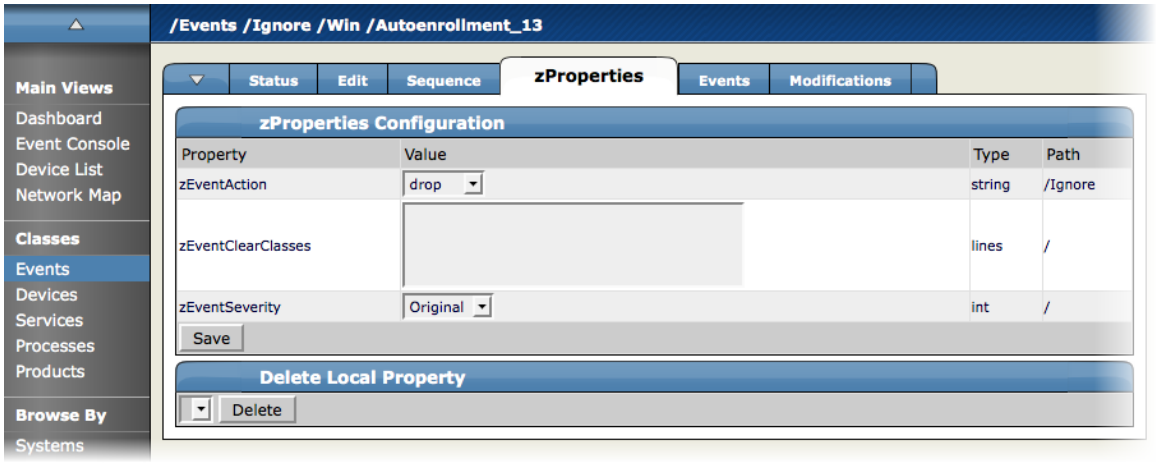
After the event context has been applied, then the device context is applied. During this process, the productionState, location, DeviceClass, DeviceGroups, and Systems are attached to the event in the event database. Once these properties have been associated with the event, Zenoss attempts to update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

To edit the Event zProperties:

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

   The zProperties tab for that mapping appears.

Figure 11.14. Event Mapping zProperties



2. Define the Event zProperties. Properties you can edit are:

   - zEventAction - Select a value that determines Zenoss' action when there is a match for this class. Available options are Drop, Status, and History.
   - zEventClearClasses - Enter matches that send clear events when they appear on this particular device.
   - zEventSeverity - Select the default severity for events of this class.

12. Mapping Events Through the Zenoss Interface

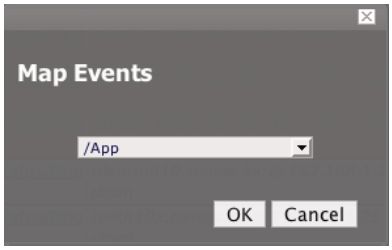
As events come into the system, Zenoss may not know how to classify them if the system has not previously encountered them. You can classify events and create event maps so that events will be managed the same way each time.

To map an event:

1. Navigate to the event console.
2. Select the event you want to map.
3. From the event console page menu, select Map Events to Class.

   The map events to class dialog appears.

Figure 11.15. Map Events Dialog



4. Select the event class to which you want to map this event.
5. Click OK.

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

13. Using Mappings to Correlate Events

You can use Zenoss to create multiple event class mappings that can correlate two events and create an action based on the occurrence of those two events. For example, you could create a correlation between the beginning and end of a failure, so that the events are cleared and then automatically moved to the event history database.

To correlate events:

1. Navigate to one of the event classes you want to correlate.
2. Click the Mappings tab and create an additional mapping.
3. Click the Classes tab again.
4. Click the mapping you just created, and then click its Edit tab.
5. Set Event Class Key back to the event class you want to correlate.
6. Optionally edit the event mapping fields.
7. Make a second mapping in the same class.

   Tip: Enter a name related to the first mapping to associate the two mappings.
8. Set the Event Class Key to the same event class key you set for the first mapping.
9. Go to the zProperties tab and set zEventSeverity to Clear (in the case where you want the second event to create a clear correlation). When this mapping matches, the event will become a clear event and will clear all active events of the same class.
10. Go to the Sequence tab. Note that all three mappings with the same event class key are listed here.
11. The order will control how the mappings are applied. The first mapping will apply to all events with the given event class key and will prevent the new rules from becoming active.
12. Put this mapping at the end of the sequence. Change its sequence number to 3 and then click Save.

14. Event Commands

Event commands allow users to define arbitrary shell commands to be executed when events are received by Zenoss that meet the criteria defined in the event command. Event commands are carried out by the zenactions daemon.

14.1. Commands Tab

The Commands tab shows all of the defined commands in Zenoss. Use the Define Commands table menu to add or delete commands.

14.2. Example: Create an Event Command

This example provides the framework for creating an event command.

1. From the Navigation menu, click Event Manager.
2. Go to the Commands page.
3. Create a new event command called writeFile.
4. Edit the command with the following values:
   - Set Enabled = True

- Default Command timeout = 60
- Delay = 0

5. Set the command as follows:

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

6. Set Clear command:

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

7. In the Where area, add a filter of Message contains "testing event commands."
8. Click Save.

14.3. Example: Test the Event Command

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

1. Start watching the output file with the command:

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

2. Create an artificial down event by using the Add Event functionality and defining it as follows:
   - Message = My Application is broken
   - Device = (any device in your system)
   - Severity = Critical
3. Look at the output file. After the next zenactions cycle you can see the down event.
4. Clear the down event by sending an up event:
   - Message = Now everything is OK
   - Device = (device used in the down event)
   - Severity = Info

15. 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, SNMP 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 Zenoss, and the system cannot identify the event (the event is classified as "/Unknown"), then you can map the event so that Zenoss handles the event consistently.

15.1. Mapping SNMP TRAPs

To map this event:

1. From the Event Console, select the unknown event.
2. From the page menu, select Map Events to Class.

   The Event Mapping dialog appears.

3. Select /App, and then click Map.

To edit this mapping:

1. From the Navigation area, select Events.
2. Click the Mappings tab.
3. Select the event map you created.
4. Click the Edit tab.

   The Event Mapper 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. Click the last column of the event to see these values as event details.

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:

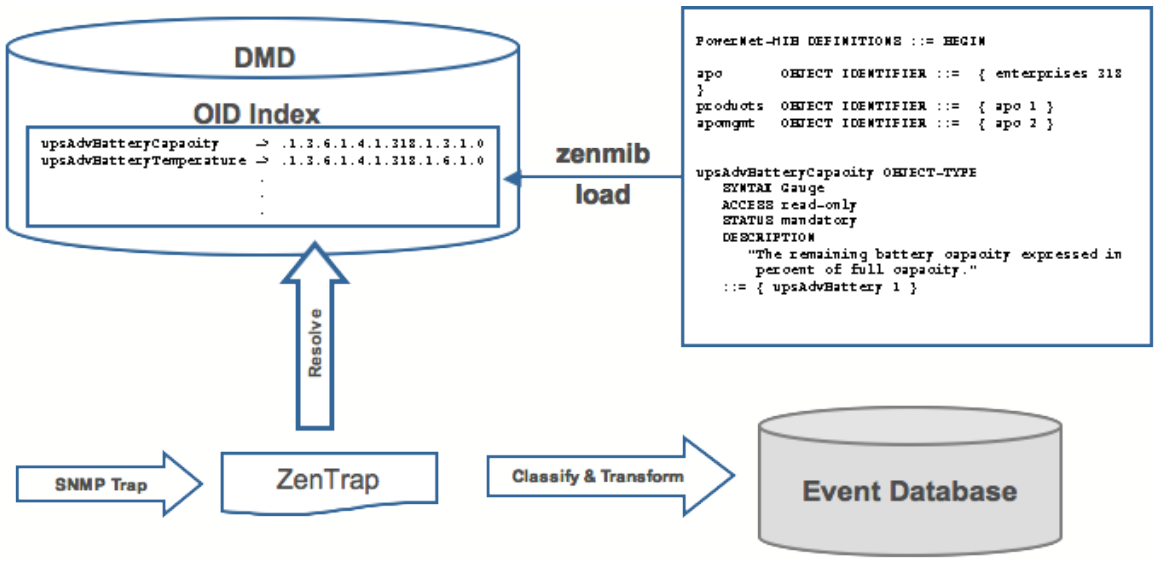
```
evt.summary = getattr(evt, ".1.3.6.1.4.9789.1500.2.5", "Unexpected missing OID")
```

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 " + devi
```

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

Figure 11.16. 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
```

15.2. 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 Zenoss 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 into Zenoss:

   ```
   $ 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-MI
   $ZENHOME/share/mibs/site
   ```

4. Run **zenmib** again and load the definitions into Zenoss:

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

5. Send the trap a second time:

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

6. 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
```

In the next section, titled "Transforming Events with Event Mappings," you will extract the "sysLocation" value (Device in Annapolis) for the summary. Doing this keeps users from having to drill down into the detail screen for this information.

15.3. 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. On the map, go to the Edit tab.
4. In the Transform area, update the event with detail data. The entry field allows you to insert arbitrary 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 all 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 have any problems with the Transform, check the zentrap.log file for errors that occurred.

15.4. Event Transforms Based on Event Class

When an event arrives in the Zenoss 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, both "evt" and "device" objects are available within the script of the transform.

16. 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:

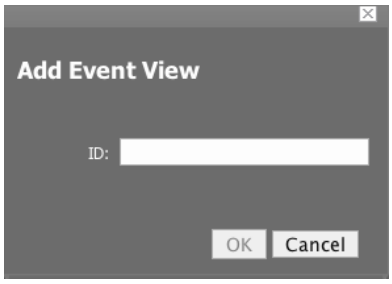
1. Click the Preferences link at the top right of the dashboard.
2. Click the Event Views tab.

The Event Views tab appears.

3. From the Event View table menu, select Add Event View.

   The Add Event View dialog appears.

Figure 11.17. Add Event View Dialog



4. In the ID field, enter the name for the event view.
5. Click OK.

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

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

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

7. Click the Edit tab.

   The Edit Event views tab appears.

Figure 11.18. Custom Event View (Edit Tab)

8. 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.

9. Click Save.

You can click the View tab to see the results of the custom event view.

17. Capturing Email Messages as Zenoss Events

ZenMail and ZenPop allow you to capture email messages as events in Zenoss. 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.

17.1. ZenMail

ZenMail serves as an SMTP server that you can bind to a specific TCP port. You can then configure your embedded system to send mail to the Zenoss server explicitly by using the Zenoss 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).

17.2. 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.

17.3. 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 Zenoss associates the event with the device with the same IP address. If the FROM field is a fully qualified domain name, then Zenoss 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 - Zenoss 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. Zenoss ignores secondary message bodies (typically HTML-encoded versions of the message). It also ignores attachments (such as files).

Chapter 12. Availability Monitoring

1. Monitoring Topology with ZenPing

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

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

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

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

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

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

1.1. Controlling the Ping Cycle Time

Follow these steps to set up the ping cycle time.

1. From the left Navigation menu, select Monitors and select the Status monitor localhost and click the localhost link.
2. Notice the list of machines being pinged by this monitor.
3. In the "Edit" tab, change "Cycle Interval" to the desired interval.
4. On the next configuration cycle, the ping monitor will ping at the interval you set.

1.2. Using the Predefined /Ping Device Class

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

2. Monitoring TCP Services

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

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

2.1. Zenstatus

ZenStatus performs monitoring of TCP services. It is configured by turning on monitoring of a service under the "Services" root on the Navigation Toolbar. Service monitoring can be turned on a service class but this can be

overridden on any service instance. For example, “SMTP” will be monitored by default but it may not be a critical service on all boxes. If this is the case, it may be removed on specific devices. Also, if the service is configured to only listen on localhost (127.0.0.1) the service will not be monitored.

2.2. Adding a Service To Monitor

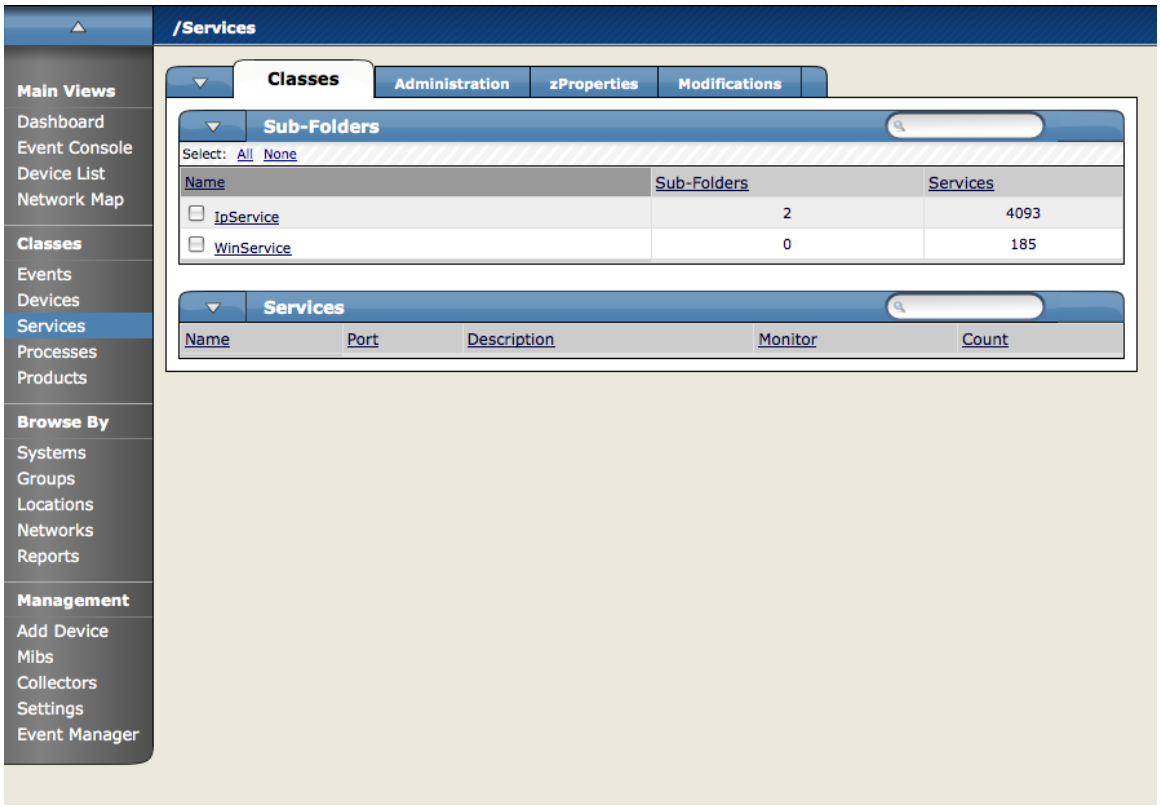
To add a service to monitor:

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

2. Click the Add button.

   The service appears in the Services list.

Figure 12.1. Services List - Classes tab



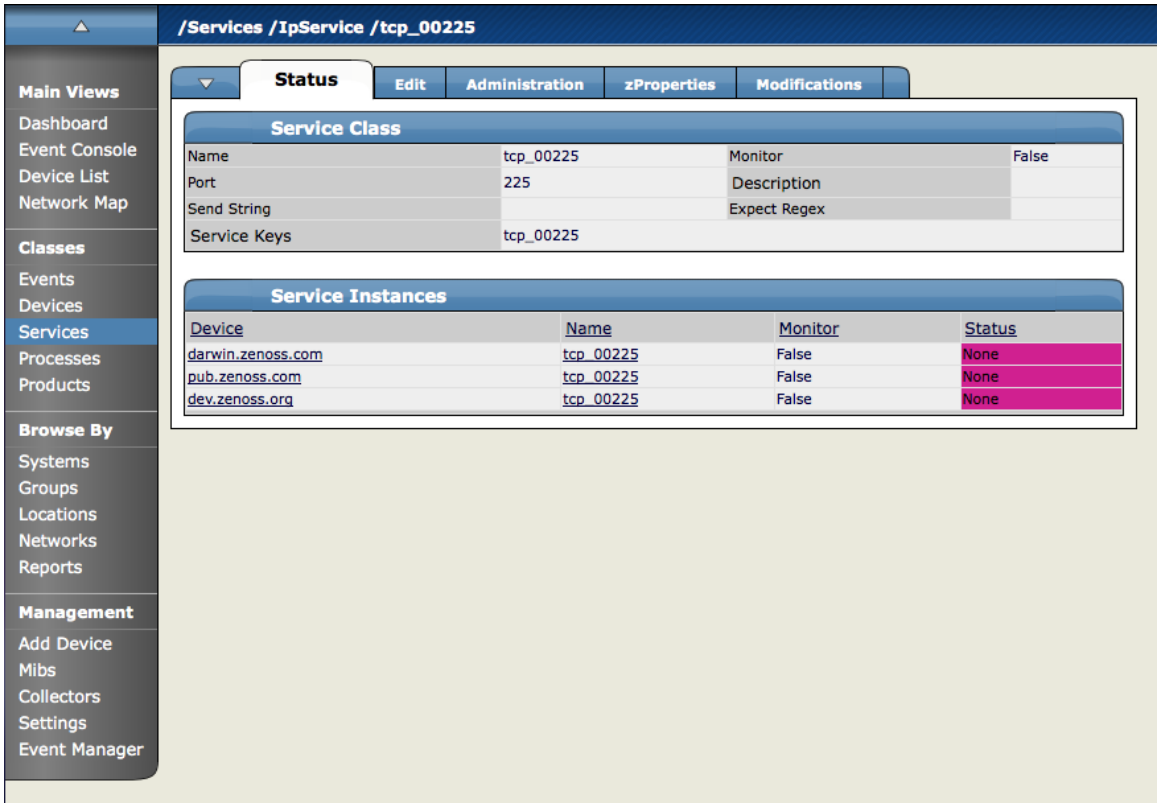
3. To set monitoring to True, click the Edit tab and set the Monitor pop-up to True. The service is now being monitored.

2.3. Monitoring Status Service Status Information

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

The Individual Service Status tab appears.

Figure 12.2. Individual Service Status Tab

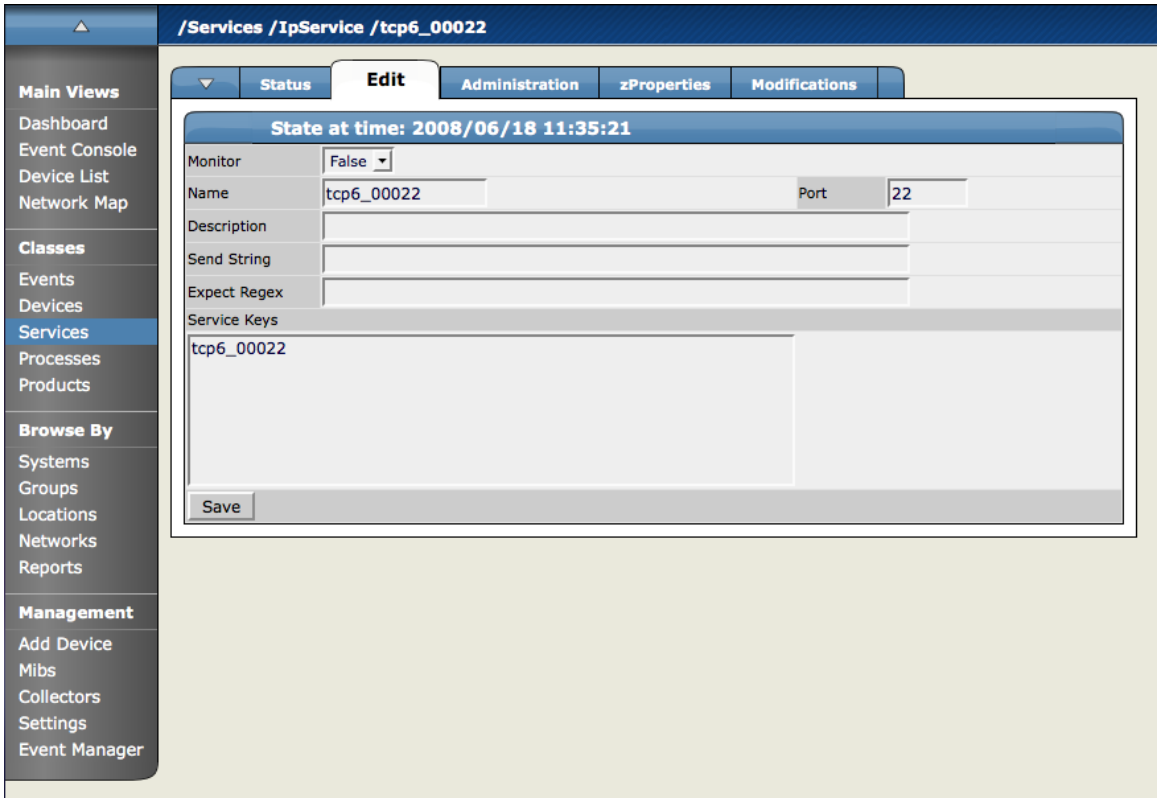


This tab shows you a summary of the details associated with this service. You can see the name of the service, whether its monitored or not, a Description, any associated Service Keys and a List of Devices where the service is currently running. To change any of this information, click the Edit tab.

2.4. Editing Service Information

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

Figure 12.3. Individual Service- Edit Tab

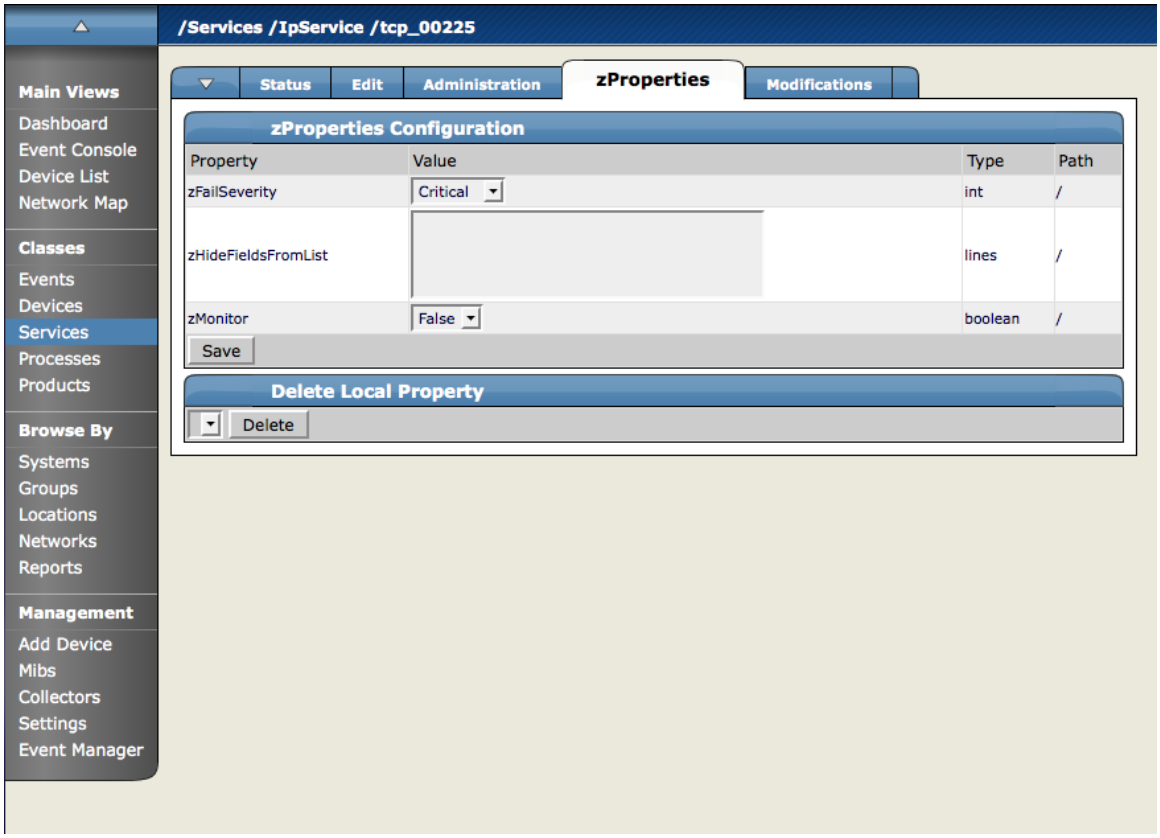


From this screen, use the Monitor pop-up to select True to monitor the service and False to not monitor the service. You can also add any associated Service Keys or enter a brief Description.

2.5. Configuring Service zProperties

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

Figure 12.4. Individual Service - zProperties Tab



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

- ZFailSeverity
- ZHideFieldsFromList
- zMonitor

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

2.6. Using the Predefined /Server/Scan Device Class

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

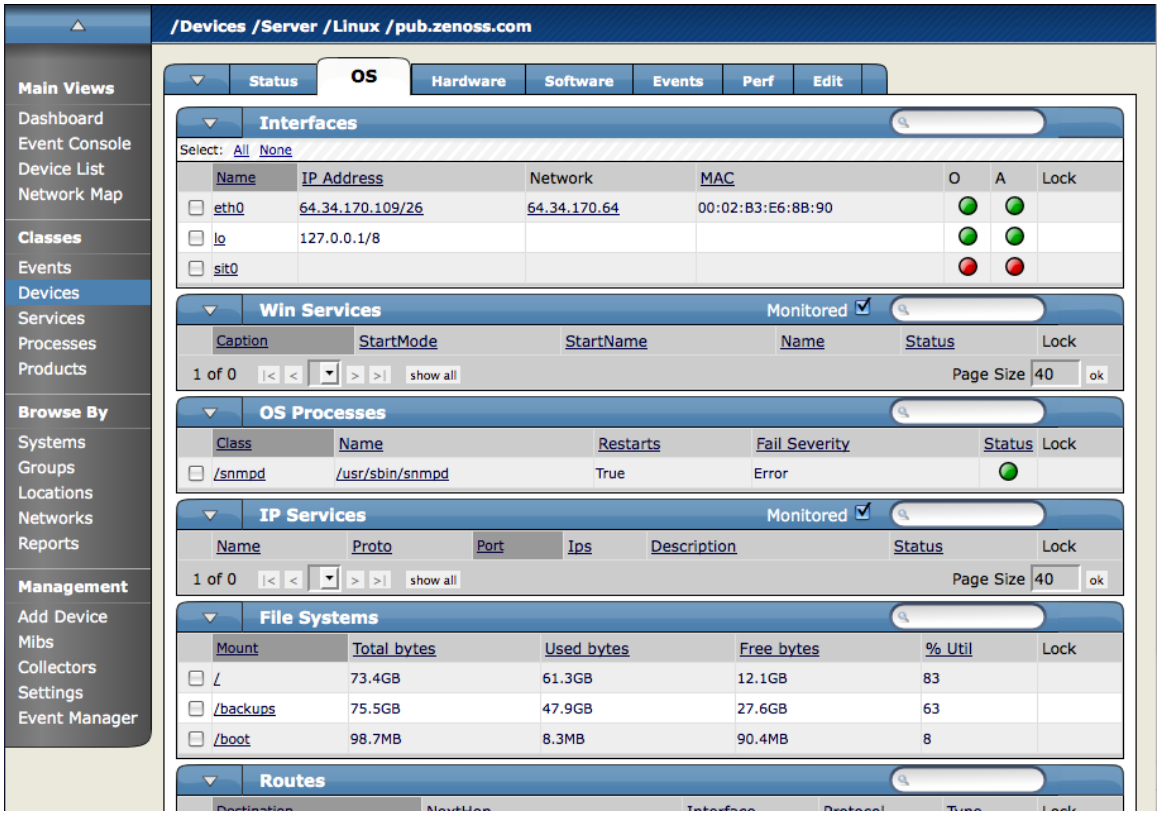
2.7. Monitoring a Service Using a Service Class

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

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

   The OS Tab for a device appears.

Figure 12.5. Showing Processes to Monitor



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

   The service summary for the service you have selected appears.

Figure 12.6. Service Summary

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

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

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

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

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

6. Click Save.

7. Click the Status tab again.

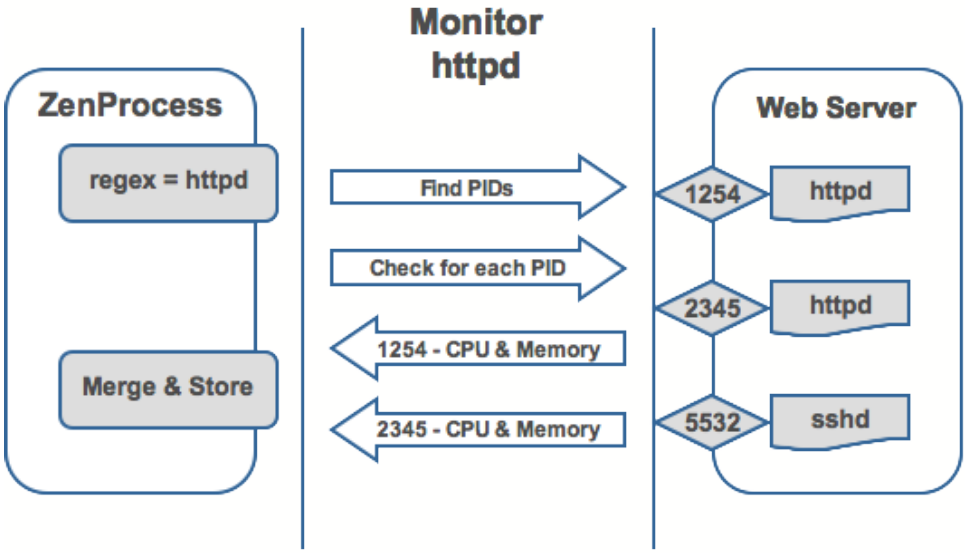
   Most of the instances of the Service will now be set to green, meaning they are monitored and up. The ones that remain unmonitored indicate that they have this service class set to not monitor at a local level.

3. Monitoring Processes

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

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

Figure 12.7. Process Monitoring



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

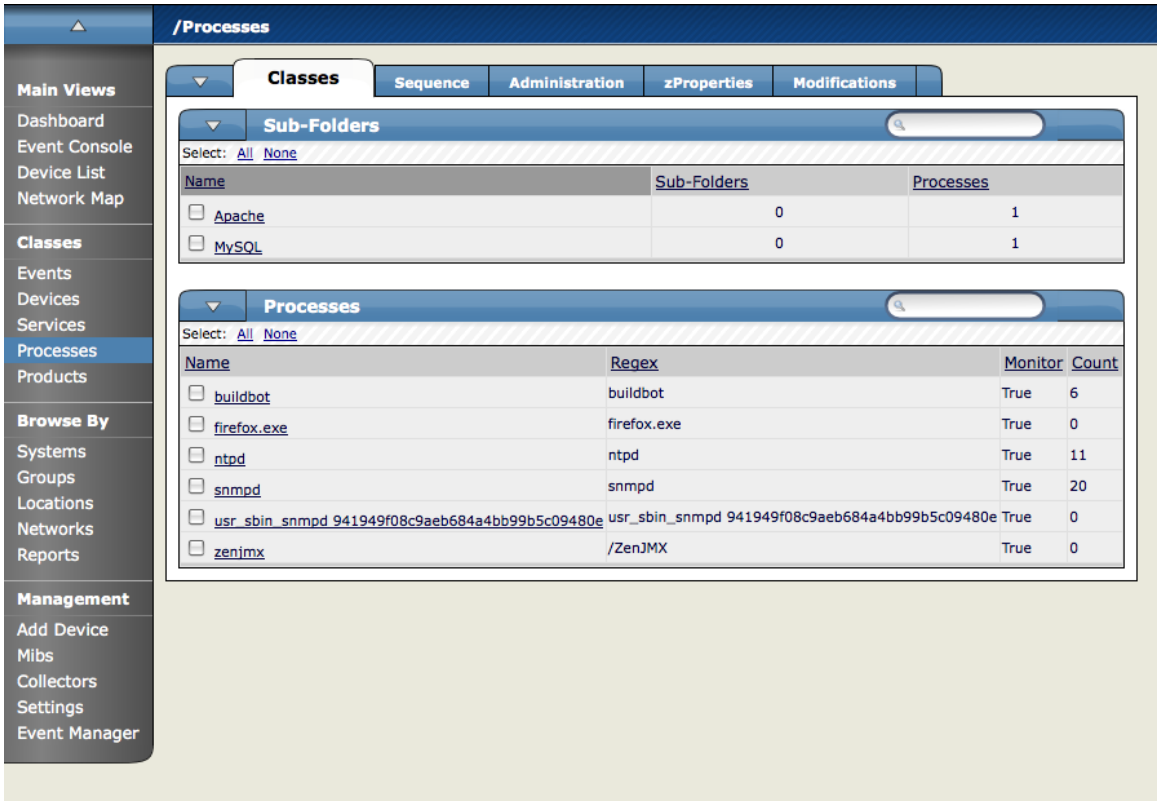
3.1. Adding Processes to Monitor

To add a process to monitor:

1. From the left navigation menu, select Processes.

   The Processes page appears.

Figure 12.8. Processes Page



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

   The Add OS Process Dialog Appears.

Figure 12.9. Add OS Process Dialog

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

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

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

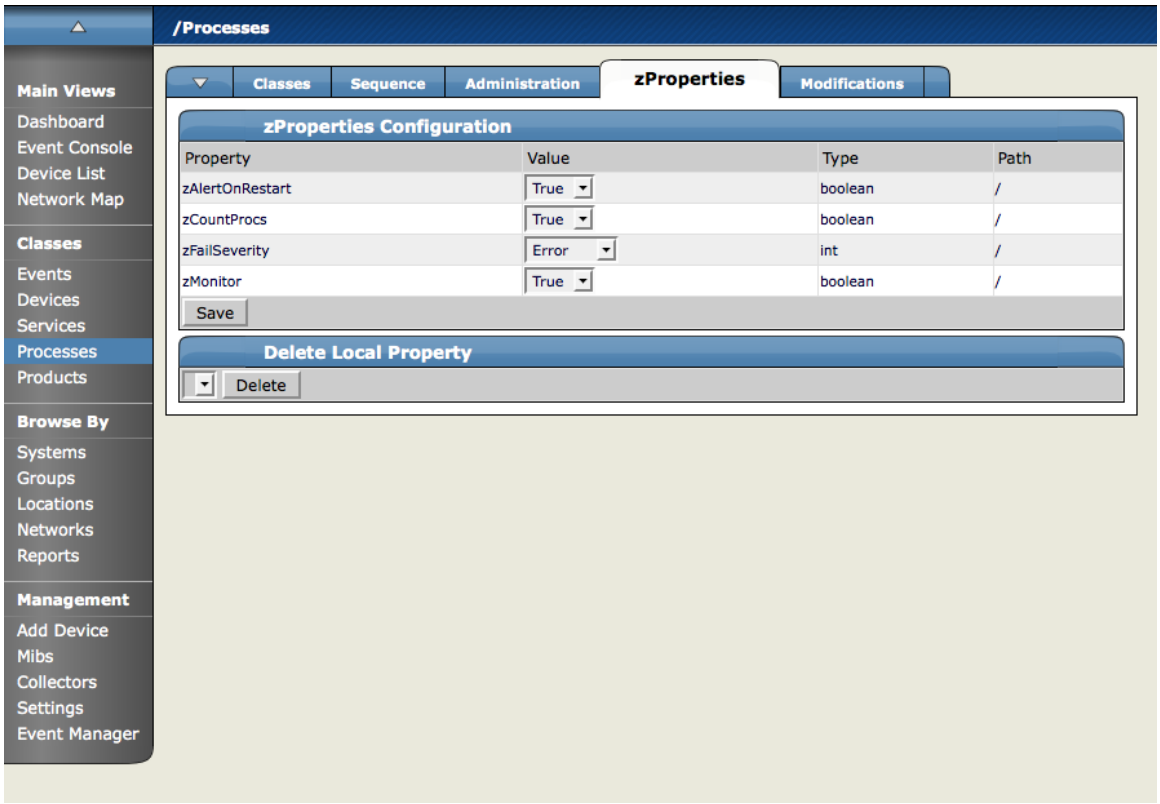
   Clicking on a specific process will take you to an interface that shows all instances of that process running across machines that have it monitored. If the process has multiple instances the Zenoss will monitor the sum

of CPU and memory utilization of all processes as well as the count of total processes running. However, if the process has only a single instance, CPU utilization and memory usage are graphed for the single process. To perform process monitoring the device SNMP agent must have a reasonable HOST-RESOURCES MIB.

3.2. Configuring Process zProperties

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

Figure 12.10. Processes zProperties tab



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

- ZAlertOnRestart
- ZCountProcs
- ZFailSeverity
- zMonitor

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

Chapter 13. Performance Monitoring

1. Performance Monitoring

Zenoss provides several methods for monitoring performance metrics of devices and device components. These include:

- ZenPerfSNMP - Collects data through SNMP from any device correctly configured for SNMP monitoring.
- ZenWinPerf (Zenoss Professional and 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, Zenoss stores performance monitoring configuration information in *performance templates*.

2. Performance Templates

Performance templates define how to collect performance data for devices and device components. You can define performance templates for device classes and for 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 in Zenoss if the data does not match those bounds.
- Graph Definitions - Describe how to graph the collected data on the device or device components.

2.1. Templates Page

The Templates page lists all of the Templates available to a device or device class.

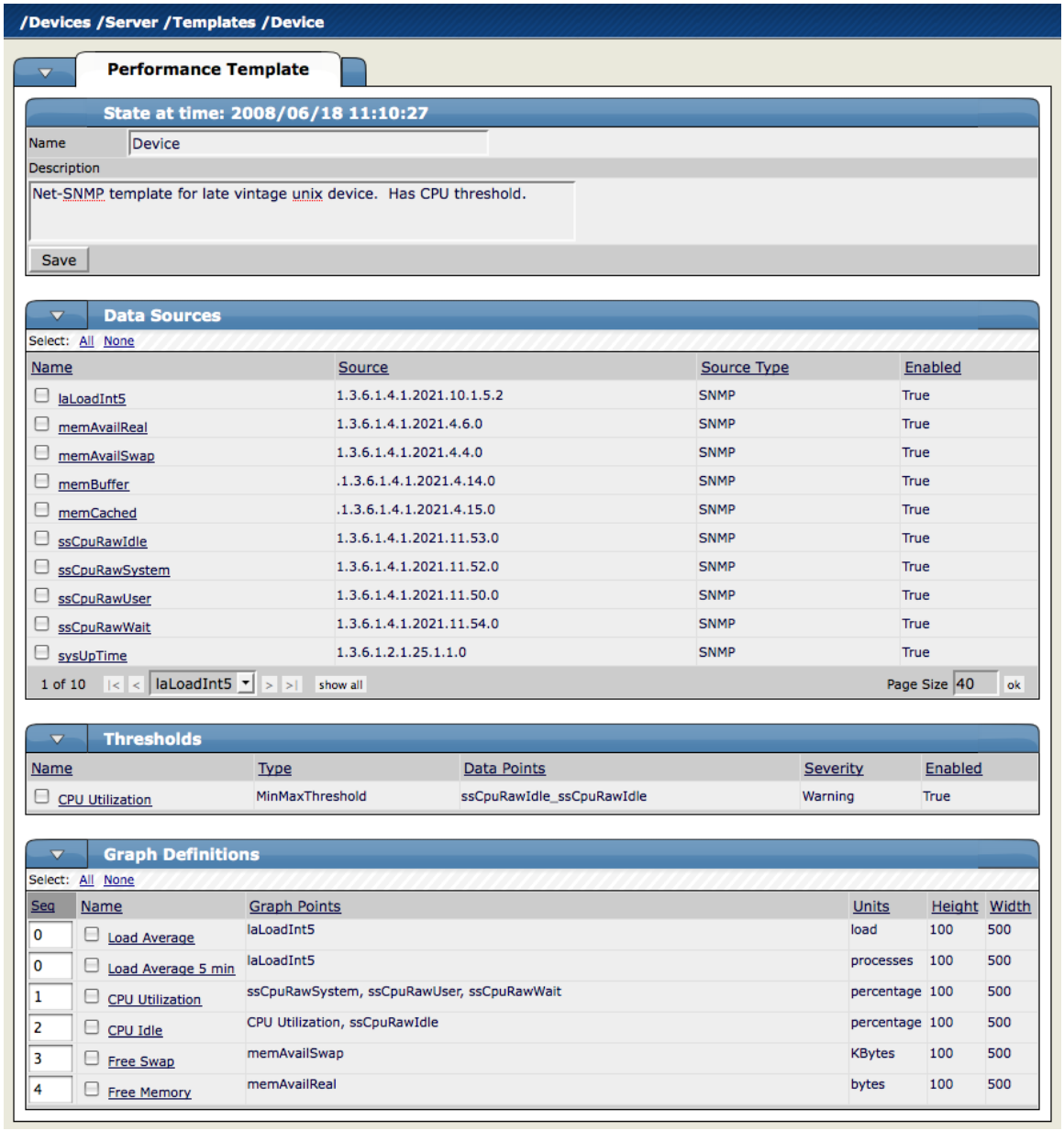
Viewing Performance Templates

To view available performance templates for a device, select More > All Templates from the Devices page menu. To view available performance templates for device classes, click the Templates tab from the Devices area.

The Available Performance Templates page shows the performance templates that are defined for a particular device or device class, and for those defined further up the device class hierarchy. If more than one template of the same name is defined, then only the one to which this device or device class can bind appears in the list.

Click a template in the list to view details about defined data sources, thresholds, and to see graph definition details.

Figure 13.1. Performance Template for Load Average Graph



3. Template Binding

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

First, Zenoss 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 zProperty.

After defining the list, Zenoss locates templates that match the names on the list. For each name, Zenoss 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.

Viewing Templates Available for Binding

To see which templates are available for binding, view the Templates page for any device or device class. This page shows all templates that are defined at this point or higher in the device hierarchy.

Changing Templates Available for Binding

To change which templates are currently bound:

1. Select Bind Templates from the Available Performance Templates table menu.

   The Bind Performance Templates dialog appears.

2. Select a performance template (Ctrl-click to select more than one), and then click OK.

   The selected performance template appears in the list of available templates.

Note

Alternatively, you can edit the zDeviceTemplates zProperty (from the zProperties page) to change which templates are bound. You cannot edit the bound name for a device component.

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

4. Data Sources

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

About SNMP Data Sources

SNMP data sources define data to be collected via SNMP by the ZenPerfSNMP daemon. They contain one additional field to specify which SNMP OID to collect. (Many OIDs must end in .0 to work correctly.) Because SNMP data sources specify only one performance metric, they contain a single data point. For more information, see the section titled SNMP Monitoring.

About COMMAND Data Sources

COMMAND data sources specify data to be collected by a shell command that is executed on the Zenoss server or on a monitored device, via telnet or ssh. 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.

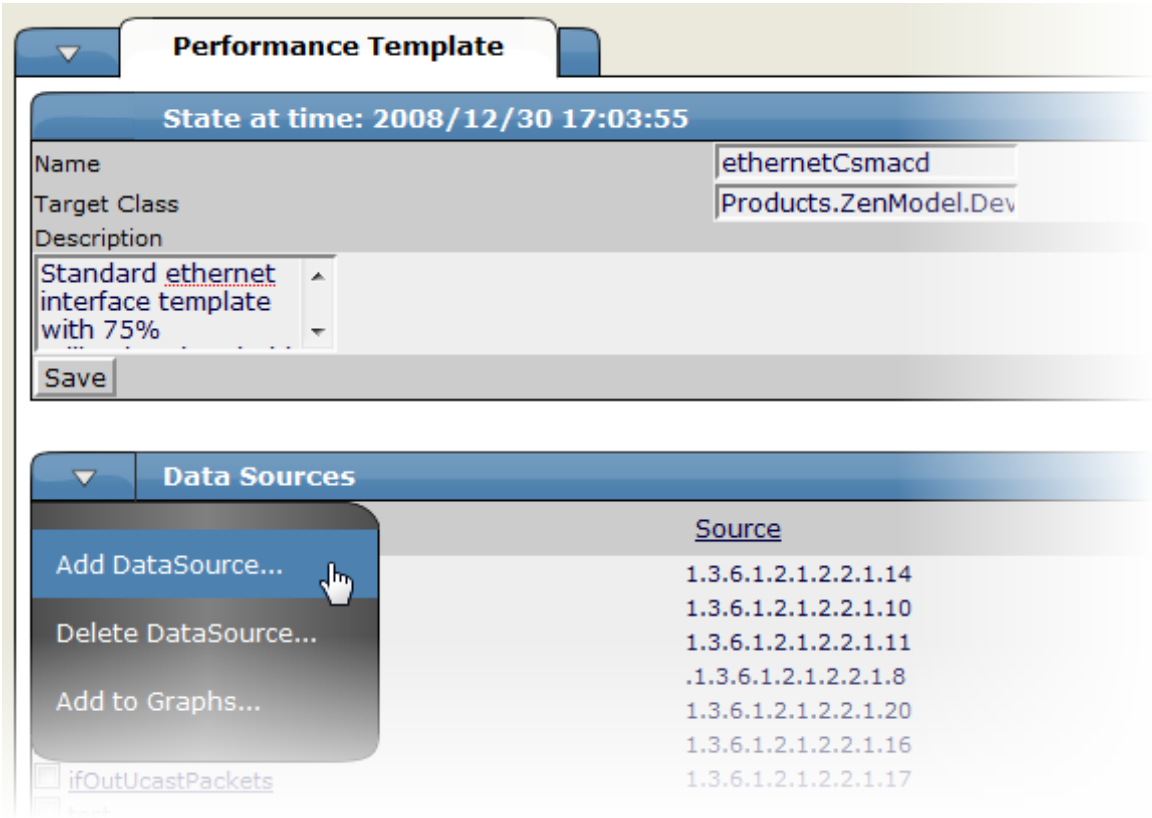
Shell commands used with COMMAND data sources must return data that conforms to the Nagio plug-in output specification. For more information, see the section titled Monitoring Using ZenCommand.

4.1. Adding a Data Source

To add a data source to a performance template:

1. From the Performance Template page, select Add DataSource from the Data Sources table menu.

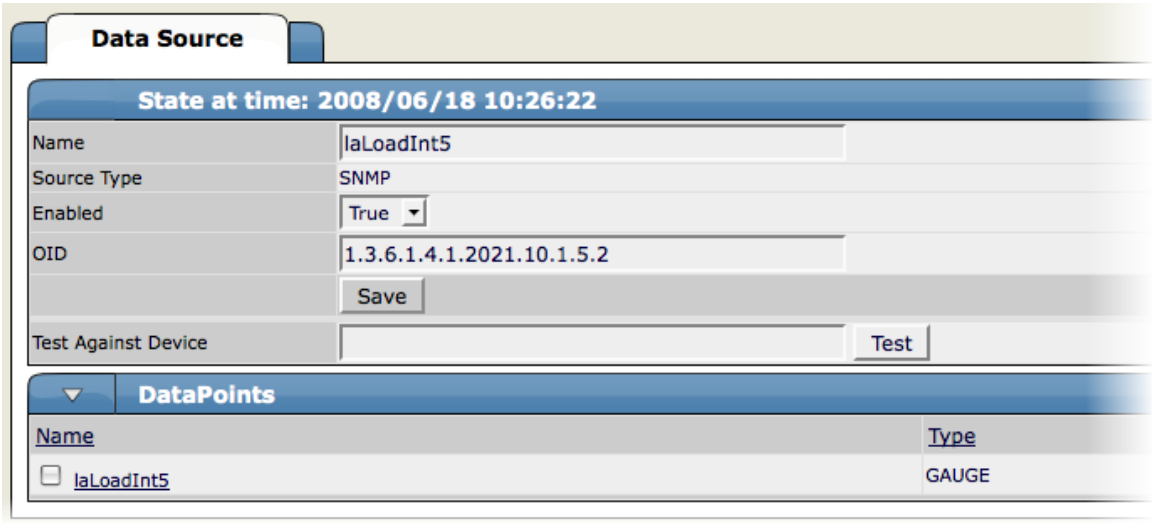
Figure 13.2. Add Data Source



The Data Source page appears.

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

Figure 13.3. SNMP Data Source



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 a data source from the Data Sources area of the Performance Template page.
2. From the DataPoints table menu on the Data Source page, select Add Datapoint.
3. In the Add a New DataPoint dialog, enter a name for the data point, and then click Add.

   ### Note

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

4. Enter information or make selections to define the data point.

   - Name - Displays the name you entered in the Add a New DataPoint dialog.
   - 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.
     - DERIVED - 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.
     - 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.

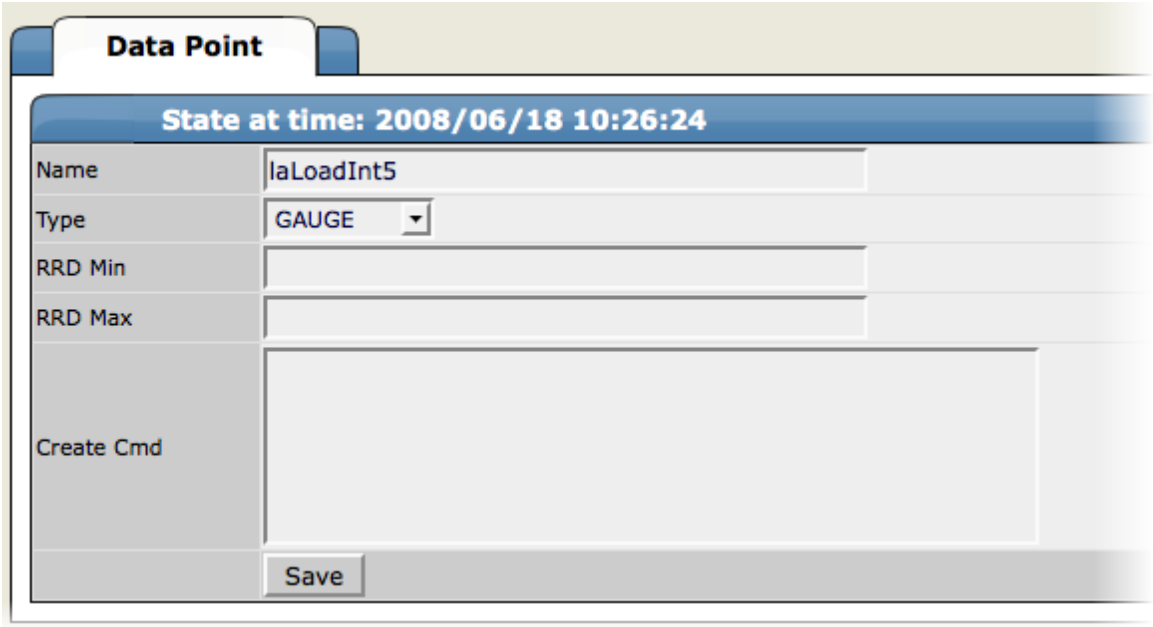
   - RRDMin - Enter a value. Any value received that is less than this number is ignored.
   - RRDMax - Ener a value. Any value received that is greater than this number is ignored.
   - Create CMD - Enter an RRD expression used to create the database for this data point. If you do not enter a value, then Zenoss uses a default applicable to most situations.

     For details about the rrdcreate command, go to:

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

5. Click Save to save the data point.

Figure 13.4. Define a Data Point



6. Thresholds

Thresholds define expected bounds for data points. When a data point goes outside a threshold, then Zenoss creates an event.

Zenoss provides one built-in threshold type: the Min/Max Threshold. Min/Max thresholds inspect incoming data to determine whether exceeds a given maximum or falls below a given minimum. Other threshold types are provided through ZenPacks.

Adding Thresholds

Follow these steps to define a Min/Max threshold for a data point.

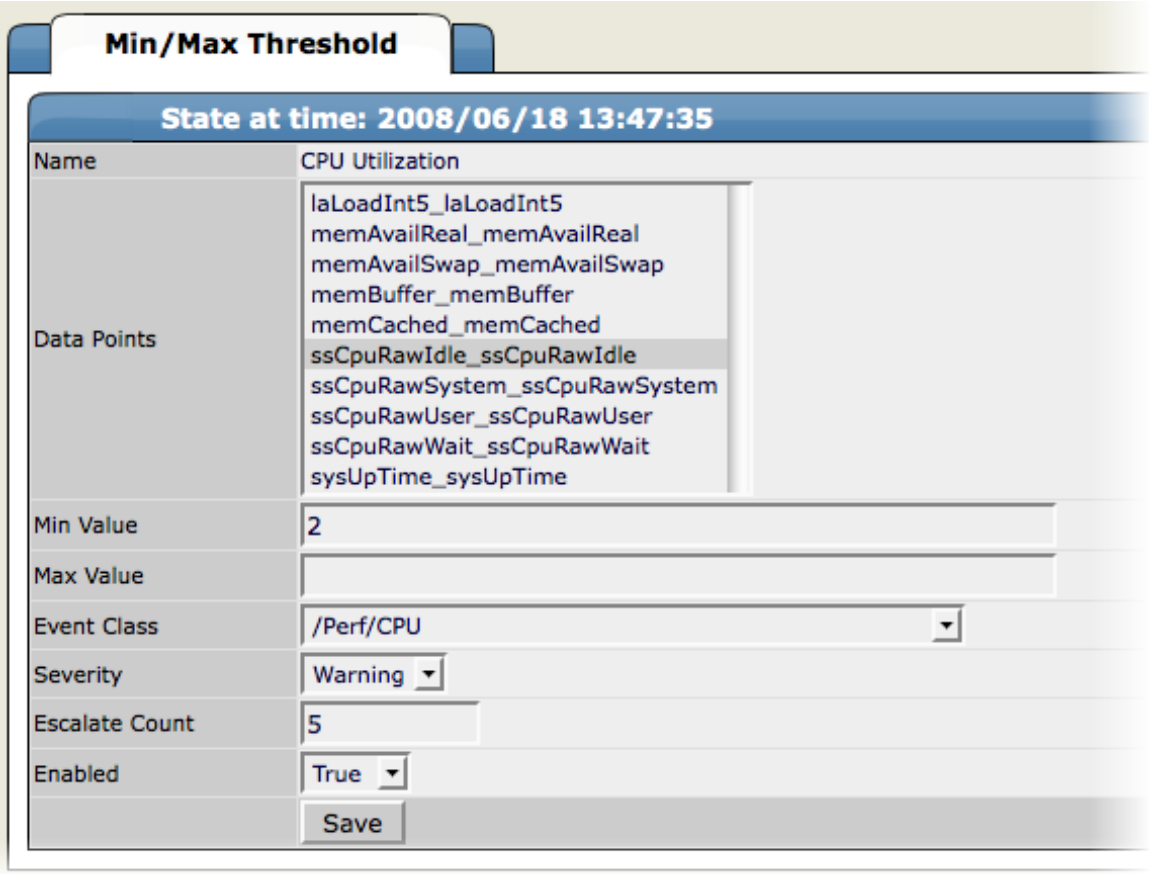
1. From the Performance Template page, select Add a Threshold from the Thresholds table menu.

   The Add a New Threshold dialog appears.

2. Enter an ID for the new threshold, and then click OK.

   The Min/Max Threshold page appears.

Figure 13.5. Add a Threshold



3. Enter or select values to define the threshold:

   - Name - Displays the value for the ID you entered on the Add a New Threshold dialog. This name appears on the Performance Template page.
   - Data Points - Select one or more data points to which this threshold will apply.
   - Min 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.)
   - Max 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.
   - Severity - Select the severity level of the first event 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.
   - Enabled - Select True to enable the threshold, or False to disable it.

4. Click Save to save the threshold.

7. Performance Graphs

Zenoss enables you to set up *performance graphs*, which are graphic representations of device performance data. You can include any of the data points or thresholds from a performance template in a performance graph.

To define a graph:

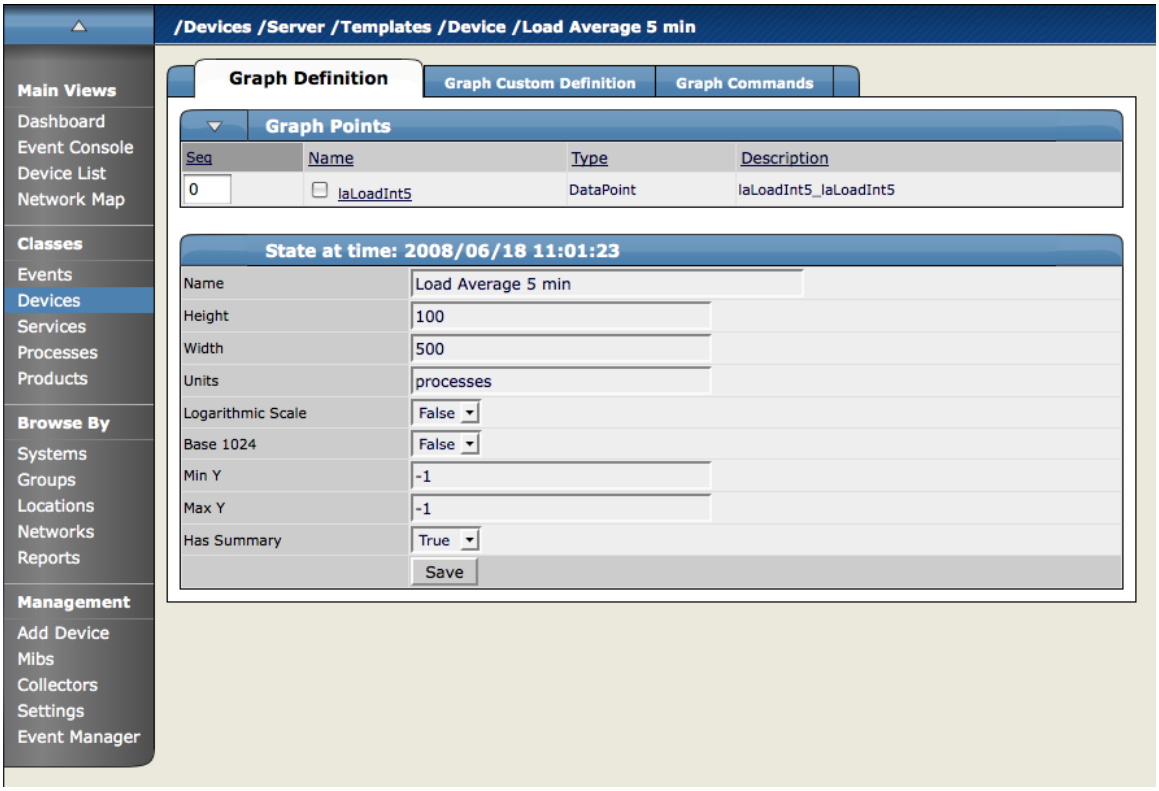
1. Navigate to a performance template whose data you want represented in a graph.
2. From the Graph Definitions table menu, select Add Graph.

   The Add a New Graph dialog appears.
3. Enter the name of the graph, and then click OK.

   The Graph Definition page appears.
4. 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 on the Perf or Status page.
   - 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 13.6. Graph Definition



7.1. 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 Points table menu:

1. Select Add DataPoint, Add Threshold, or Add Custom.
2. Select values for the graph point, and then click OK.

   The new graph point appears on the Graph Definition page.

Note

Thresholds are always drawn before other graph points.

7.1.1. Re-sequencing Graph Points

To re-sequence graph points, enter a sequence number in one or more Seq fields and then select Re-sequence GraphPoints from the the Graph Points table menu. The graphs points are re-ordered as specified.

7.1.2. DataPoint Graph Points

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

7.1.2.1. Adding DataPoint Graph Points

To define a DataPoint graph point:

1. From the Graph Points table menu, select Add DataPoint.

The GraphPoint dialog appears.

2. Select one or more data points defined in this template. One DataPoint 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 select data points as well.

4. Click OK to add the graph point.

7.1.2.2. Editing DataPoint Graph Points

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.
- 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.
- 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.
- 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.
- Color - Optionally specify a color for the line or area. Enter a three- or six-digit hexadecimal color value.
- 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

- 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.

7.1.2.3. Editing Threshold Graph Points

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

7.1.3. Threshold Graph Points

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

To add a threshold graph point to the graph definition:

1. Select Add Threshold from the Graph Points table menu.

The Add GraphPoint dialog.

2. Select one or more thresholds defined in this template. One threshold graph point is created for each threshold you select in this 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.

7.1.4. 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

For details on other RRD commands, go to:

http://oss.oetiker.ch/rrdtool/doc/rrdgraph_graph.en.html

7.2. Custom Graph Definition

The Custom Graph Definition tab allows you to specify your own set of RRD commands to draw a graph. The graph points specified on the Custom Graph Definition tab 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 here. The Available RRD Variables lists the values defined by the graph points that are available for use.

7.3. Graph Commands

The Graph Commands tab shows 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 the Custom Definition tab.

8. Changing Graph Display Order

You can change the sequence of the appearance of graphs. To do this:

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

Chapter 14. Monitoring Devices Remotely via SSH

1. Monitoring Devices Remotely via SSH

You can monitor devices remotely via 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.

1.1. Installing Zenoss Plugins on the Remote Machine

The Zenoss 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 Zenoss plugins.

1.1.1. Zenoss Plugin Installation Technique: RPM

The RPM for the Zenoss Plugins is a noarch RPM, which means it can be installed on any architecture (such as i386, amd64, or ia_64). The only external dependency needed to install the Zenoss plugins RPM is Python. Most Linux distributions include Python in their standard loads.

To install the Zenoss plugins RPM, use the following command:

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

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

1.1.2. Zenoss Plugin Installation Technique: setuptools

Enter these commands to install the Zenoss 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 setuptool's built-in easy_install command to install the plugins. To use easy_install to download and install the Zenoss plugins, run the following command:

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

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

1.1.3. Testing the Plugin Installation

The entry point to the Zenoss plugins is the zenplugin.py command. When run without any arguments, zenplugin.py reports the proper usage of the script, providing insight into which options should be run for troubleshooting.

The Zenoss plugins detect platform-specific, runtime values using plugins. For example, the CPU plugin for the linux2 platform uses /proc to read values. In comparison, the CPU plugin for the freebsd5 platform uses a different technique. In order to test the installation you 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 the zenplugin.py with the plugin name as the argument. The following command line illustrates:

```
$ zenplugin.py cpu
```

1.1.4. Troubleshooting Plugin Installation

1.1.4.1. "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 zenplugin.py was installed is included in your PATH. If you installed by using RPM, you can use the command "rpm -ql zenoss-plugins | grep zenplugin.py". If you installed via setuptools pay close attention to the "Installing..." messages to see the full directory paths.

1.1.4.2. "platform 'XXX' is not implemented. no plugins exist"

This message indicates that Zenoss 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 Zenoss team:

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

1.1.5. Changing Zenoss to Monitor Devices Remotely Using SSH

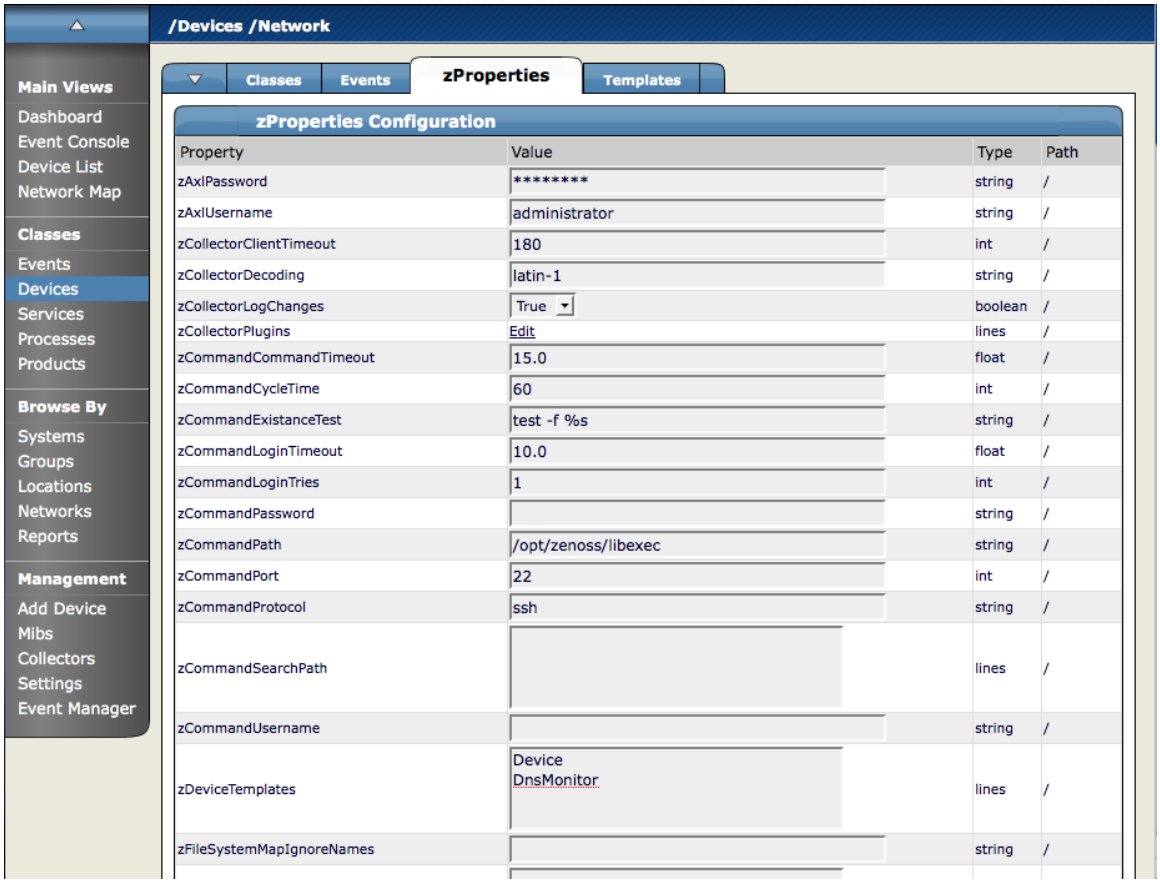
You must edit Zenoss properties for the group where you want to collect remote information using SSH.

1. Navigate to the device class path you want to monitor remotely. You can apply this monitoring per device or per device class path.

2. Change the zProperties value for the group. Click the zProperties tab.

   The zProperties tab appears.

Figure 14.1. Device Group zProperties Tab



You must make changes to the following zProperties:

- 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) setup from the collector to the remote boxes. (It can also use password authorization if the password is entered into zCommandPassword.)

zProperties	Value
zCollectorPlugins	snmp|portscan
zCommandPassword	The SSH password for the remote machine.
zCommandPath	The path to zenplugin.py
zCommandUsername	The SSH Username for the remote machine.
zSnmpMonitorIgnore	True
zTransportPreference	command

Two passes are required for full modeling. The first pass obtains the platform type (so that Zenoss 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.

1.1.6. Using the Predefined /Server/Cmd Device Class

The /Server/Cmd device class is an example configuration for modeling and monitoring devices using SSH. The zProperties have been modified as described in the previous sections, and Device, Filesystem and Ethernet interface templates that gather data over SSH have been created.

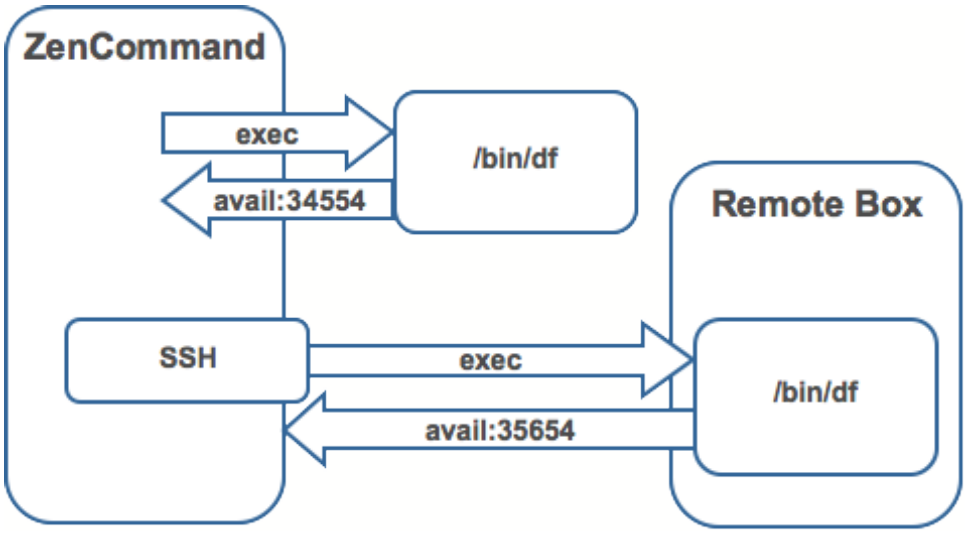
You can use this device class as a reference for your own configuration; or, if you have a device that needs to be modeled or monitored via SSH/Command, you can place it under this device class to use the preconfigured templates and zProperties. You will still need to set the zCommandUsername and zCommandPassword zProperties to the appropriate SSH login information for each device.

Chapter 15. Monitoring Using ZenCommand

1. About ZenCommands

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

Figure 15.1. Running ZenCommands



2. Example: Writing a ZenCommand (check_http example)

You can use a ZenCommand plugin (check_http) to check for specific content of a Web page. (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 with Zenoss.

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 in Zenoss, and then select More > Templates from the page menu.

4. Click Create 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 textsrting1 is text you know to 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.

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. In the Data Points table, 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. View the Perf tab 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.

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.

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 Zenoss.
- event class – Specifies the event class to use when sending events to Zenoss.
- severity – Sets the default severity to use when sending events to Zenoss.
- 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 zProperty 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}
```

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.

Chapter 16. Monitoring Windows Devices

1. Device Preparation for Windows Devices

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

- DCOM is enabled for WMI connections
- SNMP agent is enabled

  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).

SNMP Informant

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. Zenoss 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 Professional or Enterprise, SNMP Informant is not needed (its functionality is included in these versions).

2. Setting Windows zProperties

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

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

  Zenoss recommends that you set this zProperty 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.

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.

4. Optional Windows Configuration

Zenoss 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

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. In Zenoss, navigate to the Windows device, and then click the OS tab.
2. Click the service you want to monitor, and then set the vale of monitor to True.

Note

If you do not see the service you want to monitor in the list, then you can add it. Select Add WinService from the WinServices table menu.

6. Collecting Windows Eventlog Events

Zenoss uses ZenEventlog to collect WMI event log events. Enable the following zProperties to define how Windows event log events are processed and monitored:

- zWinEventLog - Tells Zenoss whether or not to read the event log into the system.
- 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).

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, Zenoss gathers and uses SNMP information the same as any other device sending SNMP traps.

8. Running Commands on Windows Servers Using Winexe

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

Usage:

```
$ZENHOME/bin/winexe [options] //host [command]
```

Options	Use
--uninstall	Uninstall winexe service after remote execution.
--reinstall	Reinstall winexe service before remote execution.
--system	Use SYSTEM account.
--runas=[DOMAIN\]USERNAME%PASSWORD	Run as user (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 17. Windows Performance Monitoring (Zenwinperf)

1. About Windows Performance Monitoring

Note

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

ZenWinPerf is an Enterprise ZenPack that allows performance monitoring of Windows servers without an intermediary Windows server doing the data collection. ZenwinPerf provides the WinPerf Data Source, which uses a Windows performance counter rather than an SNMP OID to specify the value to collect. WinPerf Data Sources are processed by the zenwinperf daemon.

2. Zenwinperf Daemon

The zenwinperf daemon does the actual work of opening the winexe/typeperf connections and sending the data back to Zenoss. This daemon appears on the Zenoss Daemons page and can be started, stopped and restarted from there. From the command line it can be controlled via the zenoss script or by issuing commands directly to the daemon which is symlinked at $ZENHOME/bin/zenwinperf.

2.1. ZenwinPerf zProperties

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

- zWinPerfCycleSeconds - This is how frequently (in seconds) winperf datasources are collected. By default this is set to 300.
- zWinPerfTimeoutSeconds - Deprecated. If no data is collected on a connection after this many seconds the connection is closed and a new one created. This value should be greater than zWinPerfCycleSeconds. By default this is set to 600.
- zWinPerfCyclesPerConnection - Deprecated. This is the number of collection cycles requested of typeperf. After (at most) this many cycles typeperf will exit and zenwinperf will create a new connection to the server and a new call to typeperf. (This value is passed as the -sc value to typeperf.) By default this is set to 12.

2.2. How to Create a WinPerf Data Source

There is an example template with WinPerf Date Sources on the /Devices/Server/Windows device class called ZenWinPerf Example. To create your own WinPerf Data Sources follow these steps:

1. Navigate to either a new or an existing Performance Template and select "New DataSource" from the Data Sources table menu.
2. Enter a name for the Data Source, select WinPerf as the type and click OK.
3. Enter a Windows performance counter in the Perf Counter field. See below for more details on Windows perf counters.
4. Click the Save button. (Notice that a datapoint is created with the same name as the perf counter you selected.)
5. If you wish you can test the counter by entering a device id in the Test Device field and clicking the Test button.

2.3. Windows Performance Counters

ZenWinPerf uses a Windows utility called typeperf to obtain performance data. Information on performance counters can be found with the typeperf documentation here: http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/nt_command_typeperf.mspx?mfr=true

3. /Device/Server/Windows/WMI Device Class

The ZenWinPerf ZenPack includes a /Device/Server/Windows/WMI class that has several new device templates bound and several collector plugins from ZenPacks.zenoss.WinModelerPlugin enabled.

Any Windows device placed in this device class will use WMI as its primary data modeling and data collection mechanism.

A WMI connection cannot be established without a valid set of credentials. These credentials are set with two zProperties: zWinPassword and zWinUser. Like all Windows credentials, the domain should be specified in the zWinUser entry, so use .\username for an account that isn't in the domain but just on the local computer.

The following collector plugins are enabled by default with this new device class:

- zenoss.wmi.WinServiceMap
- zenoss.wmi.CpuMap
- zenoss.wmi.FileSystemMap
- zenoss.wmi.IpInterfaceMap
- zenoss.wmi.IpRouteMap
- zenoss.wmi.MemoryMap
- zenoss.wmi.ProcessMap
- zenoss.wmi.WindowsDeviceMap
- zenoss.wmi.SoftwareMap

The SoftwareMap plugin collects data from the Win32_Product class. The WMI provider for this class is not installed by default on some versions of Windows, especially the Standard Editions of the Server products. To enable to the modeling of software inventory, the WMI Installer Provider must be installed. This can be done by going to the Add/Remove Programs applet in control panel, choosing Windows Components, selecting Management and Monitoring Tools, and then finally the WMI Installer Provider component.

3.1. Device Templates

The following device templates are bound to the new /Devices/Server/Windows/WMI class: Device, FileSystem, and ethernetCsmacd.

3.1.1. Device Template

The Device template provides the following Data Sources:

- cpuPercentProcessorTime
- diskAvgQueueLength
- memoryAvailableKBytes
- # memoryPagesPerSec

- sysUpTime

And the following graphs:

- CPU
- Free Memory
- Paging
- Avg. Disk Queue Length

3.1.2. File System Template

The File System template includes the following Data Sources:

- diskTime
- freeMegabytes
- freeSpace

And then the following graphs:

- Utilization
- IO ServiceTime

3.1.3. ethernetCsmacd Template

The ethernetCsmacd template includes the following Data Sources:

- bytesReceivedSec
- bytesSentSec
- packetsReceivedErrors
- packetsReceivedSec
- # packetsSentErrors
- packetsSentSec

And then the following graphs:

- Packets
- Errors
- Utilization

Chapter 18. SNMP Monitoring

1. SNMP from the Command Line

OID represent the data points where the data for the graphs comes from. Sometimes the reason that a graph is not appearing is because the OID for the particular graph is not valid for the device. You can test this validity using the command line to see if you can return a value. To test the validity of an OID data point giving performance data:

1. SSH to the Zenoss instance.

   Use Username: root

   Password: zenoss

2. Run the command snmp get for one of the OIDs

   In this case lets use the command:

   ```
   $ snmpget -v1 -cpublic build .1.3.6.1.4.1.2021.4.14.0
   ```

   If the OID is valid it will return a value.

   Here are some basic SNMP Operators to gather certain information.

   a. Walk a basic system MIB.

      ```
      snmpwalk -v1 -cpublic <device name> system
      ```

   b. Walk an interface description

      ```
      snmpwalk -v1 -cpublic <device name> ifDescr
      ```

   c. Get a single value.

      ```
      snmpget -v1 -cpublic <device name> ifDescr.2
      ```

   d. Detailed description of an OID value.

      ```
      snmptranslate -Td RFC1213-MIB::ifDescr
      ```

   e. Convert a name to a raw OID.

      ```
      snmptranslate -On RFC1213-MIB::ifDescr
      ```

   f. Convert a raw OID to a short name

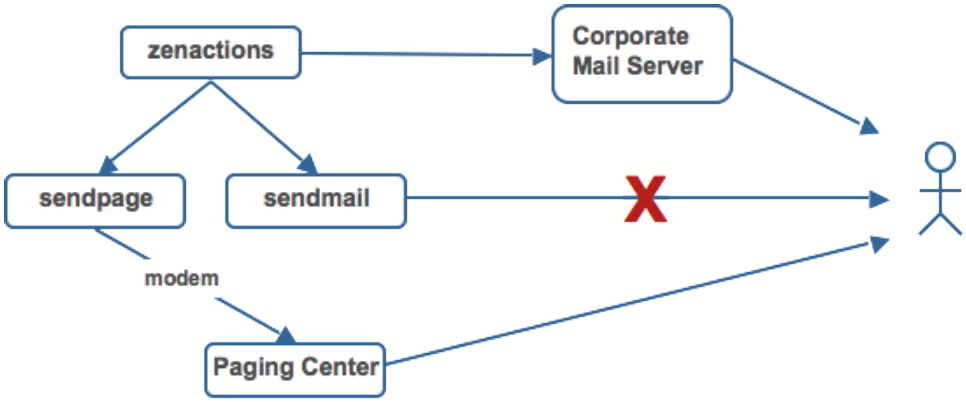
      ```
      snmptranslate -OS .1.3.6.1.2.1.2.2.1.2
      ```

Chapter 19. Alerting Rules

1. Creating and Using Alerts

The daemon ZenActions provides functionality for sending emails or pages based on events received. It continuously evaluates every user's paging rules against the event database. Each user has their own set of alerting rules.

Figure 19.1. Sending Alerts



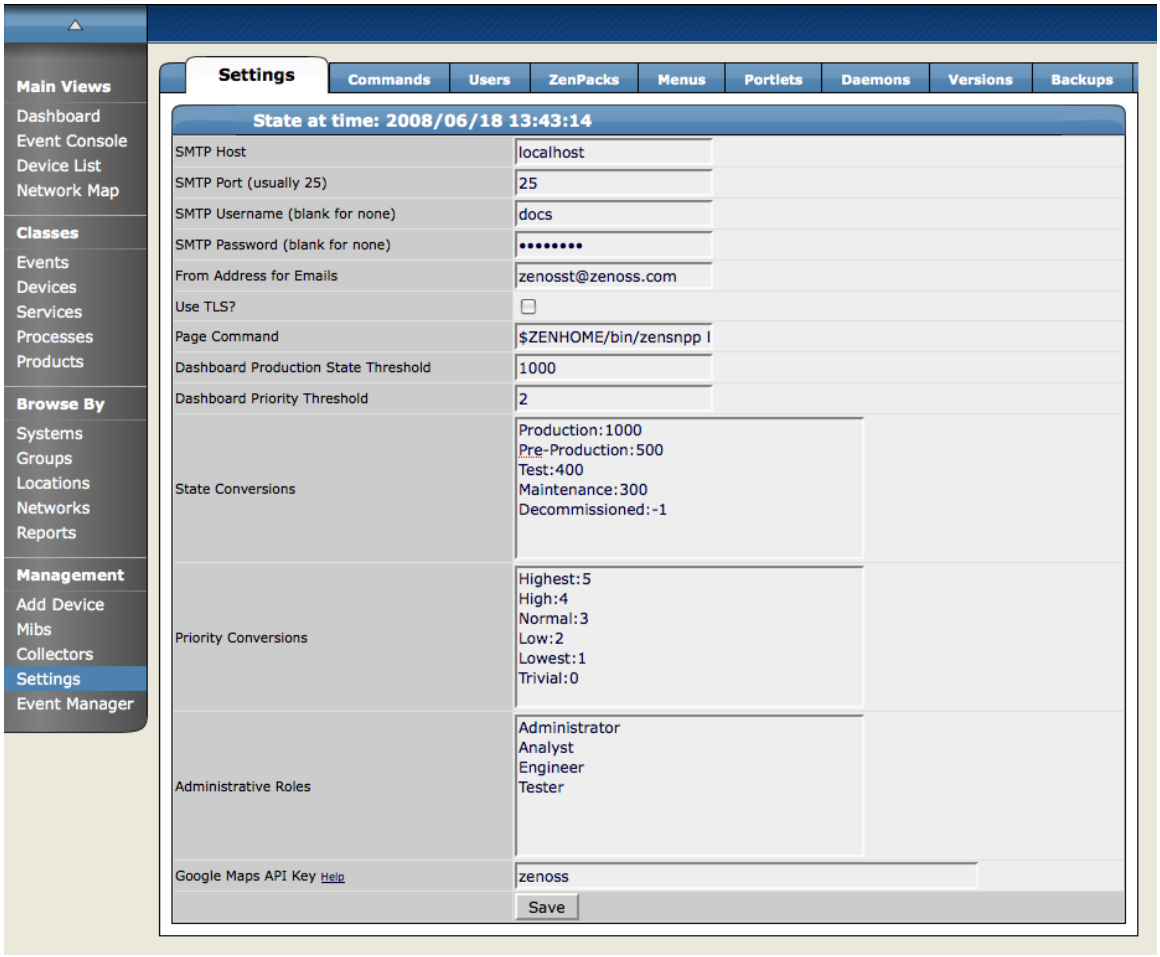
1.1. Setting SMTP Settings For Alerts

To use email and pager alerts, you must have Zenoss pointing to an SMTP relay with the proper settings.

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

   The Settings page appears.

Figure 19.2. Settings Tab Showing SMTP Settings



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

   Now you are prepared to create and use Alerting Rules for the Zenoss system.

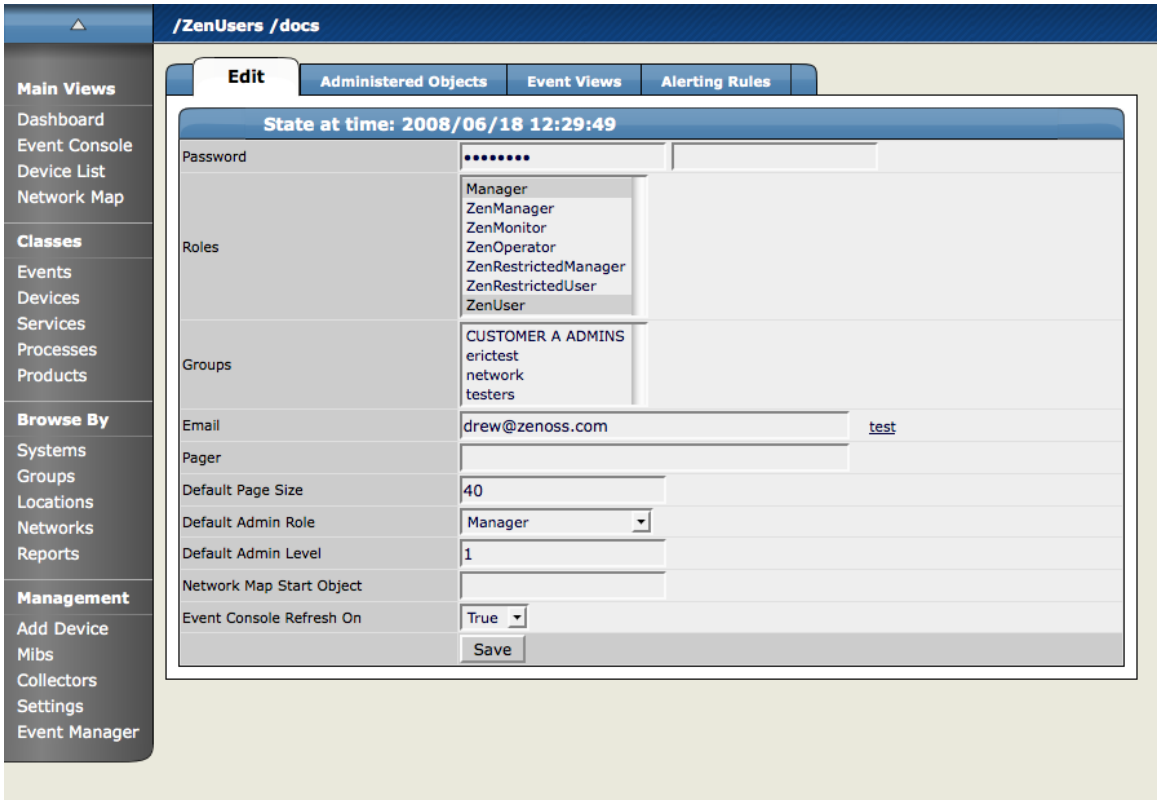
2. Creating a New Alerting Rule

Alerting rules are created on a per user basis. You can add additional recipients for rules, but upon creation, the rules are tied to a user account.

1. From the upper right corner of the Zenoss Dashboard, click the Preferences link.

   The Preferences page appears.

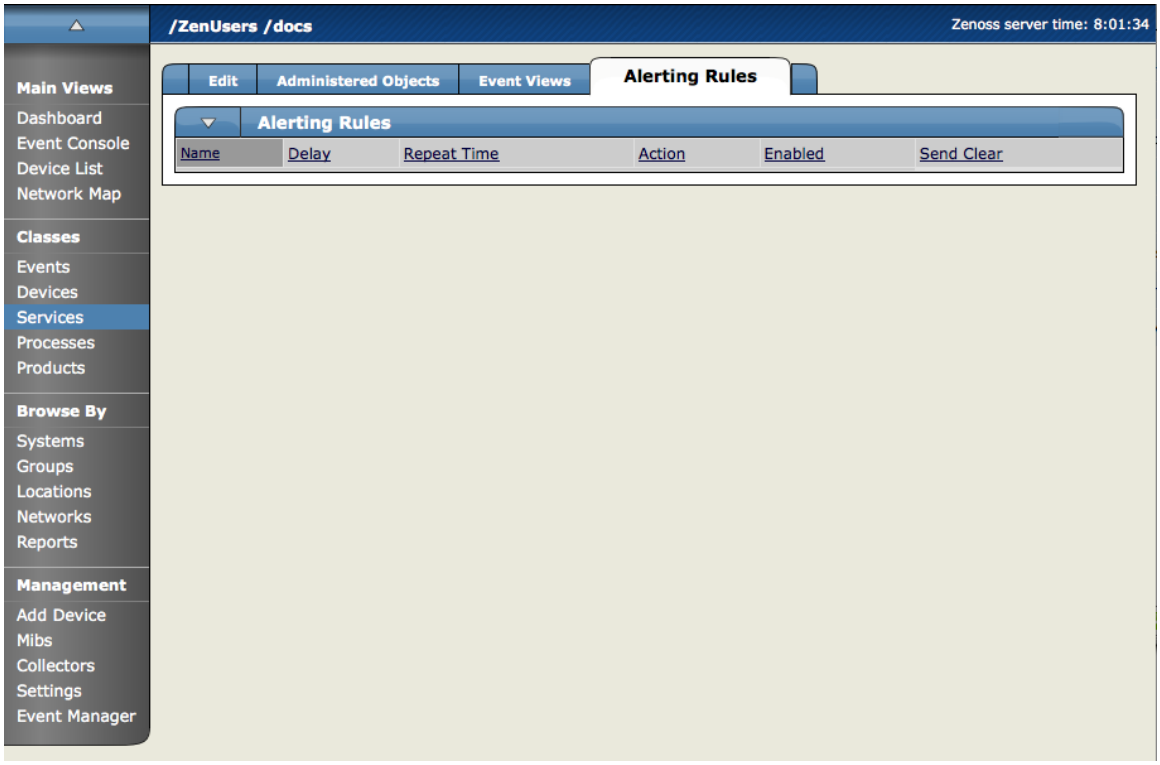
Figure 19.3. Preferences Tab - Edit Tab



2. Select the Alerting Rules tab.

   The Alerting Rules tab appears.

Figure 19.4. Alerting Rules Tab



3. From the Alerting rule table menu, select Add Alerting Rule.

   The Add Alerting Rule dialog appears.

Figure 19.5. Add Alerting Rule Dialog

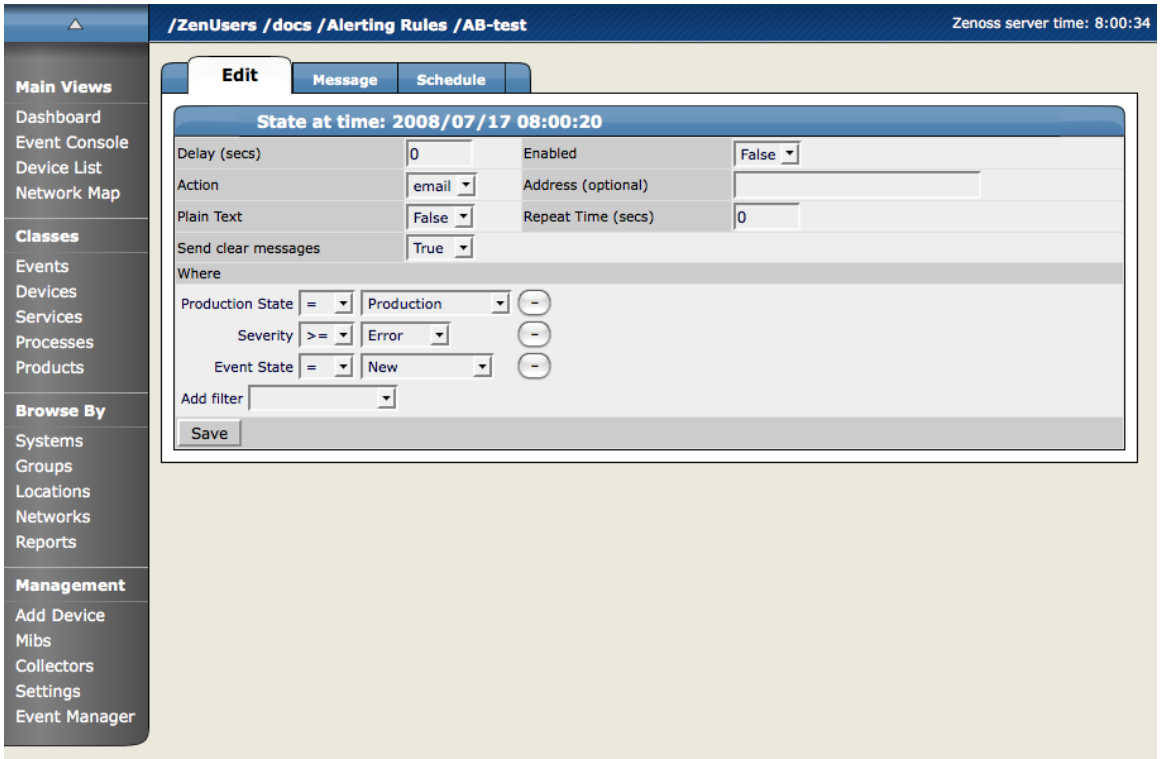
4. In the ID field, enter a name for the alert.
5. Click the OK button.

   The main Alerting Rules page appears showing the alert you just created.

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

   The Alert Details page appears.

Figure 19.6. Alert Details Edit Page



2.1. Define and Enable This Alert

Set the attributes from the Alert Details page, and clicking the Edit tab.

1. Use the Delay field to set the number of seconds to wait before sending the alert. If an event clears before delay time no alert is sent.

2. To enable the alert, set Enabled to True.

3. Use the Repeat Time to set the time for repeating the alert to send the alert every x seconds until the event is acknowledged.

4. In the Action field, select whether you want the system to send email or a page.

   If action is defined as email the event will be emailed. If the default action is set to page, you will need to have an SNMP paging server setup (see the external libs dir of install directory and the sendpage lib does this). Many wireless phone systems have SMTP to SMS gateways so in many cases you use email to act like a page as well.

   By default email alerts will be sent to the email address for this user and pager alerts will go to the specified pager address.

   You can override this by filling in the Address (optional) field.

5. The Where area of the tab sets the thresholds for the Alert.

   The default rule that is created contains the thresholds for an event occurrence where the Event State is “New”, Severity is “greater than Error”, and Production State is “Production”. You can change these thresholds by changing the values in the pop-up menus.

6. You can also add additional filters to the Where area by choosing a filter from the Add Filter menu. Adding a filter creates an additional pop-menu in the Where area where you can choose additional values to filter the event. To Remove any of the filters for the alert, click the (-) minus button.

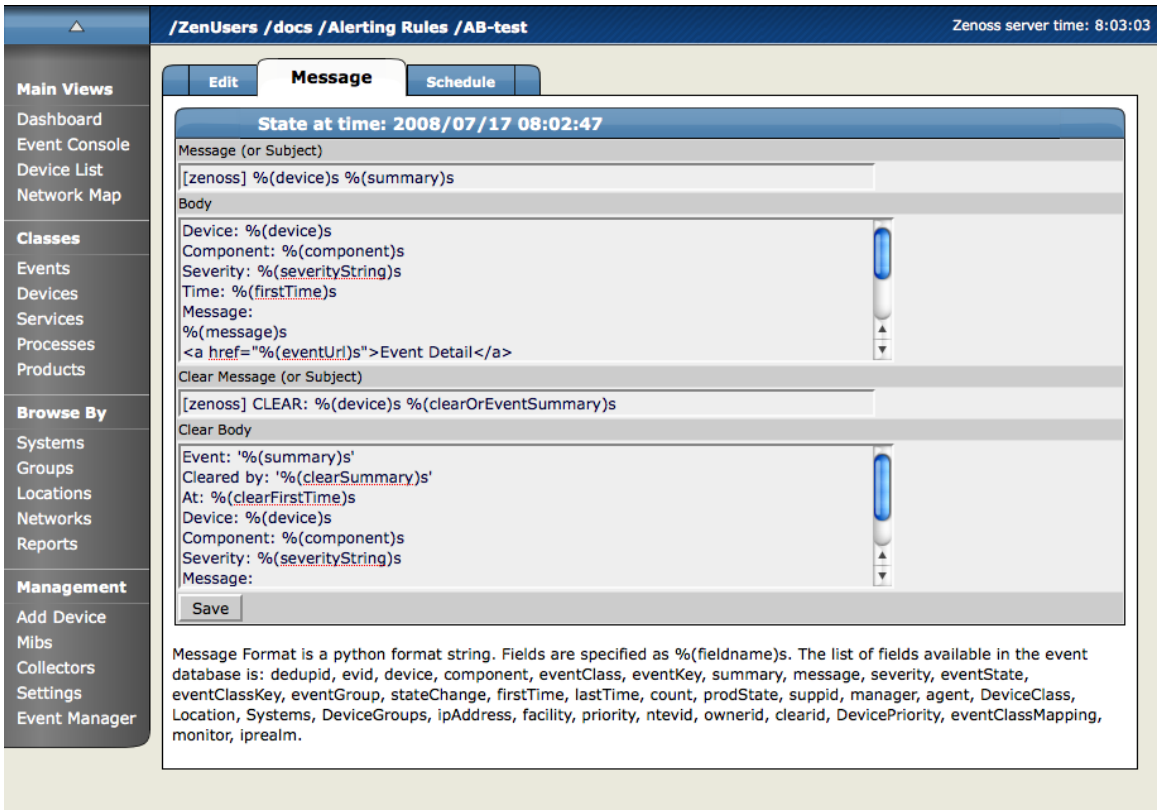
7. Click Save to save the values you entered on this tab.

2.2. 1.1.1 Create the Content of the Alert Message

1. From the Alerting Rules Page, click the Message tab to customize the message that is sent to the specified address.

   The Message tab appears.

Figure 19.7. Alerting Rules Message Tab



2. Use the Message tab to specify the email message subject and body. You actually have two (2) message to create. The first (called Message) is the message to send when the thresholds for the alert are met or exceeded. The second message is the one to send when the event has cleared (called Clear Message).

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

   At the bottom of the page there is a list of the fields available for an alert. A clear events fields are accessed by prefixing the field name with “clear”. For instance the field prodState becomes clearProdState. There is also a special field clearOrEventSummary which will print the clear summary or if it does not exist the original alert summary. This is useful for the subject of a clear alert. In the case where an alert has no clear (it was deleted for the UI for instance) a meaning full subject will still be created.

3. Click Save to save the data you entered on this page.

2.3. Create a Schedule for Sending the Alert

1. From the Alerting Rules page, click the Schedule tab to set up a schedule for the Alert.

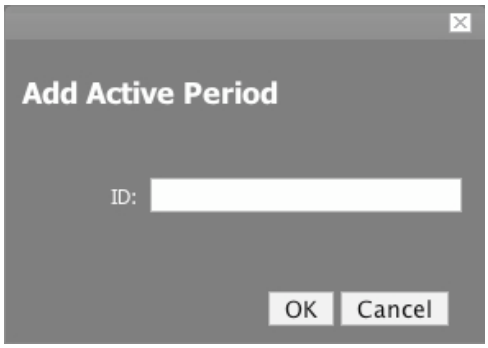
   The Schedule tab appears.

Figure 19.8. Alerting Rules - Schedule Tab

2. To add a new Schedule for the Alert, from the Active Periods table menu, select Add Rule Window.

   The Add Active Period dialog appears.

Figure 19.9. Add Active Period Dialog



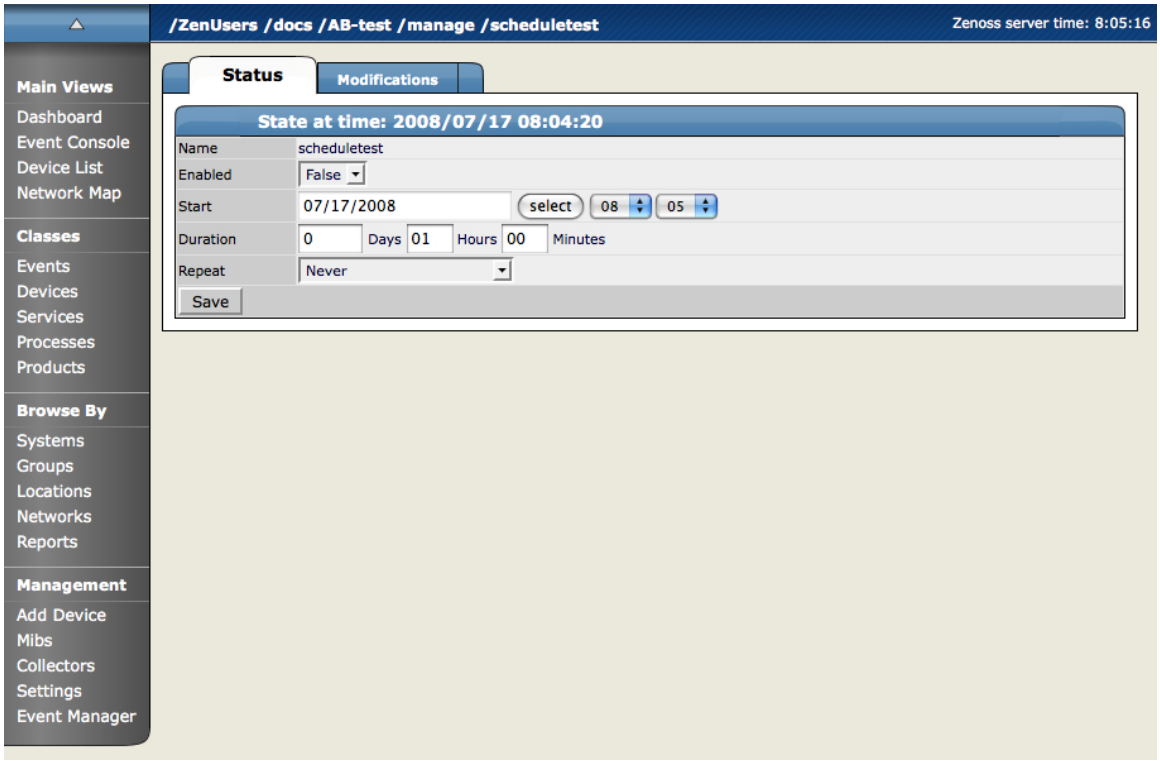
3. Enter a name for the schedule in the ID field and click the OK button.

   The Schedule you added appears in the Active Periods list.

4. Click the name of the new Schedule to set the details for the schedule.

   The Schedule Details page appears.

Figure 19.10. Alerting Rules - Schedule Details Page



5. If you want to restrict this Alert to only monitor at certain times for certain durations, set the Enabled field to True.

6. In the Start area, enter the date you want the alert to start, or click the Select button to choose the date from a calendar.

7. In the fields to the right of the date, select an hour and minute for the Alert to start.

8. Use the Duration area to specify the length of time you want to Alert to be listening based on the start time.

9. If you want the Alerting period to repeat you can choose a time frame from the Repeat pop-up menu. You can choose from:

   - Never
   - Daily
   - Every Weekday
   - Weekly
   - Monthly
   - First Sunday of the Month

10. Choose a number of times to repeat the selected interval.

11. Click Save.

    You have now saved all of the options for creating a new alert.

3. Escalation of Messaging in Zenoss

You can create a messaging hierarchy using some of the different managing tools available in Zenoss.

3.1. Creating an Alerting Hierarchy

You can create a alert Hierarchy based on event status and delays using Alerting Rules. For example, you send an alert to Person A (for example the first level, on call tech) as soon as any event of a given priority occurs, if that event is not Acknowledged or Suppressed (changing the status of the event) within a given time frame (For example, an hour) you create an additional alerting rule to send email to the next person in the Alert Hierarchy, for example Person B.

To create this hierarchy, first, you create an Alerting Rule for the Default case (the initial state). This case is "Any when any new event of whatever priority occurs, alert Person A. To create this default rule, just create the Alerting rule as usual. Set the Delay at 0 (Zero) seconds. Now for the next level in the alerting hierarchy (Person B) you want to say "OK, if Person A does not acknowledge or suppress the event within an hour - then send an Alert to the next person in the hierarchy (Person B). To create this hierarchy in Zenoss, you have to create an additional alerting rule for Person B. There are 2 ways of doing this - you can A) Create an additional rule for the User account you are currently logged in as or B) You can just add the additional (Person B's email address in the Address (optional field). This added address overrides the User account email so email will only be sent to the newly added email address.

For the 2nd Alerting Rule, set the delay to the number of seconds you want to wait after an event has come in and has not been acknowledged or cleared. For our example, we have determined that time-frame to be one hour. One hour is 3600 seconds. So in the Delay box for the Alerting Rule, enter 3600.

Now we need to pass a condition that will keep this alert from firing on all events, even the ones that Person A acknowledges or Suppresses before the hour expires. In the Add filter area, select Event State and then select the event state that will keep this rule from being executed on ALL events, even ones acknowledged by Person A. In this case, an Event is considered to have a state of New unless it has been either Suppressed or Acknowledged so let’s leave the setting at New. So now this alerting rule will send email to Person 2's email if an event remains in the New State for more than 3600 seconds (one hour)

So this section defines a small 2 person hierarchy, but you can use the filters for alerting rules to create much more sophisticated alerting hierarchies and scenarios. Device priority is a new attribute that will work great for defining which person gets which alerts unless a device has a certain priority level in which case everyone gets the alert.

4. Adding Delay and Schedules to Alerting Rules

You can use Delays when creating alerting rules to set up on-call schedules and elevation hierarchies. Using delay will allow you to specify that if an event is not acknowledged in a certain amount of time, then send email to the next person in the hierarchy. You accomplish this by filtering on event state ('New' not acknowledged) and adding a delay. Create an alerting rule for the tier 1 support person that does not have a delay so they find out immediately and can acknowledge the event if possible.

1. Create a second alerting rule (this one will be for the tier 2 person in the hierarchy) and enable it.

2. Use the 'where' clause to say this rule is only in effect for events that have not been acknowledged.

   Delay = 300 (in seconds, 5 minutes)

   In the Where area,

   Production State = Production

   Severity >= Error

   Event State = New

This rule now says fire this alert if there is an event in the system that is New (not acknowledged) for 5 minutes send email to this user.

3. Click the Message tab and in the Message (or subject) field enter the following:

   [Zenoss-delayed] %(device)s %(summary)s

4. In the Clear message (or Subject) area, enter the following.

   [Zenoss-delayed] CLEAR: %(device)s %(clearOrEventSummary)s

5. Click the Schedule tab to edit the schedule. You can tell the rule to only be active when this user is on call (remember each alerting rule is user based).

6. In the Add field enter a name for the new schedule.

7. The new schedule appears in the list. Click the name of the new schedule.

   Name = name of the new schedule

   Enabled = True

   Start = whenever you want the rule to start

   Duration = how long you want this rule to be in effect

   Repeat = number of times to repeat the schedule

   Every = how many of the time periods to repeat for

8. Click Save.

Chapter 20. UI Commands

1. About UI Commands

User commands allow users to execute arbitrary shell commands from within Zenoss. These commands can then be run manually against a single device or a group of devices. The user commands are executed on the Zenoss server, not on the remote device (unless the user command explicitly uses ssh to connect to the remote device.) User commands can be defined on any device class, system, group or location. They can also be defined globally from the Settings page under the Manage tab.

2. Defining UI Commands

1. To access the Define Commands area, from any device you have loaded into your system, open the Device page menu, select More >> Administration.

   The Administration tab appears.

2. From Define Commands table menu, select Add User Command.

   The Add User Command dialog appears.

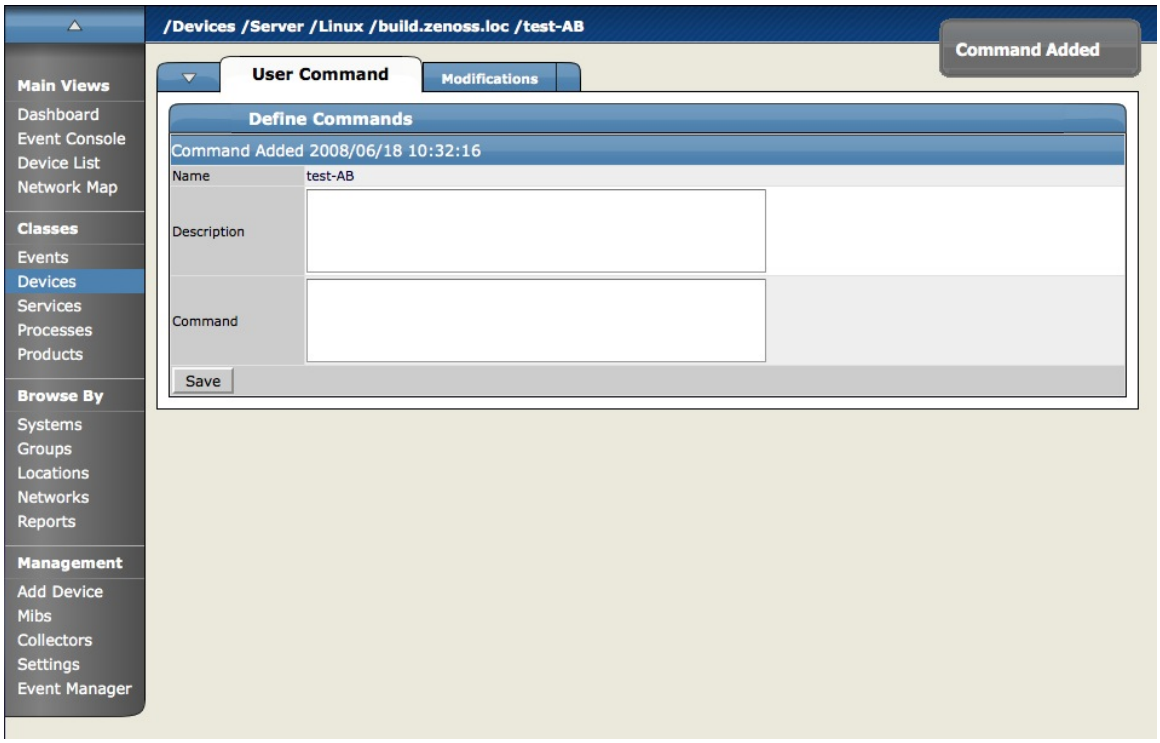
   Figure 20.1. Add User Command Dialog

3. In the Command Id field, enter a name for the command and click OK.

   The Define Command page appears.

Figure 20.2. Define User Command Dialog



4. In the Description field, brief description of what the command will do.

5. In the Command section, enter the TALES expression based command you want to be run on the selected device or devices.

6. Click the Save button.

   The Command is saved and added to the command drop-down menu so you can choose to run it at any time.

2.1. UI Command Example: Echo Command

This section will walk you through creating an echo UI command. You can see the use of TALES expressions in the definition of this command as in many other commands in Zenoss.

1. Go to the Settings - Manage tab.

2. Add a new command called “echoDevice”

3. Echo the name and IP of the device

   ```
   echo name = ${here/id} ip = ${here/manageIp}
   ```

   In a TALES expression ‘here’ is the object that the expression is executed against. Some TALES expressions in Zenoss have other variables like evt for event and dev or device for the device. See the TALES Expression appendix for more information on the syntax of the various TALES expressions.

4. Go to a device and run this command.

5. Now go back to the editing of this command and add some more information to the command.

   ```
   echo name = ${here/id} ip = ${here/manageIp} hw = ${here/getHWProductName}
   ```

6. Now try running the command against a group of devices and see the command outputs.

3. Running Commands from the Zenoss Web UI

Zenoss allows commands to be run though the user interface. It comes with several built in commands such as ping and traceroute. Commands can be run on a single device or on a group of devices.

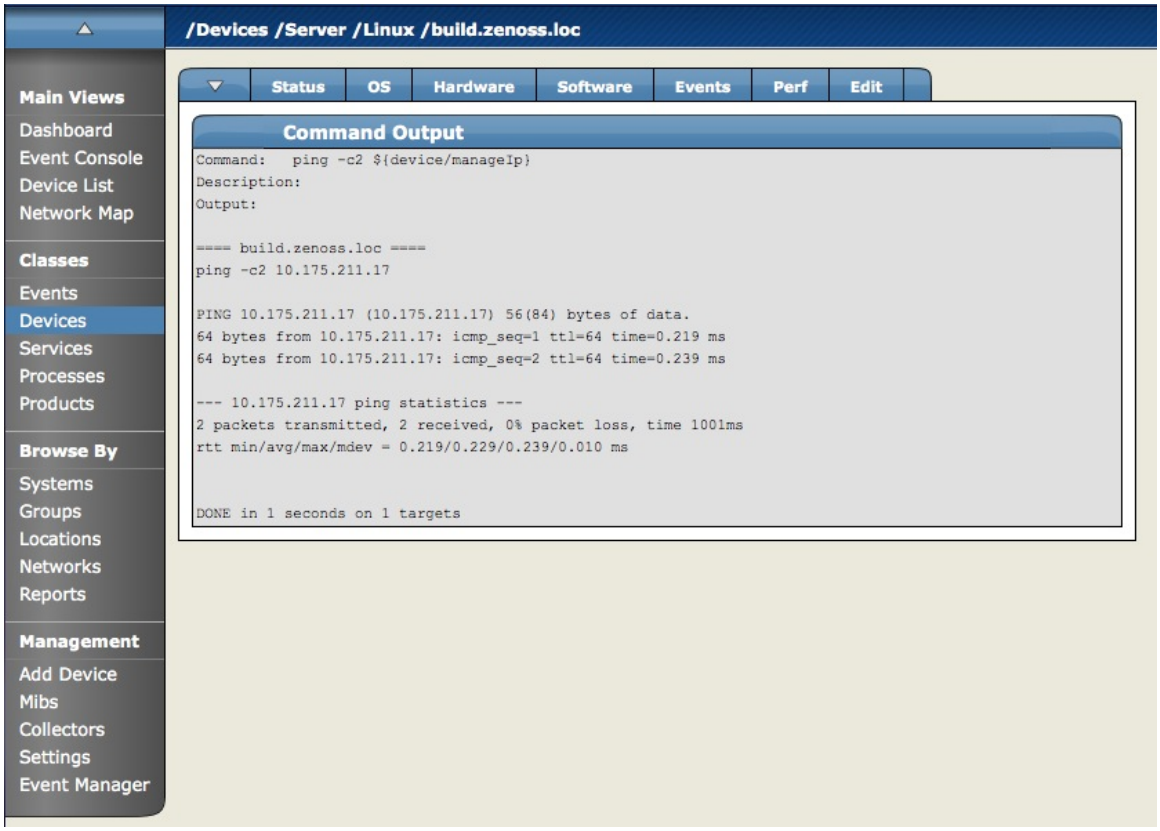
You can run a command one time on a device or group of device from the Zenoss System.

1. Navigate to the Device or Device Group where you want to run the command (and where you have the command defined).

2. 1.From the Device page menu, select Run Commands and then the command name you want to run.

   The command is run and the output appears on the screen.

Figure 20.3. Command Output



4. Auto-Running Commands Based on Events

Zenoss allows you to set up commands to automatically be run against a device based on an event entering into the system.

To run events based on Events:

1. From the Zenoss Dashboard, from the left navigation menu, in the Management area, select Event Manager.

   The Event Manager Edit tab appears.

2. Click the Commands tab.

   The Commands tab appears.

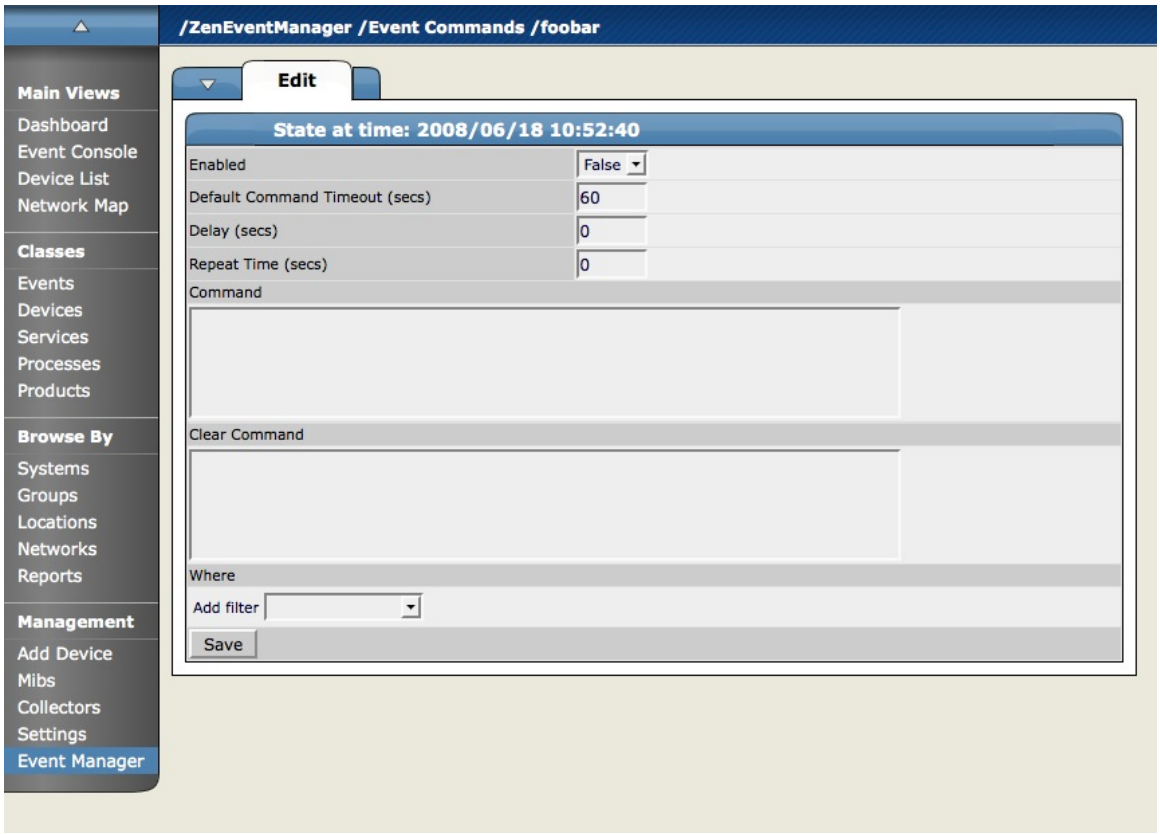
3. Enter a name for the new Command, and click the Add button.

The new command appears in the command list.

4. Click on the name of the new command in the command list to give the command its attributes.

   The Edit Command window appears.

Figure 20.4. Edit Event Command Page



5. Enable the command by setting the Enabled box to True.

6. Now set the other attributes for the command change the default timeout if necessary. Default timeout for the command is 60 seconds.

7. You can also change the delay for the command so there is a gap between when the conditions for the command to be run are met and when the command actually runs.

8. In the Command area, enter the command you want to run when the conditions are met. These commands use TALES expressions. (See the section on TALES Expressions for more detail.)

9. In the clear Command area, enter a command to be run once the event has cleared.

   Commands are built using TALES Expressions. Documented in the Appendix. When the conditions in the Where clause are met, the command you create runs. The power of this command and where and what you can run is only limited by the power of the scripts you create and the permissions on the machine where you want to execute the commands.

   This is optional, but could be useful if, for example, you were running an SMS modem separate from your alerting rules and wanted to send on message in the Command area to send an SMS message if Ping went down, and then when the Ping Down event was clear, you wanted to send an additional message from the SMS modem that it is up.

10. At this point you use the “Where” area to set the conditions for this Command to be run.

11.In the Add Filter drop down select the filters for your Command.

12.You can filter on the Following Event cases:

- Agent
- Count
- Device Class
- Device Priority
- Event Class Key
- Facility
- Manage
- ntevid
- Priority
- Component
- Device
- Device Groups
- Event Class
- Event State
- IP Address
- Message
- OwnerId

  You can add as many conditions as you want to further narrow the cases in which the Command will be executed.

13.Once you have completed setting up the conditions, click the Save button. Now the command is entered and will run when the conditions are met.

Chapter 21. Production States and Maintenance Windows

1. About Production States and Maintenance Windows

Zenoss has the capability to support “Maintenance Windows” or time periods (either scheduled or on the fly) where the monitoring and alerting rules are changed. A set of rules governing monitoring, display and alerting can be collectively defined as “Production States”. When there are temporary changes in Production State, and then a reversion to the original state, this is a Maintenance Window.

2. Production States

Production State is an important concept in Zenoss. It determines whether or not a device is monitored and can be used to control several elements of the event system such as whether or not an event will produce a remote alert (email or page). Typically devices start off their life in state “Pre-Production.” In this state, devices are monitored by default but no remote alerting occurs and events aren’t shown on the Dashboard. Once a device is in full “Production” state monitoring is occurring and remote alerts are sent. If service needs to be performed on a device its state can be set to “Maintenance” to temporarily block any remote alerts.

There are three factors that affect and define the Production State for devices:

1. Whether or not the device is being monitored.

2. Whether or not you want alerting to occur.

3. Whether or not the device appears on the dashboard.

The available Production States are merely combinations of the above:

- Production - you want all three: monitoring, alerting and dashboard.
- Pre-production - you may want monitoring but not alerting or the appearance of the device notices on the dashboard
- Test - you may want monitoring and alerting (sent to one email) and but not displaying device info on the dashboard.
- Maintenance - you want monitoring and collection to occur, and maybe or maybe not the device on the dashboard, just not alerting occurring
- Decommissioned - no monitoring, no dashboard, no alerting

2.1. Defining Production States for Devices

You can set the Production State for a device or group of device(s) by going to the Edit Tab for the device or device group and changing the Production State drop-down to whatever state you want the new Production State to be. The default Production state when you add a device is Production. If you change the Production State for a hierarchy of devices, the Production state propagates down the hierarchy except when you define an exception to the Production State further down the hierarchy.

3. Maintenance Windows

Zenoss Maintenance Windows allow scheduled production state changes of a device or all devices in a System, Group, or Location. Maintenance Windows are defined on the Manage tab of these objects.

A Maintenance Window has a Start Time, Duration, Repeat Cycle and Start and End Production State. A typical Maintenance Window changes the Production State to Maintenance at its start, and to Production at its end. The

Start Production State for the Maintenance Window is the state that the monitoring for the device (or group of devices) is in when the Maintenance Window begins or “opens”. For example, if your device(s) are running along in a Production State of “Production” (meaning you are monitoring and alerting on the devices normally) and the Maintenance Window opening time arrives, the Production State changes to the maintenance Window’s Start Production State.

For example, if the Start Production State is set to Maintenance, this means you want monitoring and data collection to continue to occur for the device, but you don’t want alerts to occur or any warnings to appear on the dashboard. You can use this time to reboot the machine or make configuration changes that would normally create alerts and not have them actually send alerts. You can either schedule a Maintenance Window or change the Production State for the device manually at the time you want to make the changes. When the Maintenance window closes, the device(s) change to the End Production State for the Maintenance Window. You define the End Production State for the Maintenance Window. This refers to the Production State that you want the device(s) to revert to when the Maintenance Window ends. So the Start Production State is the state you want when the Maintenance Window opens - if I were setting up a Maintenance Window, I would define the window such that when its time for the Maintenance Window to occur, I want the Start Production State to be Maintenance, and then when the Maintenance Window time frame expires, I want the Stop Production State to be Production, meaning its back to monitoring and alerting as normal. This would save sending out known alerts as you rebooted or created other, known, alerting events.

3.1. Creating and Using Maintenance Windows

You can create a Maintenance Window for an individual device or for any grouping of devices in any hierarchy you create and define. You can create these windows either for a single device or create a window per device class or system, and have the window propagate to all devices that fall below where you create the window.

To create a new maintenance window:

1. Navigate to the device or group of devices where you want to define the maintenance window. Open the page menu and select More and then Administration.

   The Administration page appears.

2. From the Maintenance table menu, select Add Maint Window.

   The Add Maintenance Window appears.

3. In the ID field enter a name for the maintenance window and click OK.

   The name appears in the Maintenance Window list.

4. To define the window, click the name of the Maintenance Window.

   The Maintenance Window Status Tab appears.

5. Define the attributes for this Maintenance window.

   - Name - Name of the maintenance window.
   - Enabled - True or False as to whether you want this maintenance window active.
   - Start - The time for the window to become active.
   - Duration - The length of time for the window to be in effect starting from the Start time.
   - Start Production State - Defines the production state for the window before it opens.
   - Stop Production State - Defines the production state for the object once the maintenance window closes.

6. Click Save.

Chapter 22. Reporting

1. About Zenoss Reporting

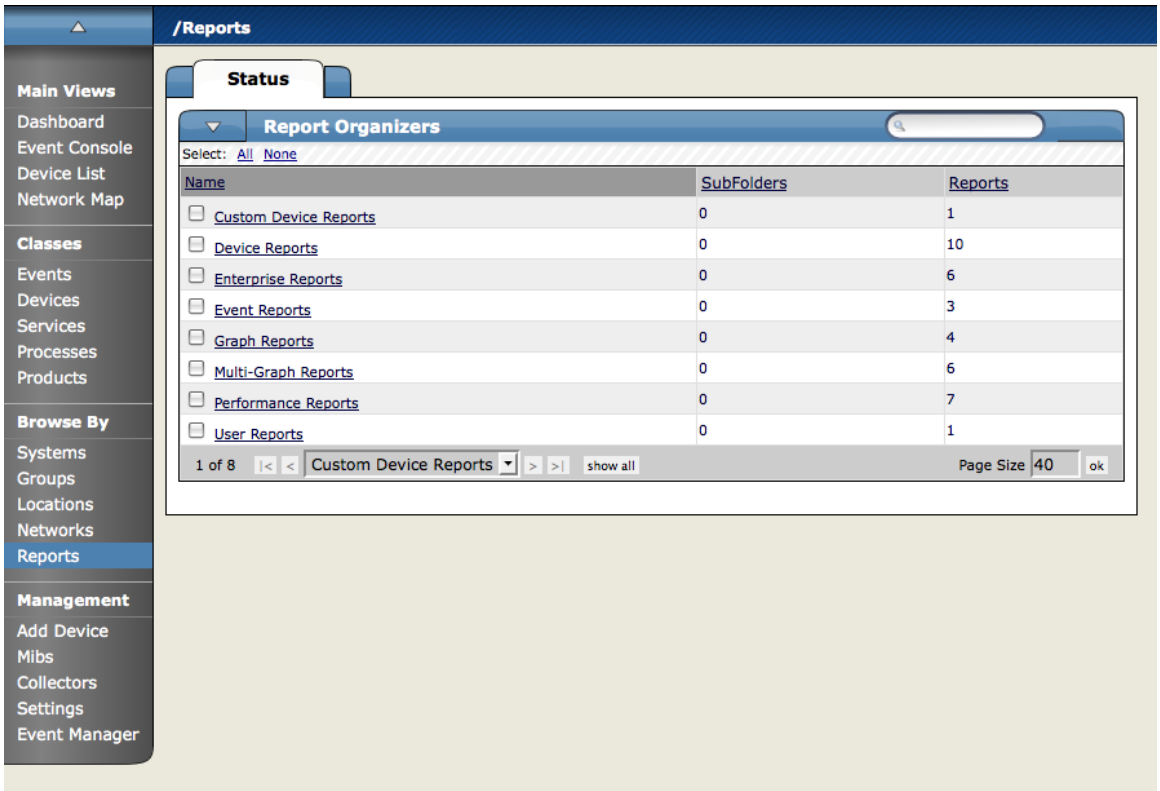
Zenoss aggregates and reports, over time, on the data you set it up to monitor.

Zenoss reports are categorized as:

- Device reports
- Performance reports
- Graph and multi-graph reports
- Performance reports
- Other reports, which includes a notification schedule and heart beat report
- Custom device reports

To work with Zenoss reports, select Reports from the navigation area. The Reports list appears.

Figure 22.1. Reports List



2. Organizing Reports

You can organize reports in Zenoss in the same way you organize other elements in the system. You can create categories, sub-categories, and organizers, as well as report hierarchies and relationships.

3. Navigating and Sorting Report Results

Click a report column header to sort the report by that information category. You also can sort reports based on a key word that you enter in the Filter field.

4. Exporting Reports

You can export data from any Zenoss report, as comma-separated value (.csv) files.

To export report data, click Export All (located at the bottom of a report).

4.1. Advanced: 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 the following:

```
<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">
```

The normal report markup goes here:

</tal:block>

</tal:block>

</tal:block

The first definition is a call to some method that retrieves the objects for the report. This might be a list, tuple or an iterable class.

If we are doing an export then we need this to be a list, so the second tal:define line makes sure we have a list in the event that we are doing an export. It's good to not do this if we are not doing an export. Large reports might run into performance issues if an iterable is converted to a list unnecessarily.

tablename is defined here for use by the getBatch() call that follows.

exportFields is a list of data to be included in the export. These can be attribute names or names of methods to call. See DataRoot?.writeExportRows() for more details on what can be included in this list.

Within the <tal:block metal:fill-slot="report"></tal:block> block goes the report markup you would use when not including the export functionality.

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.

5. Reports Included With Zenoss

Basic Zenoss reports are described in the following sections.

5.1. Device Reports

Device Reports aggregate and display data over many devices and device attributes.

- All Monitored Components – The All Monitored components shows all of the components currently being monitored. This is NOT all components in the system, only the ones that Zenoss is currently collecting performance data on. The information that appears in this report is: the device name, Component, the type of component (when available), the description, and the status of each device.
- Model Collection Age – The Model Collection age report shows information about the history of the modeling that Zenoss does of each device. The information available in this report includes the device name, Device Class, the time when the device was first seen by Zenoss, the last time Zenoss Collected Data on this device, and any changes made to the configuration at that time.
- Device Changes - The Device Change report shows information about the history of any changes that Zenoss detects when modeling each device. This report only shows devices with changes. The information available in this report includes the device name, Device Class, the time when the device was first seen by Zenoss, the last time Zenoss Collected Data on this device, and any changes made to the configuration at that time.
- Ping Status issues – The Ping Status Report shows the device name, the device class, the Product name, the current state of the device and the Ping and SNMP status. The only devices that will show up here are devices that actually have or have had Ping issues.
- SNMP Status Issues - The SNMP Status Report shows the device name, the device class, the Product name, the current state of the device and the Ping and SNMP status. The only devices that will show up here are devices that actually have or have had SNMP issues.
- New Devices – The New Devices report shows devices that have been recently added. The report shows the Device Name, the Device Class, when the device was first seen and the model collection age.
- New Devices – The New Devices report shows devices that have been recently added. The report shows the Device Name, the Device Class, when the device was first seen and the model collection age.

5.2. Event Reports

Event reporting gives you aggregate data about events, event mappings and event classes.

- All Heartbeats – The All Heartbeats Report shows all heartbeats for monitored devices. Heartbeats are reported by component and number of seconds.
- All Event Classes – The All Event Classes report shows all of the event classes that reside in the Zenoss system. It also breaks these classes down by SubClasses, the number of instances of that class in the system and the number of events in the system associated with each Event Class.
- All Event Mappings – The All Event Mappings shows all of the event mappings that you have created throughout the system. You can sort them by Event Class key, Evaluation or number of events per event class.

5.3. Performance Reports

Performance Reporting allows you to roll up performance data across the Zenoss system.

- Aggregate Reports – The Aggregate reports shows the performance graphs for all of the devices in the system in graphical format. Common performance stats include PU Usage, Aggregate Free Memory, Aggregate Free Swap, and Network Output/Input. You can edit the graph parameters by clicking on the graph. You can change the Width, height, Min Y and Max Y axis. You can also specify which devices are included in the aggregate and the time span for the graph.
- File System Utilization Report – The File System Utilization Report shows the Total Bytes, used Bytes, Free Bytes and % Utilization for each device. You can customize the report through the interface for such attributes and start/end Date and Summary type; either average or Maximum.
- CPU Utilization Report – The CPU Utilization report shows all of the Monitored Interfaces, the list of devices and the load average and % Utility. You can customize the report through the interface for such attributes and start/end Date and Summary type; either average or Maximum.

- Threshold Summary – The Threshold Summary Report identifies the devices that approach or exceed their thresholds and reports them in a list. You can see the device, the component, the event class, count, duration and percentage. You can filter this list by Event Class or to see all Event Classes, leave the event class Selection List blank. You can also change the start and end date for the reporting data.

- Availability Report – The Availability Report reports availability of devices in percentage form. You can filter on device, component, event class or severity. You can also further limit the time frame for the availability.

  The availability report provides an **estimate** of the time a device or component is available for monitoring things such as Service-level Agreements (as an example).

  This report summarizes the amount of time a device or component can be considered "unavailable". The definition of unavailable differs depending on what type of service you require. If a web server goes down, the web service is not available, even though the device may actually be reachable ("ping-able") on the network.

  As Zenoss collects events against devices and components, these events are classified into categories that are useful for determining if a particular service is unavailable. For example, the class "/Status/IpService" can provide all the events related to a failed IP Service. The availability report adds up all of the time (lastTime - firstTime) for all of the events based on device, component, or event class. This defines the "unavailable" period. The availability is the remainder of the period.

  This is an optimistic estimate of availability because events that occur only once and events that have the same first and last time, are not counted. Another assumption the estimate makes is that devices without any matching events during the Availability time range are assumed to be 100% available.

- Memory Utilization – The Memory Utilization Report provides system wide information about the memory usage for devices in the system. This report shows Total memory, Available memory, Cache memory, Buffered memory, and percent of memory utilized.

5.4. User Reports

User Reports refer to report based on user account information and changes within the Zenoss system.

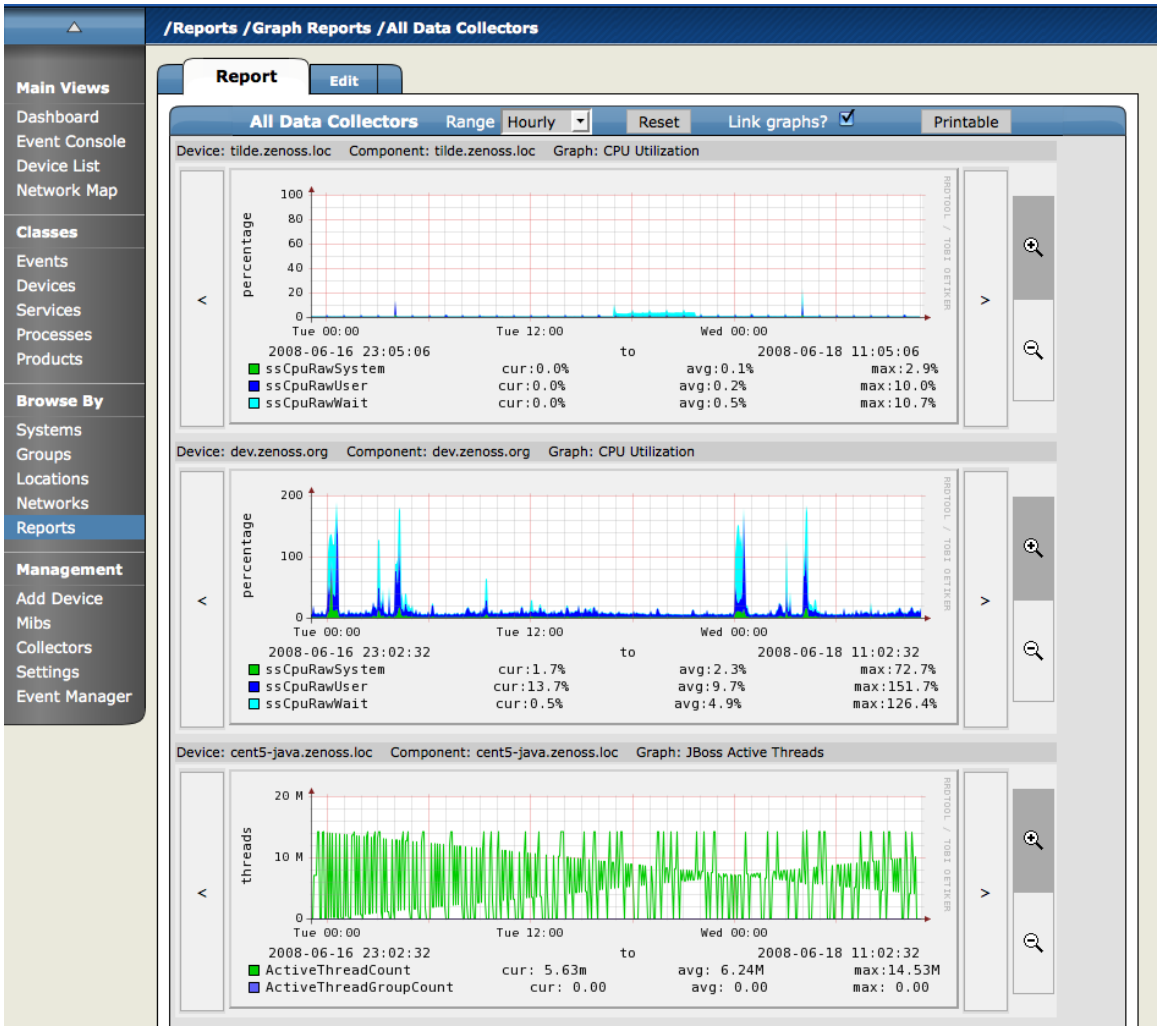
- Notification Schedules – The Notification Schedules Report shows all of the alerting rules and their associated details with each one.

6. Graph Reports

Graph Reports allow you to assemble graphs from specific Devices and Device Components into a single report. Click on Reports in the left navigation menu, then on the Graph Reports organizer to view or create Graph Reports. Graph Reports have a normal view which is similar to the graph views for Devices and Device Classes. Graph Reports also have a Printable view more appropriate for printing which can be brought up by clicking on the Printable button at the top of the report view.

Note that Graph Reports can only display graphs that already exist on Device or Components within Zenoss. You cannot define new graphs or alter existing graphs from within a Graph Report. If you need this type of functionality you probably want MultiGraph Reports instead.

Figure 22.2. Graph Report



6.1. Creating a Graph Report

From within the Graph Reports organizer or a sub-organizer use the Add Graph Report menu item to create a new report. After entering a name for the new report you will be taken to the edit page. The fields for a Graph Report are:

- Name - The name of the report is displayed at the top of the report.
- Title - This description is displayed in the list of reports for the report organizer. It is also available for use in the comments.
- Number of Columns - This is the number of columns the graphs will be displayed in on the report.
- Comments - The comments are displayed at the top of the Printable version of the report. This is a TALES evaluated string that may contain html formatting. The variables available to the TALES expression are now (the current datetime) and report (the report object itself.)

6.2. Adding Graphs

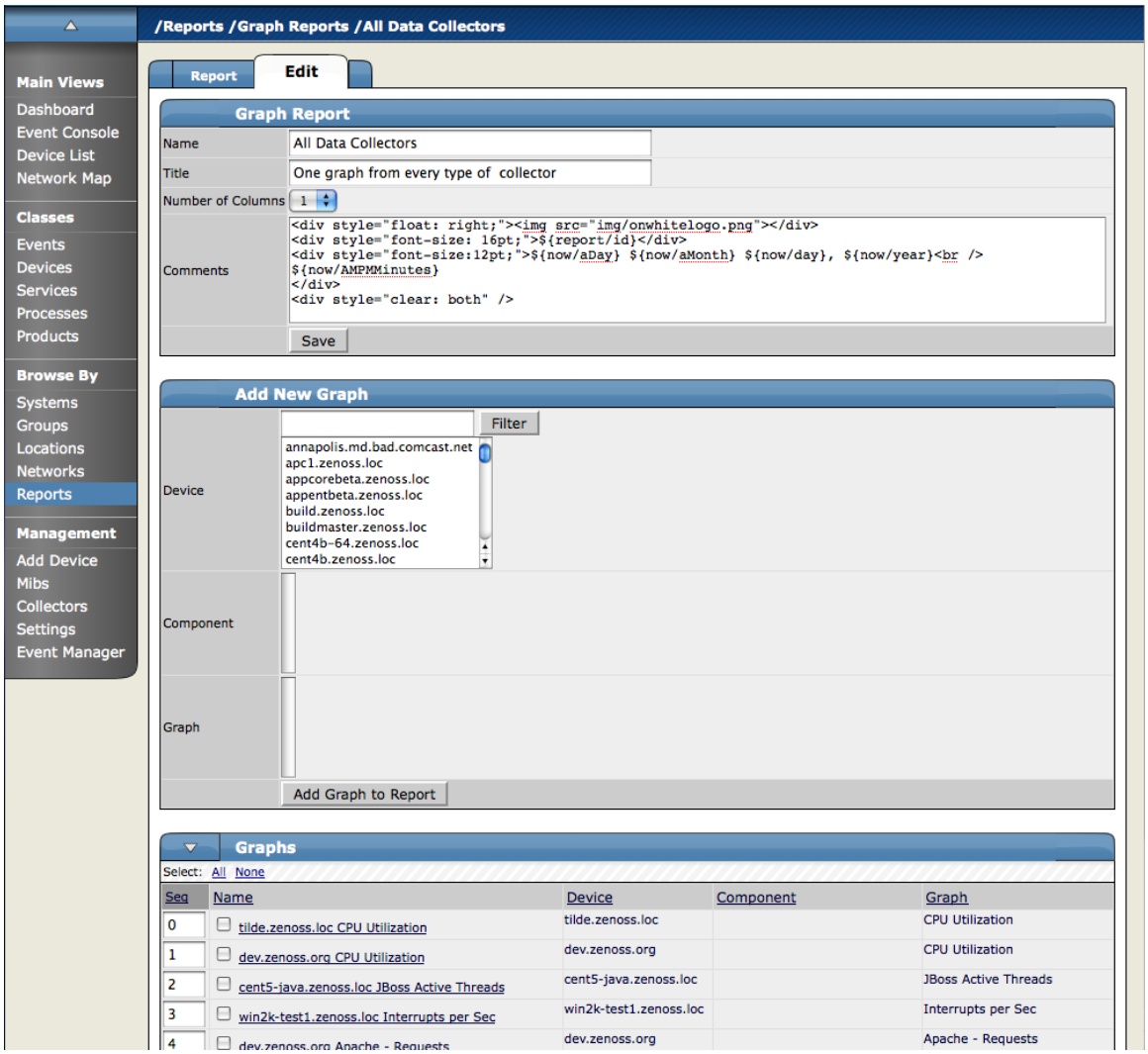
The Add New Graph section of the Edit page allows you to add one or more graphs to the report. The first step is to select one or more Devices. The list of Devices can be narrowed by entering a search string to the left of the Filter button and pressing Return or clicking the Filter Button. When you select one or more Devices the Component list will display the names of all Components defined on at least one of the selected Devices. The Graph list will

list the Graphs available on one or more of the listed Devices. When one or more Components are selected the Graph list will display Graphs from the selected Components rather than the selected Devices.

At any point you may select one or more Graphs and use the Add Graph to Report button. Zenoss steps through each selected Component (if any are selected) or Device (if no Components are selected) looking for graphs with the given names. Matching graphs are added to the Graph Report.

Graph Reports maintain a static list of graphs which does not change when graphs are added or deleted from Performance Templates. For example, take DeviceA which has only one graph called Graph1 and DeviceB which has two graphs named Graph1 and Graph2. On the Graph Report edit page if you selected DeviceA and DeviceB the list of graphs would include Graph1 and Graph2. Selecting both graphs and clicking the Add Graph to Report button would add three graphs to the report: DeviceA's Graph1, DeviceB's Graph1 and DeviceB's Graph2. If at some later date you created a Graph2 on one of DeviceA's Performance Templates it would not automatically appear on the Graph Report, you would have to edit the report to specifically add it. Similarly, if one of the graphs was removed from DeviceB's Template (or if DeviceB was deleted from Zenoss) you would need to manually remove them from the Graph Report.

Figure 22.3. Graph Report Edit Page

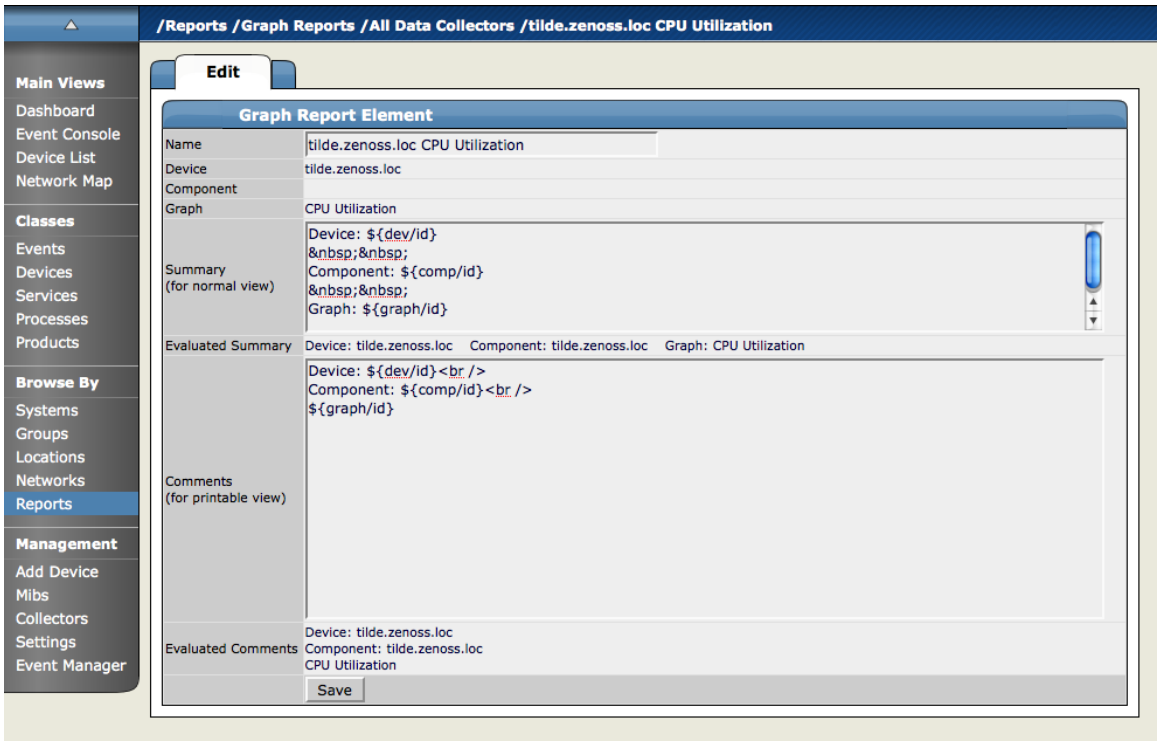


6.3. Customizing Graph Text

The Graphs section of the Edit page lists the graphs that are included in this report. Click the name of any of the graphs takes you to an edit page where you can edit the text that appears with the graph when the report is viewed. The Summary field is displayed above the graph in the normal report view and the Comments field is displayed

to the left of the graph in the printable view. Both of these fields may contain TALES expressions with these variables: dev (the device), comp (the component) and graph (the graph.)

Figure 22.4. Graph Report Element



6.4. Organizing Graphs

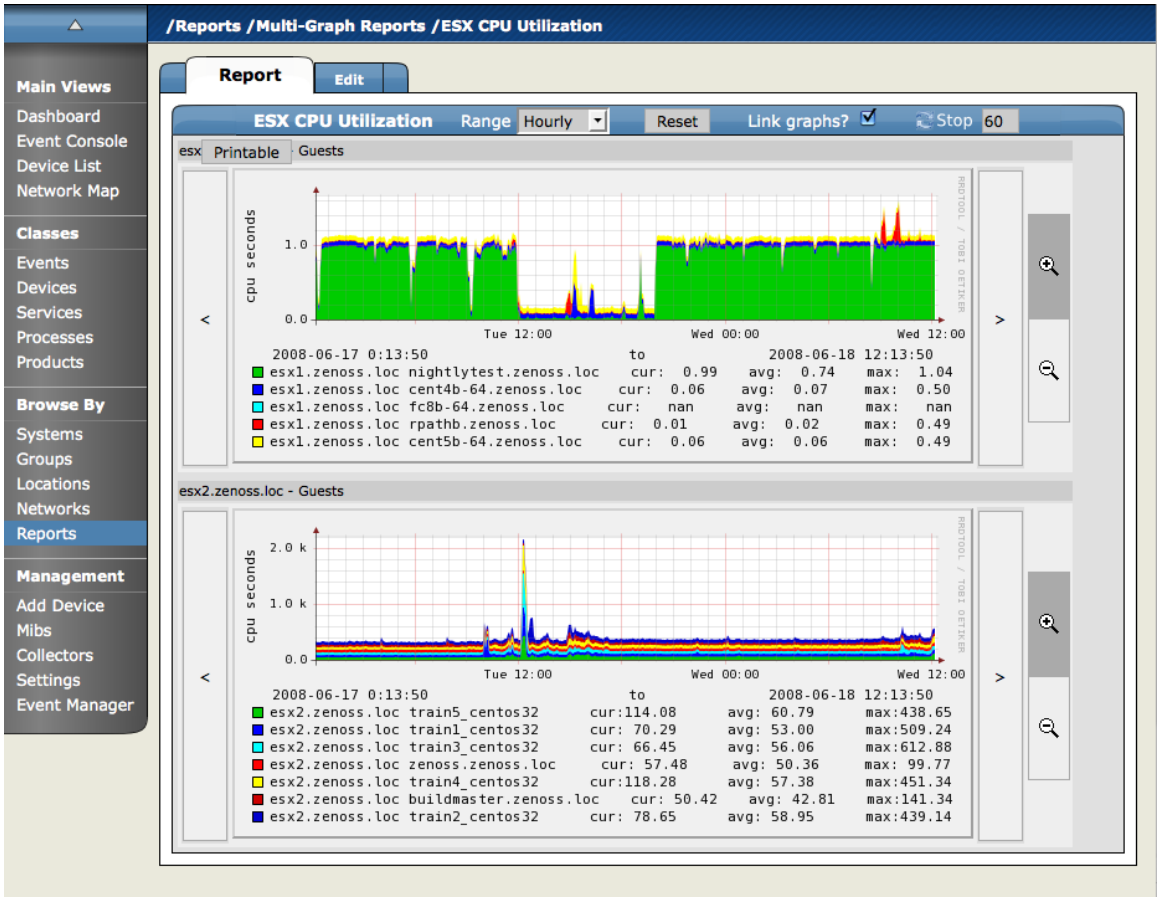
In both the normal and printed view of the report graphs are laid out from left to right in as many columns as specified on the report's edit page. For example, if a report has three columns then the first three graphs are placed on the first line of the report and the fourth graph would be the first one on the second row. By altering the sequence of the graphs on the edit page you can control where each graph appears on the report view. You change the sequence by editing the values in the Seq column beside each graph name and choosing the Re-sequence Graphs menu item.

7. MultiGraph Reports

MultiGraph Reports are a powerful mechanism for combining data from different Devices and Components into a single report. You can create a Graph Definition and have it drawn once for each in a group of Devices and Components that you define. Alternatively, you can have the data for those graphs combined into a single graph. The Graph Definitions are very similar to those used in Performance Templates except that these are defined and used exclusively within a single report. The groups of Devices and Components you assemble are called Collections. Specifying which Graph Definitions to apply to which Collections is done through Graph Group objects. Like Graph Reports, MultiGraph Reports have two different views, normal which looks similar to Device and Component graph pages and printable which is formatted more suitably for printing. The normal view is seen on the Report tab of any MultiGraph Report. At the top right of that view is a button named Printable for displaying the printable version.

MultiGraph Reports include their own Graph Definitions and thus do not use the Graph Definitions that are defined within Performance Templates. If you want to create a report that includes graphs defined on Templates then you may wish to use GraphReports rather than MultiGraph Reports.

Figure 22.5. MultiGraph Report Graphs



7.1. Creating A MultiGraph Report

From within the MultiGraph Reports organizer or a sub-organizer use the Add MultiGraph Report menu item to create a new report. After entering a name for the new report you will be taken to the edit page. The fields for a MultiGraph Report are:

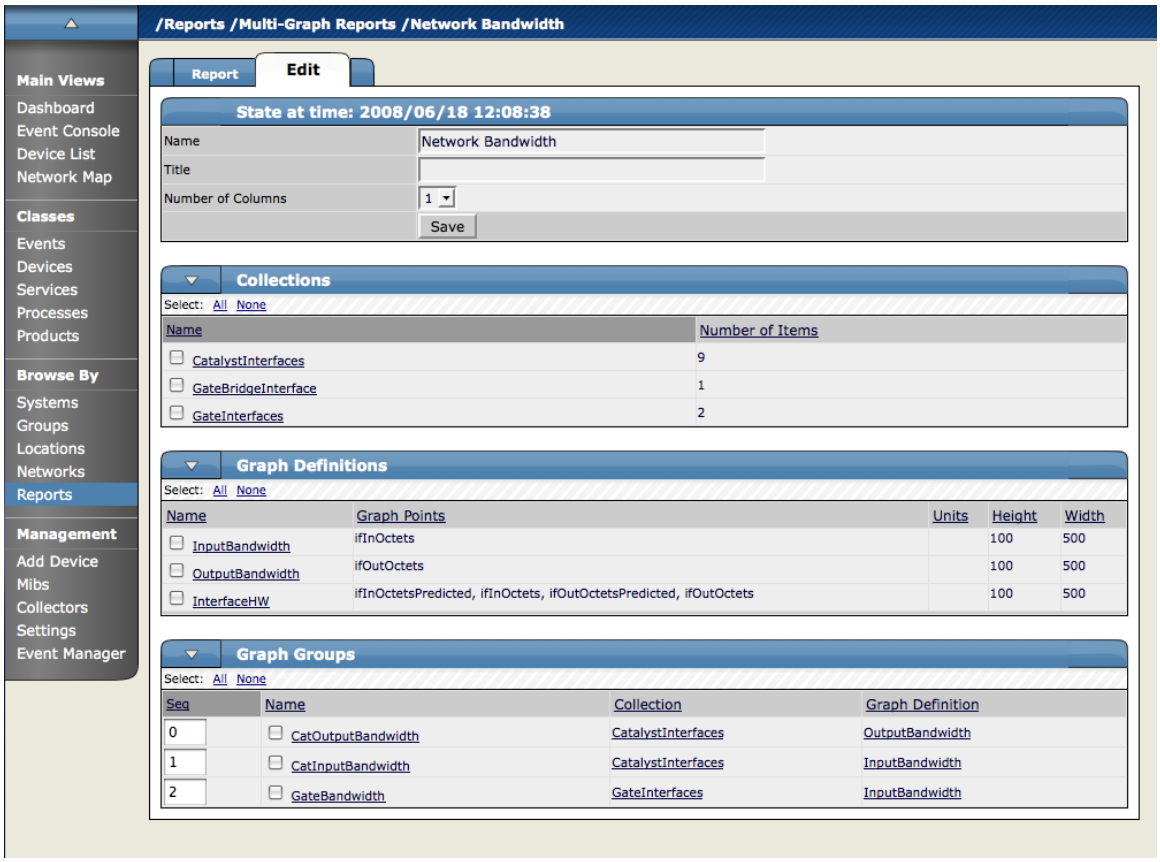
- Name - The name of the report is displayed at the top of the report.
- Title - This is displayed at the top of the printable version of the report.
- Number of Columns - The report will display the graphs in this many columns on the report.

Once you've created the MultiGraph Report there are three steps required to get graphs showing on the report:

1. Create a Collection which contains the Devices and/or Components you want to graph.
2. Create a Graph Definition that describes the graph(s) you want on the report.
3. Create a Graph Group which specifies the Collection and the Graph Definition you just created. The Method setting in the Graph Group lets you choose to have the graph drawn once for each Device/Component in the Collection or you can have the data from all the Devices/Components combined into a single graph.

These are just the minimal steps to getting a functional MultiGraph Report. You can create any number of Collections, Graph Definitions and Graph Groups in a single report. See the sections that follow for details on creating Collections, Graph Definitions and Graph Groups.

Figure 22.6. MultiGraph Report Edit Page



7.2. Collections

A Collection is a group of Devices and/or Components. A MultiGraph Report must have at least one Collection. Collections are listed in the Collections table on the report's Edit page. You can create a new Collection with the Add Collection menu item on that table then specifying a name in the dialog that appears.

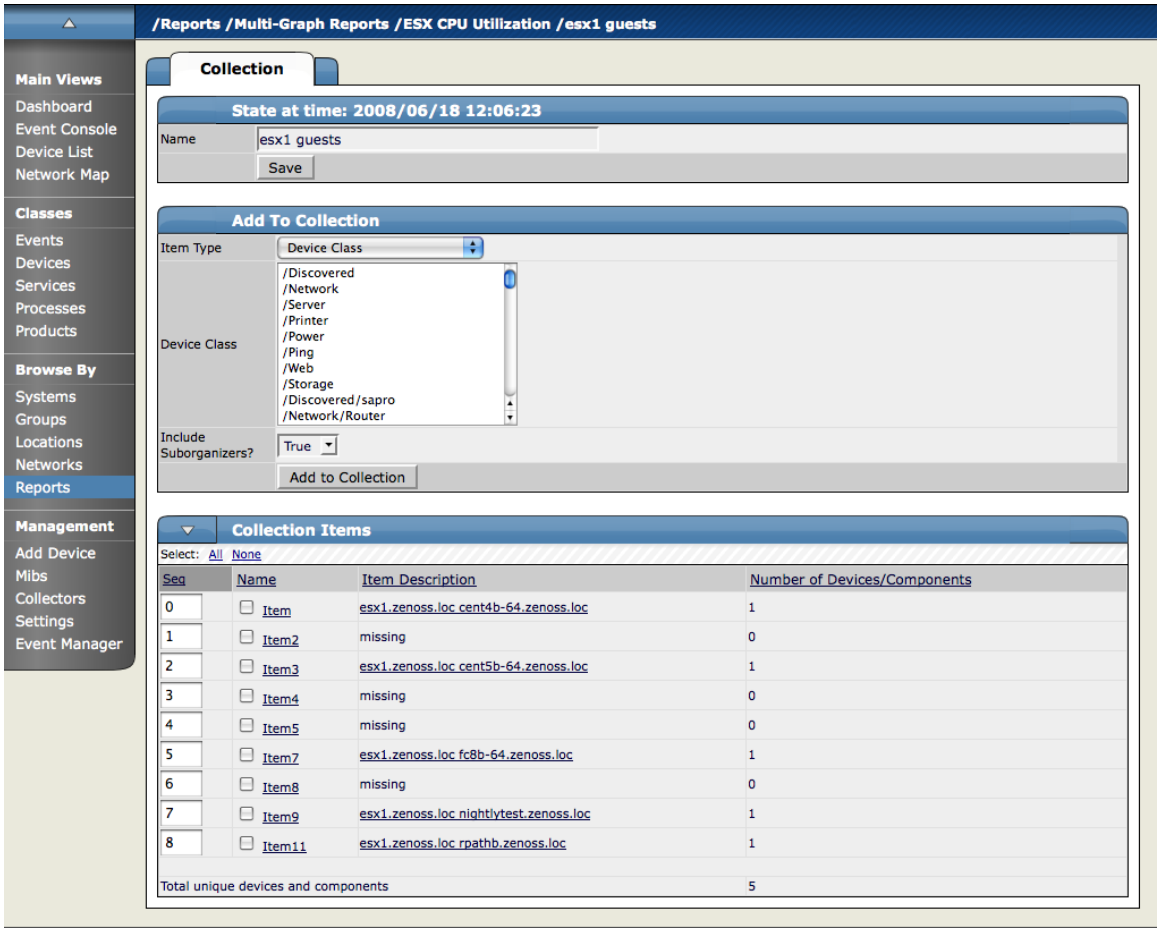
A Collection consists of one or more Collection Items. A Collection Item is a list of Device Classes, Systems, Groups, Locations or specific Devices and Components that should be included in this Collection. You can create as many Collection Items of the various types as you wish within a single Collection. The controls for creating Collection Items are in the Add To Collection table of the Collection edit page. The Item Type menu lets you select one of the following:

- Device Class/System/Group/Location - Selecting one of these options reveals a list of all organizers of that type. You can select one or more of the organizers to include in the Collection. By selecting True for the Include Suborganizers field the Collection will also include all organizers recursively beneath the ones you selected. These Collection Items are dynamic - when devices are added or removed from these organizers they will appear or disappear from the report. Clicking the Add to Collection button creates a new Collection Item for each of the selected organizers.

- Specific Device/Component - Selecting this type reveals a list of all Devices in Zenoss. You can use the Filter field to narrow this list by entering a full or partial Device name. Selecting one or more Devices will display a list of Component names that apply to one or more of the selected Devices. If you do not select any Components and click the Add to Collection button then a new Collection Item is created for each selected Device.

Once you have specified an Item Type and made your selection click the Add to Collection button to create the new Collection Item. It will be added to the list of Collection Items at the end of the page. Collection items can be deleted or reordered within this list. Order of the Collection Items determines the order that the graphs are drawn in or the order that data is drawn on a combined graph. See the section on Graph Groups for more details.

Figure 22.7. MultiGraph Report Collection

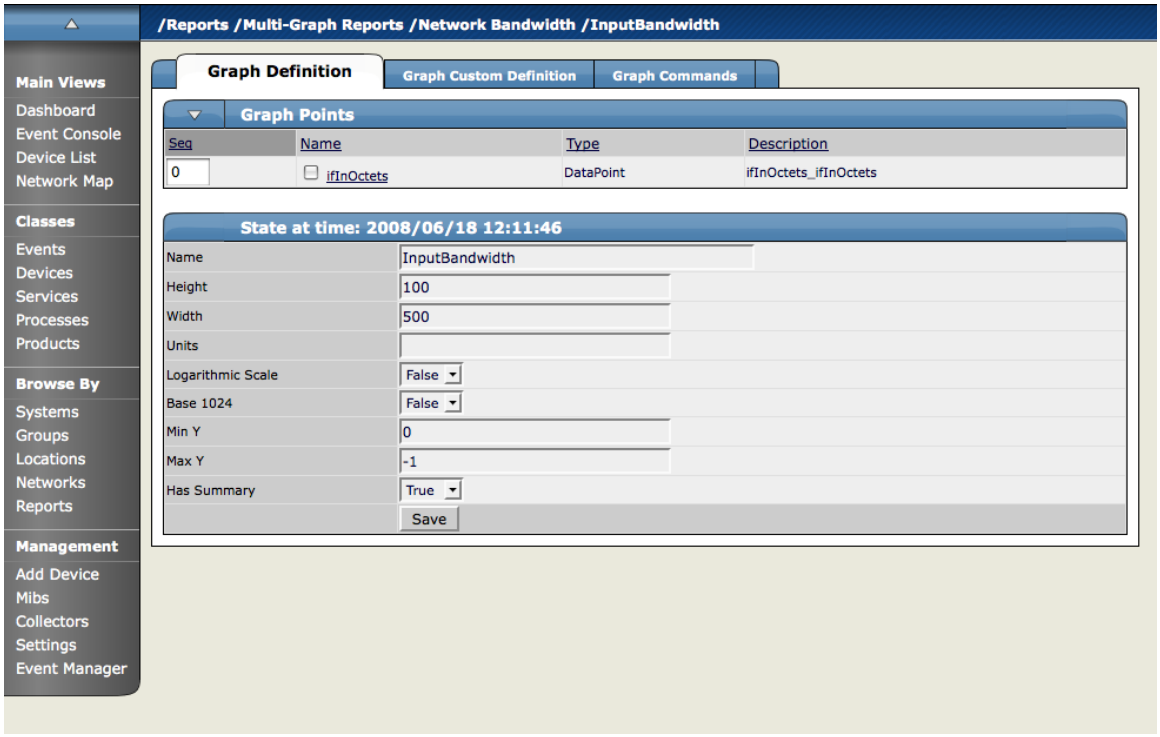


7.3. Graph Definitions

Graph Definitions in the context of MultiGraph Reports are very similar to those in Performance Templates. Settings on the Graph Definition itself define some basic parameters, then you add Graph Points to specify which data should be drawn. See the section on Graph Definitions in the Performance Chapter for details on creating Graph Definitions.

The most significant differences between Graph Definitions in the two contexts is how DataPoint Graph Points and Threshold Graph Points are added. When adding a DataPoint Graph Point to a Graph Definition within a Performance Template you can select from a list of DataPoints that are defined on that Template. But within the context of a MultiGraph Report there are no GraphPoint definitions to list. So instead of listing the available DataPoints, the DataPoint Graph Point dialog has a text field where you enter the name of the DataPoint. To make things easier the input has an auto-complete feature which knows the names of every DataPoint defined in Zenoss. This same situation is true with Threshold Graph Points.

Figure 22.8. MultiGraph Report Graph Definition



7.4. Graph Groups

Graph Groups are used to combine one Graph Definition with one Collection to produce graphs for the report. You must have at least one Graph Group or your report will have no graphs. You create a new Graph Group by selecting the Add Group menu item from the Graph Groups table menu. After entering a name for the Graph Group you are presented with the Graph Group edit page. There are 4 settings on this page:

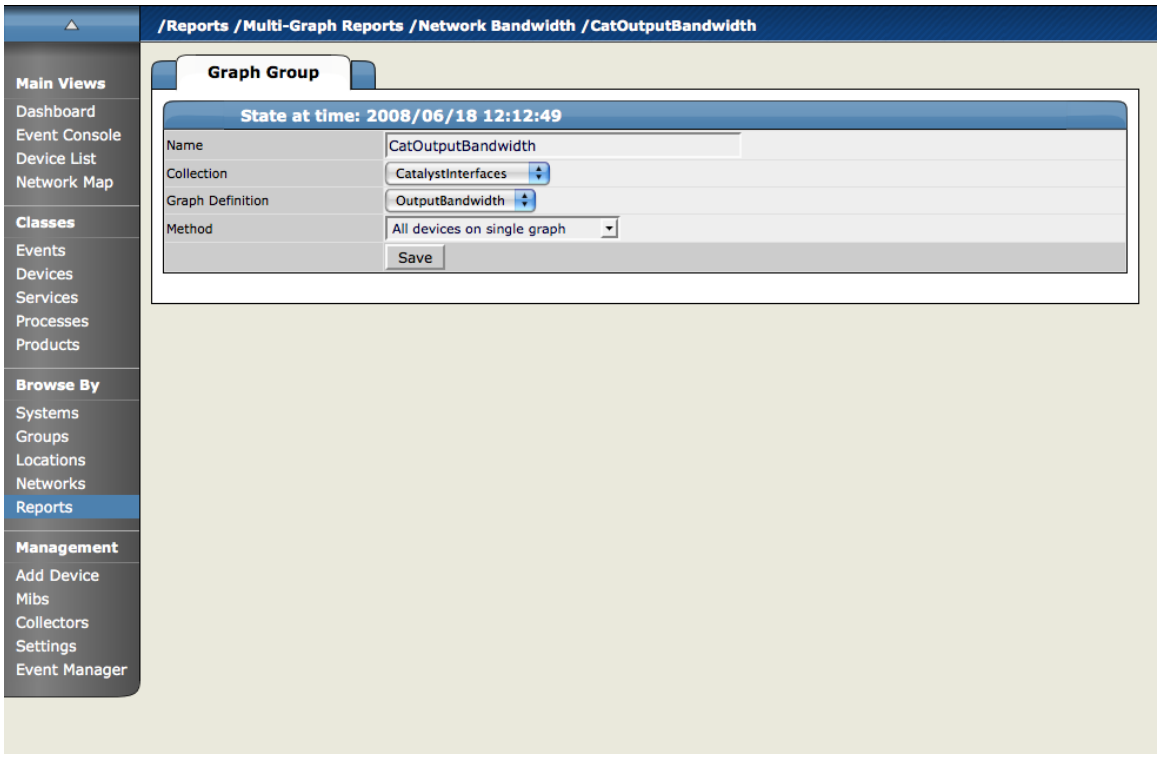
- Name - This is used to identify the Graph Group on the MultiGraph Report page. It does not appear anywhere on the actual report.

- Collection - Select one of the Collections that have been defined for this report.

- Graph Definition - Select one of the Graph Definitions that have been defined for this report.

- Method - There are two options for applying the Graph Definition you selected to the Collection that you selected: * Separate graph for each device: The Graph Definition will be used to draw one graph for each Device and Component listed in the Collection. The graphs will appear in the list in roughly the same order they are specified within the Collection. * All devices on a single graph: This draws one graph with the data from all the Devices/Components on it.

Figure 22.9. MultiGraph Report Graph Group



7.5. Graph Order

Graph Groups are drawn in the order they are listed on the MultiGraph Report edit page. You can change the order of the Graph Groups by editing the sequence number next to each then using the Re-Sequence Items menu item from that table's menu. If a Graph Group results in multiple graphs, the graphs are drawn in the order that the Collection Items are listed within the corresponding Collection. If a Collection Item specifies a Device organizer then the order of the Devices drawn from that Collection Item is indeterminate.

As with Graph Reports, if you have specified multiple columns for a report then the graphs are drawn left to right in that number of columns using as many rows as necessary.

8. Creating Custom Reports

There are a few ways to create reports of your own. You can create some reports through the Zenoss user Interface or using the Zope management Interface (ZMI).

8.1. Creating Custom Reports Using the ZMI

Zenoss Reports are written in python and templates are available through the Zope Management Interface (ZMI). To access the ZMI for a particular page, add a /manage to whichever report you are looking at. The ability to create individual reports is largely dependent on your ability to create python strings to get and display the information. More details on this will be forthcoming as we add more new reports and create more.

8.2. Create A Custom Device Report Example

Here are the steps for creating a Custom Device Report that will show device name, network address, and device serial number.

1. From the left navigation menu, select Reports.

2. From the Report Organizers list, click the Device Reports link.

   This is where you will be adding a new report.

3. From the bottom of the page enter a name for this custom report in the Add text box.

4. Click the Add button.

   The report you just created will appear in the Report list.

5. Click the name of the new report from the list.

   The Report detail page appears.

6. Click the Edit tab to define the report parameters.

7. Fill out the fields for this report as follows.

   - Name- The name you want to give to the report.
   - Title - The way the report is named in the display. Can be different from the name.
   - Path -The path in the hierarchy where the report will be stored.
   - Query - The actual query string for If you want to limit the report to just those devices that have a serial number, you can set the Query value to:

     here.hw.serialNumber != ""

   - Sort Column - What column you want to use to sort the report.
   - Sort Sense - The sense that the system uses to sort (For example asc is ascii)
   - Columns These columns are the actual data that be retrieved and displayed in the report.

     For example:

     getId – would get the name of any devices

     getManageIp – would get the IP addresses of the devices

     getHWSerialNumber – will grab serial numbers for device.

   - Column Names - You can add column names to make the column headers more descriptive.

     For the example columns above you could use the column names:

     Device

     Address

     Serial #

     The information that appears in the fields and how you actually get this information is found in the admin guide in the Tales Expressions appendix in the Device schema section.

8. Click the Save button.

9. Now click the Report tab at the top of the page.

   The new device report appears showing the devices that meet the criteria.

9. Using Reports to Help Troubleshoot Zenoss Daemons

This section will show you how to find and view certain reports that will aid you in troubleshooting Zenoss daemons.

From the Zenoss web interface, navigate to Reports. Follow the path to the various reports listed below to see the reports. The troubleshooting items on the right give you clues as to what to expect to find in the various reports.

Troubleshooting Items	Report Name	Where It Lives
zenmodeler issues	Model Collection Age	/Reports/Device Reports
Any internal Zenoss issues	All Heartbeats	/Reports/Event Reports/
zendisc errors, adding devices	New Devices	/Reports/Device Reports
Any alerting rule issues, will show all rules in the system	Notification Schedules	/Reports/User Reports
Summary of snmp status across the system including non-monitored	SNMP Status Issues	/Reports/Device Reports
Which devices are monitored and whether ping is turned on or off	Ping Status Issues	/Reports/Device Reports

10. Advanced Zenoss Reports

ZenEnterpriseReports, available only for Zenoss Professional and Enterprise versions, adds reports to the standard core reports. These are:

- Maintenance Windows Report - Shows all maintenance windows in the Zenoss instance. Use this report to track system up and down times.
- Organizer Availability Report - Give you the availability percentage of all network organizers in the system. This report can be filtered by Organizer, Event Class, Component, and Date.
- Systems Availability Report - Shows you systems availability in time and percentage. Can be filtered by date.

Chapter 23. General Zenoss Administration

1. Working with Zenoss from the Command Line

When working with Zenoss from the command line, you always want to be logged in as the user “zenoss”. If you log in as root you must become zenoss by issuing the command:

su - zenoss

2. Minimal Zenoss - ZEO and Zope

The Zenoss user interface can run without any other Zenoss processes running. In this mode, only ZEO and Zope are running.

This mode is useful for troubleshooting many Zenoss issues.

1. Stop all Zenoss processes.

   ```
   $ zenoss stop
   ```

2. Start the Zope object database.

   ```
   $ zeoctl start
   ```

3. Start Zope

   ```
   $ zopectl start
   ```

4. Go to the web UI and confirm that you can access the Dashboard.

5. Start the other Zenoss daemons.

3. Checking the Version of Zenoss

You can use the Version tab to check the version of Zenoss and all of the associated components. To access this information from the Navigation menu in the Dashboard, click Settings. Now click the Versions tab. The Versions tab appears.

This tab shows the version of the following Zenoss components:

- Zenoss
- Zope
- Database
- Twisted
- OS
- Python
- RRD
- SNMP

  This page also gives the Zope uptime.

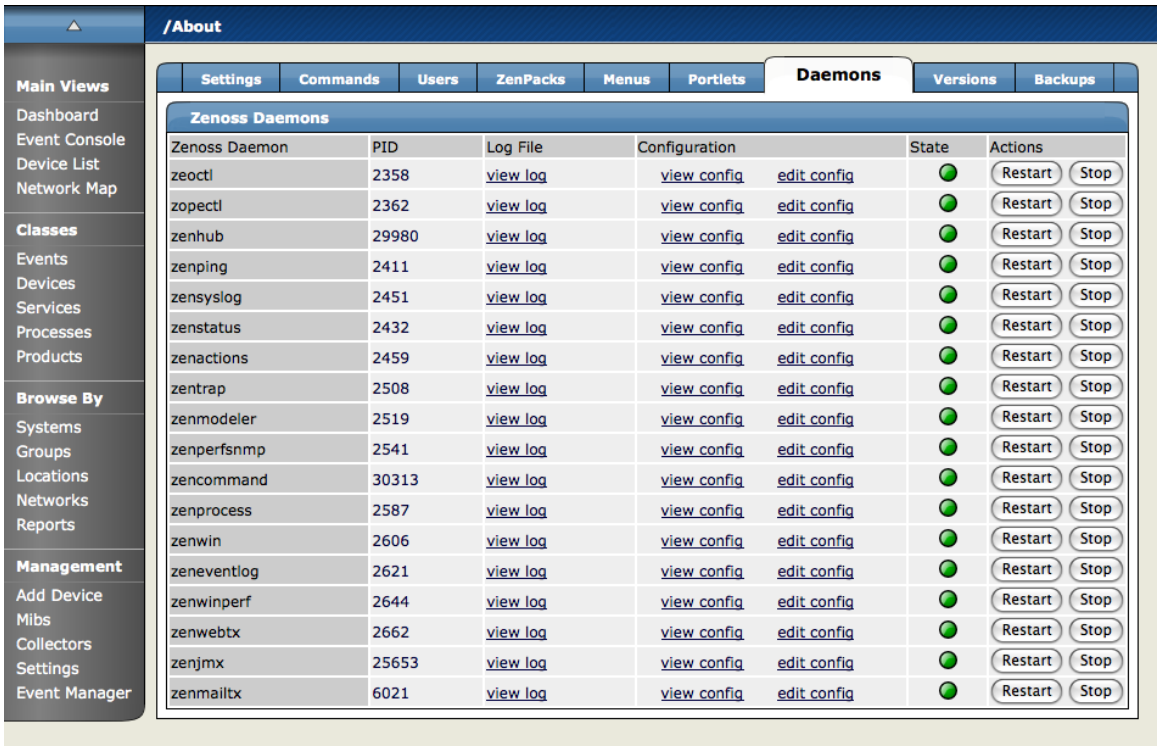
4. Checking for Zenoss Updates

Use the Settings page Versions tab to check for updates to the Zenoss software. You can have Zenoss to check daily or click the Check Zenoss Version Now button to check for an update. You can also see the last time a check for updates was made.

5. Starting and Stopping the Zenoss Daemons

You can use the Zenoss management console to start and stop the Zenoss daemons as well as check the status of the daemons. To access this information from the Navigation menu in the Dashboard, click Settings tab and then click the Daemons tab.

Figure 23.1. Settings Page - Daemons Tab



This tab is made up of a list of all of the Zenoss daemons, their process IDs (PIDs), a status indicator and then an Action area where you can restart a process or stop it. If a daemon is stopped, the Restart button changes to start.

6. Zenoss Daemon Commands and Options

All Zenoss daemons share some similar commands. You run these commands from the command line on the device where Zenoss is installed.

Each daemon has the following commands:

- run – Starts the daemon, but does not put it into the background. This command is useful when debugging problems.
- start – Starts the daemon running in the background (detached from the shell).
- stop – Stops the daemon.
- restart – Stops and then restarts the daemon. Occasionally, the start command will run before the daemon fully stops. If this happens, run the command again.

- status – Checks the status of a daemon. This command prints the current process number, if running.
- help – Displays a list of all options for the daemon.

6.1. Configuring Zenoss Daemons

You can configure any Zenoss daemon by adding key/value pairs to the $ZENHOME/etc/DAEMONNAME.conf file. Valid keys are the long option of any command line option. List these by using the daemons “help” command.

6.2. General Options for Zenoss Daemons

Command	Description
--version	show program's version number and exit
-h, --help	show this help message and exit
-vLOGSEVERITY,--logseverity=LOGSEVERITY	Logging severity threshold
--logpath=LOGPATH	override default logging path
-CCONFIGFILE,-- configfile=CONFIGFILE	config file
--uid=UID	user to become when running default:zenoss
-c, --cycle	Cycle continuously on cycleInterval from zope
-D, --daemon	Become a Unix daemon
--host=HOST	hostname of zeo server
--port=PORT	port of zeo server
-RDATAROOT, --dataroot=DATAROOT	root object for data load (i.e. /zport/dmd)
--cachesize=CACHESIZE	in memory cachesize default: 1000
--pcachename=PCACHENAME	persistent cache file name default:None
--pcachedir=PCACHEDIR	persistent cache file directory

6.3. zenhub Options

Command	Description
-x XMLRPCPORT, --xport=XMLRPCPORT	Port to listen to for XML RPC calls
--pbport=PBPORT	pb port
--passwd=PASSWORDFILE	Location the password file is stored
--workers=NUMBER	Allows zenhub to use subordinate processes to handle collector requests. Use to reduce the load on zenhub without resorting to a multiple-hub configuration. The value of NUMBER must be an integer greater than zero.

6.4. zenmodeler Options

Command	Description
--debug	don't fork threads for processing
--parallel=PARALLEL	number of devices to collect from in parallel
--cycletime=CYCLETIME	run collection every x minutes
--ignore=IGNOREPLUGINS	Comma separated list of collection maps to ignore
--collect=COLLECTPLUGINS	Comma separated list of collection maps to use
-pPATH, --path=PATH	start path for collection ie /NetworkDevices
-dDEVICE, --device=DEVICE	fully qualified device name ie: www.zenoss.com

Command	Description
-aCOLLAGE, --collage=COLLAGE	do not collect from devices whose collect date is within this many minutes
--writetries=WRITETRIES	number of times to try to write if a read conflict is found
-F, --force	force collection of config data (even without change to the device)
--portscantimeout=PORTSCANTIMEOUT	time to wait for connection failures when port scanning
-uUSERNAME, --user=USERNAME	Login username
-PPASSWORD, --password=PASSWORD	Login password
-tLOGINTRIES, --loginTries=LOGINTRIES	number of times to try login
-LLOGINTIMEOUT, --loginTimeout=LOGINTIMEOUT	timeout login expect statements
-TCOMMANDTIMEOUT, --commandTimeout=COMMANDTIMEOUT	timeout when issuing a command
-KKEYPATH, --keyPath=KEYPATH	Path to use when looking for keys
-sSEARCHPATH, --searchPath=SEARCHPATH	Path to use when looking for commands
-eEXISTENCETEST, --existenceTest=EXISTENCETEST	how to check for command
-rPROMPTTIMEOUT, --promptTimeout=PROMPTTIMEOUT	timeout when discovering prompt
-xLOGINREGEX, --loginRegex=LOGINREGEX	regex that will find the login prompt
-wPASSWORDREGEX, --passwordRegex=PASSWORDREGEX	regex that will find the password prompt
--enable	enter enable mode on a cisco device
--termlen	enter send terminal length 0 on a cisco device
--enablePause=ENABLEPAUSE	time to wait before sending enable command
--enableRegex=ENABLEREGEX	regex that will find the enable password prompt

NOTE: --ignore and --collect are mutually exclusive

6.5. zenperfsnmp Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSERNAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.6. zenperfxmlrpc Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSERNAME	username for zope server

Command	Description
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.7. zenprocess Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSERNAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration

6.8. zenping Options

Command	Description
--configpath=CONFIGPATH	path to our monitor config ie: /Monitors/StatusMonitors/localhost
--name=NAME	name to use when looking up our record in the dmd defaults to our fully qualified domain name as returned by getfqdn
--test	Run in test mode: doesn't really ping, but reads the list of IP Addresses that are up from /tmp/testping

6.9. zensyslog Options

Command	Description
--statcycle=STATCYCLE	Number of seconds between the writing of stats
--dmdpath=DMDPATH	zope path to our dmd /zport/dmd
--parsehost	try to parse the hostname part of a syslog HEADER
--stats	print stats to log every 2 seconds
--logorig	log the original message
--debug	debug mode no threads
--minpriority=MINPRIORITY	Minimum priority that syslog will accept
--heartbeat=HEARTBEAT	Number of seconds between heartbeats
--syslogport=SYSLOGPORT	Port number to use for syslog events

6.10. zenstatus Options

Command	Description
--configpath=CONFIGPATH	path to our monitor config ie: /Devices/Server
--parallel=PARALLEL	number of devices to collect at one time
--cycletime=CYCLETIME	check events every cycletime seconds

6.11. zenactions Options

Command	Description
--cycletime=CYCLETIME	check events every cycletime seconds
--fromaddr=FROMADDR	address from which email is sent
--zopeurl=ZOPEURL	http path to the root of the zope server

6.12. zentrap Options

Command	Description
--statcycle=STATCYCLE	Number of seconds between the writing of stats
-tTRAPPORT, --trapport=TRAPPORT	Port number for listening for traps

6.13. zencommand Options

Command	Description
-zZOPEURL, --zopeurl=ZOPEURL	XMLRPC url path for performance configuration server
-uZOPEUSERNAME, --zopeusername=ZOPEUSERNAME	username for zope server
--zopepassword=ZOPEPASSWORD	password used to login to the zope server
--zem=ZEM	XMLRPC path to an ZenEventManager instance
-dDEVICE, --device=DEVICE	Specify a specific device to monitor
--monitor=MONITOR	Specify a specific name of the monitor configuration
--parallel=PARALLEL	number of devices to collect at one time

7. Troubleshooting Zenoss Daemons

When Zenoss is having an issue, it will write stack traces into its log files. These log files are important for deciphering exactly what went wrong with the application. This section demonstrates how to generate a stack trace. This will generate a stack trace where you can see errors in the log.

1. While logged in as the zenoss user, stop all zenoss processes.
2. Shut Down MySQL.

```
$ sudo service mysqld stop
```

3. Start the Zope Object Database.

```
$ zeoctl start
```

4. Run zenperfsnmp in the foreground in debug mode.

```
$ zenperfsnmp run
```

5. Notice there are stack traces saying that the daemon can’t connect to zenoss.
6. Start Zope.

```
$ zopectl start
```

7. Go to dashboard and see the error message “Lost connection to Zenoss”. This error on the dashboard can appear for many reasons related to the loss of connectivity or some back end failure of the system.
8. Look for a more useful error in the zope log file $ZENHOME/log/event.log.

9. Restart mysqld.

```
$ sudo service mysqld start
```

10. Restart Zenoss.

```
$ zenoss start
```

11. Look at the end of all zenoss log files for any errors.

```
$ tail $ZENHOME/log/*.log | less
```

8. Automatic Startup in Linux Environments

Zenoss can be controlled entirely by the bin/zenoss script. To provide automatic startup in Linux environments, you must link Zenoss to the /etc/rc.d/init.d directory

```
$ ln -s $ZENHOME/bin/zenoss /etc/rc.d/init.d
```

It is important to either add $ZENHOME to the root environment or to the zenoss script itself.

9. Using Zenoss with a Remote MySQL Instance

If you are interested in using a remote MySQL database for Zenoss, set it up the same way you would set it up if you were using a native database. Then log into the management console.

1. First change the database address for the Backend Type for the mysql database.

   From the Zenoss Dashboard, in the Navigational menu, select Event Manager.

   The ZenEventManager page appears.

2. Change the Hostname field to the address of the MySQL database you want to use.

3. Restart Zenoss.

   The MySQL database you entered is now the default database.

10. Loading and Registering MIBs with Zenoss

The MIB loader is called zenmib. To load MIBs, first copy them to $ZENHOME/share/mibs/site. Then run zenmib using the command:

```
zenmib run mibfile
```

The MIB is now loaaded into Zenoss. You can see the MIBs you have loaded by using the left navigation menu and clicking the Mibs link.

10.1. Resolving MIB Dependencies

You loaded aaa.mib, and when you attempt to load bbb.mib, you get this error:

```
INFO:zen.zenmib:Unable to find a file providing the MIB AAA-MIB
```

To resolve the dependency, load them like this

```
zenmib run aaa.mib bbb.mib
```

Chapter 24. Backup, Recovery and Maintenance

1. Backup and Restore

Zenoss provides tools to backup the configuration and data from a Zenoss install and restore that information later. This can be useful in taking periodic snapshots of your install for backup purposes, moving your data from one Zenoss installation to another or restoring your setup and performing a fresh install of Zenoss. These are the specific items that are included in the backup and restore:

- The entire events database in mysql.
- The Zope database, which includes all devices, users, event mappings, etc.
- The $ZENHOME/etc directory, which contains config files for the zenoss daemons.
- The $ZENHOME/perf directory, which contains performance data,

The sections below describe in detail the backup and restore scripts and the options for controlling their behavior. Typical use of zenbackup looks like:

```
> zenbackup --save-mysql-access --file=BACKUPFILEPATH
```

Typical use of zenrestore looks like:

```
> zenrestore --file=BACKUPFILEPATH
```

Suggestions for a satisfying backup/restore experience:

- If you have the available disk space, tar and zip $ZENHOME before starting any backup or restore operation. This gives you a chance to recover in case something goes awry.
- Make sure Zenoss, including all daemons, is stopped before performing a restore.
- Using these tools to go from a newer version of Zenoss to an older version could be bad news and should really be avoided.
- If you use these tools to go from an older version of Zenoss to a newer version you should run zenmigrate after the restore.
- If restoring to a different zenoss installation than the backup is initially from, make sure file paths in the $ZENHOME/etc/*.conf files are appropriate for the new environment after you restore.

1.1. Backup Details

The script for backup is $ZENHOME/bin/zenbackup. If zenoss is running then you can run zenbackup without any arguments and a backup file will be placed in $ZENHOME/backups. zenbackup --help will give a full list of the available options. Some of the more interesting options are:

--dbname

This is the name of the mysql database zenoss uses to hold event data. By default this is "zenoss" but this can be specified when zenoss is installed. This value can be seen by looking at the database field on the Event Manager page in zenoss. If you don't specify --dbname then zenbackup will attempt to retrieve this information from zeo unless you specify --dont-fetch-args.

--dbuser, --dbpassword

These are the mysql username/password used to access the events database. If you don't specify --dbuser or --dbpassword then zenbackup will attempt to retrieve this information from zeo unless you specify --dont-fetch-args.

--dont-fetch-args

This instructs zenbackup not to attempt to get values for dbname, dbuser and dbpassword from zeo.

--file=FILE

Use --file to specify a location for the backup file. By default it will be named zenoss_<DATE>.tgz and placed in $ZENHOME/backups.

--stdout

This flag tells zenbackup to send the backup information to stdout instead of to a file. Incompatible with --verbose.

--save-mysql-access

This instructs zenbackup to save dbname, dbuser and dbpassword as part of the backup file so that zenrestore will have this information during a restore operation. Use this with caution as it means your backup files will contain a mysql username and password.

--no-eventsdb

Do not include the mysql events database as part of the backup.

-v, --verbose

Print progress messages. Incompatible with --stdout.

1.2. Backups Tab

Zenoss provides a simple web GUI for creating and managing backups. Navigate to Settings->Backups to view the Backups page. The Create New Backup section allows you to create a backup through the GUI. The options available are a subset of those available with the zenbackup command line tool (see above for details on the options.) Below that is the Backups section which lists all backup files in $ZENHOME/backups. You can delete one or more backup files by selecting them using the checkboxes and selecting the Delete Backup... menu item. Backup files can become large as your databases grow, so you may want to limit the number of backups you keep if drive space becomes an issue.

1.3. Remote Backups

Keeping backups on your zenoss server should help you recover if one of your databases becomes corrupt, your configuration becomes problematic, etc. It will probably not help you recover from a situation where your disk fails or any other failure that affects the system as a whole. For this reason it is advisable to keep at least one recent backup file on a different server, ideally at a different physical location.

1.4. Restore Details

The script for restoring zenoss from a backup is $ZENHOME/zenrestore. Make sure that zenoss is stopped before performing a restore. If you used the --save-mysql-access option when you created the backup file then you only need to specify --file when you run zenrestore. Otherwise you need to specify dbname, dbuser and dbpassword also.

--file

This is a backup file created with zenbackup You must specify either --file or --dir.

--dir

The path to an unzipped backup file. You must specify either --file or --dir.

--dbname

This is the name of the mysql database zenoss uses to hold event data. This database must exist before zenrestore is run. If there are any zenoss tables in the database they will be dropped by zenrestore before it restores the backed up tables and data. If you use a different dbname than was in use when the backup was created then after the restore you'll need to set the database name on the Event Manager page.

--dbuser, --dbpassword

These are the mysql username/password used to access the events database. If you don't specify --dbuser or --dbpassword then zenrestore will attempt to use values stored in the backup file if --save-mysql-access was used in creating it.

--no-eventsdb

Do not restore the mysql events database. If the backup file does not contain mysql events data then zenrestore will not modify your events database even if you do not specify --no-eventsdb.

-v, --verbose

Print progress messages.

1.5. Periodic Backups

Backing up your Zenoss data and configuration is a very good idea. $ZENHOME/bin/zenbackup can create backups that include your zeo database (devices, etc), RRD files (performance data), MySQL tables (events) and your Zenoss configuration files. See section **** for information on using zenbackup and zenrestore.

1.5.1. Pack ZEO Database

The Zeo database needs to be packed periodically to reclaim space. To do this you should set up a cron job that runs the following command weekly:

```
$ZENHOME/bin/zeopack.py -p 8100
```

1.5.2. Log Rotate Script

If your system uses logrotate to manage files put the following in /etc/logrotate.d/zenoss to manage Zenoss' log files:

```
/usr/local/zenoss/log/*.log {
weekly
rotate 2
copytruncate
}
```

1.5.3. Backing up the MySQL Event Backend

MySQL should be backed up following the MySQL manual.

Chapter 25. ZenPacks

1. Introduction to ZenPacks

A ZenPack is a package that adds new functionality to the Zenoss. A ZenPack may add Action Rules, Event Classes, Event Commands, User Commands, Service Classes, Data Sources, Graphs, Performance Templates, Reports, Model Extensions or Product Definitions. A ZenPack may also add new daemons and new UI features such as menus.

ZenPacks are a mechanism for extending and modifying Zenoss. This can be as simple as adding new Device Classes or Performance Templates or as complex as extending the data model and providing new collection daemons. ZenPacks can be distributed for installation on other Zenoss systems. Simple ZenPacks can be created completely within the Zenoss user interface while more complex ZenPacks require development of scripts or daemons in Python or another programming language.

For example, say you have developed a Performance Template for a new piece of hardware. You've created Data Sources for the OID's you think are worth monitoring, Thresholds to make sure some of these values stay within reasonable limits and several Graph Definitions to show this data graphically. Perhaps you've also created a new Device Class for this hardware. You can create a ZenPack to easily distribute your Performance Template and Device Class to other Zenoss administrators. This ZenPack can be entirely created from within the Zenoss user interface.

For another example, say you want to monitor a new piece of software running on one of your servers. You would like to monitor several performance metrics of this software, but they are available only via a programmatic API provided with the software. You could develop a new collector daemon to gather data via this API and provide it back to Zenoss. You might also create a new type of Data Source to provide configuration data for the new collector. Obviously this effort would require development skills and intimate knowledge of Zenoss not necessary for the previous example, but this functionality can still be distributed as a ZenPack.

1.1. Installing ZenPacks

ZenPacks are usually distributed as .egg files. Zenoss also supports .zip files, though support for this format will be removed in a future version of Zenoss. Either ZenPacks of either type can be installed from the command line on the Zenoss server or via the Zenoss user interface.

1.1.1. Installing via the Command Line

The following ZenPack command can be used from the command line to install ZenPack files. After installing or updating ZenPacks you need to restart Zope and ZenHub:

```
zenpack --install <filename>
zopectl restart
zenhub restart
```

If you have the source code for the ZenPack you can install directly from that rather than from a .egg or .zip file. The command is the same, you just specify the directory containing the source code. This copies the source code into either $ZENHOME/ZenPacks (for newer egg ZenPacks) or $ZENHOME/Products (for older style ZenPacks.)

```
zenpack --install <directoryname>
zopectl restart
zenhub restart
```

If you are developing a ZenPack you usually will want to maintain your source code outside of $ZENHOME/ZenPacks or $ZENHOME/Products. This is advisable for two reasons. First, if you issue a zenpack --remove command it will delete your code from either of those two locations and you would lose your files unless you had them backed up elsewhere. Second, if you are maintaining your source code in a version control system it is frequently more convenient to have the files reside elsewhere on the filesystem. Using the --link option you can install the ZenPack but have Zenoss use your code from its current location. Instead of installing your code in

$ZENHOME/ZenPacks or $ZENHOME/Products Zenoss will create a link in one of those locations that points to your source code directory.

```
zenpack --link --install <directoryname>
zopectl restart
zenhub restart
```

1.1.2. Installing via the User Interface

You can upload and install a ZenPack .egg or .zip file via the user interface. From the Settings->ZenPacks page choose the Install Zenpack... menu item. In the dialog that follows use the Browse button to select the .zip file from your local computer. When you click the OK button the file is uploaded to the Zenoss server and installed.

1.1.3. Installing All Core ZenPacks via RPM

The Zenoss Core ZenPacks, along with third party ZenPacks, are available for download individually from http://www.zenoss.com/community/projects/zenpacks/. Also on that page is a link to download an RPM that includes the most popular Core ZenPacks. To install via the Core ZenPacks RPM follow these steps:

1. Download the appropriate file from http://www.zenoss.com/community/projects/zenpacks/all-core-zenpacks
2. Make sure zeo is running (as zenoss user):

   ```
   zeoctl start
   ```

3. Install the rpm (as root user):

   ```
   rpm -ihv <rpm file>
   ```

4. Restart Zope and ZenHub:

   ```
   zopectl restart
   zenhub restart
   ```

1.2. Creating a ZenPack

When logged into Zenoss as an Administrator click on the Setting link and then on the ZenPacks tab. Select the "Create a ZenPack..." menu item. You will get a dialog asking for a name for your new ZenPack. The name must be of the form ZenPacks.<organization>.<identifier>, where organization is a name that identifies you or your organization and identifier is a string that represents the intent of your ZenPack. For example, ZenPacks.zenoss.HttpMonitor was created by zenoss to help monitor HTTP sites. Once you have entered a name click the save button. This creates both the ZenPack object in the database as well as a new directory in the filesystem $ZENHOME/ZenPacks/<your zenpack id>.

Many types of objects can be added to a ZenPack via the user interface. Some examples are:

- Device Classes
- Event Classes
- Event Mappings
- User Commands
- Event Commands
- Service Classes
- Device Organizers
- Performance Templates

Devices are the conspicuous omission from this list. Any individual Device is usually specific to a particular site and therefore not likely to be useful to other Zenoss users.

To add one of these database objects to a ZenPack navigate to that object and use the "Add to ZenPack..." menu item. Zenoss will present a dialog which lists all installed ZenPacks. Select the ZenPack to which you want to add this object and click the Add button. To view the objects that are part of a ZenPack navigate to the Settings page then the ZenPacks tab. Click on the name of the ZenPack and you will see a page that lists both the files and the objects that are part of this ZenPack. You can remove objects from the ZenPack by selecting the checkboxes next to them and using the "Delete from ZenPack..." menu item.

ZenPacks can contain items that are not zeo database items, such as new daemons, Data Source types, skins, etc. These are added to a ZenPack by placing them in the appropriate subdirectory within the ZenPack's directory. See the Core ZenPacks at http://www.zenoss.com/community/projects/zenpacks/ for examples of how to incorporate such items into your ZenPack. Further information regarding ZenPack development is available in the Zenoss Developers Guide.

Discussion regarding development of ZenPacks takes place on the zenoss-dev mailing list and forums: http://community.zenoss.com/forums/viewforum.php?f=3

1.2.1. Packaging and Distributing Your ZenPack

ZenPacks are usually distributed as .egg files. To create the installable .egg file for a ZenPack use the "Export ZenPack..." menu item in the page menu when viewing a ZenPack. The dialog that follows has two options. The first option simply exports the ZenPack to a file named <ZenPackId>.egg in the $ZENHOME/exports directory on the Zenoss server. The second option does the same but then downloads the exported file to your browser. Other Zenoss administrators can install this exported .egg file as described in the Installing ZenPacks section.

For information on how to make your ZenPack available on the zenoss.com site see http://www.zenoss.com/community/projects/zenpacks/how-to-contribute-a-zenpack

1.3. Removing a ZenPack

Warning: Removing a ZenPack can have unexpected consequences. For example, removing a ZenPack that installed a Device Class will remove both the Device Class and all Devices within that class. Also, before removing a ZenPack you should delete any Data Source of a type provided by the ZenPack. You should always perform a backup of your Zenoss data before removing a ZenPack. See Section 21.1 Backup and Restore for information on how to backup your Zenoss data.

2. Zenoss Core ZenPacks

2.1. ZenJMX ZenPack

Read this section to learn more about the ZenJMX ZenPack.

2.1.1. About ZenJMX

ZenJMX is a ZenPack that allows Zenoss to communicate with remote Java Management Extensions (JMX) agents. The ZenJMX ZenPack introduces a data source of type 'JMX'. A JMX data source allows you to define which JMX attributes should be monitored by Zenoss, as well as which operations you want Zenoss to invoke. The ZenPack also introduces the zenjmx daemon, which Zenoss uses to retrieve data from a JMX agent.

2.1.2. JMX Background

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

In Java 5, Sun introduced the Remote API for Java Management Extensions. This enhancement defines an RMI wrapper around a JMX agent and allows for independent client development. ZenJMX accesses remote JMX agents via the "Remote API for Java Management Extensions." It currently does not support local connections (provided via the temporary directory) to JMX Agents.

2.1.3. ZenJMX Capabilities

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

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

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

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

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

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

2.1.4. Single Value Attribute Calls

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

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

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

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

- MBean Name: java.lang:type=OperatingSystem
- Attribute Name: OpenFileDescriptorCount

- Data Points:
  - OpenFileDescriptorCount (type: gauge)

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

2.1.5. Complex-Value Attribute Calls

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

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

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

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

2.1.6. Operation Calls

Some management values need to be computed. These situations frequently arise when custom MBeans are deployed alongside an enterprise application. An MBean named "Accounting" might be deployed within an enterprise application that defines operations intended for operators or support staff. These operations might include methods such as "getBankBalance()" or "countTotalDeposits()".

ZenJMX has the ability to invoke operations, but there are some subtleties in how ZenJMX sends parameters to the JMX Agent and interprets the response.

2.1.6.1. Operation Calls: Scenario #1-No parameters, single return value

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

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

2.1.6.2. Operation Calls: Scenario #2-No parameters, multiple values returned in List format

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

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

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

2.1.6.3. Operation Calls: Scenario #3-No parameters, multiple values returned in Map format

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

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

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

2.1.6.4. Operation Calls: Scenario #4-Single parameter in polymorphic operation

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

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

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

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

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

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

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

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

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

2.1.6.5. Scenario #5: Multiple parameters in polymorphic operations

The above example describes how polymorphic behavior in Java functions and how method resolution can be provided by identifying the Class[] that represents the parameters passed to a method. The situation where multiple parameters are passed to a polymorphic operation is no different then the situation where a single parameter is passed to a polymorphic operation, except that the length of the Class[] and Object[] is > 1.

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

The following example demonstrates a configuration that passes 2 parameters to an MBean operation. The second parameter passed is a default value to return if no account can be located matching the first parameter.

- MBean Name: Application:Name=Accounting,Type=Accounting
- Operation Name: getBalances()
- Parameter Types: java.lang.String, java.lang.Integer
- Parameter Values: sbb552349999, 0
- Data Points:
  - balance (type: gauge)

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

2.1.7. ZenJMX Behavior

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

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

Lastly, ZenJMX heartbeats after each collect to ZenHub to let Zenoss know that ZenJMX is still alive and well.

2.1.8. Running the ZenJMX Daemon

The zenjmx daemon can be started by running "${ZENHOME}/bin/zenjmx start". Output from zenjmx is logged to ${ZENHOME}/log/zenjmx.log".

You can also run zenjmx in the foreground by running "${ZENHOME}/bin/zenjmx run". Additional parameters (such as --cycle or --cycleTime) can be provided after the "run" or "start" command. This is consistent with how other Zenoss daemons behave.

Most users will want to start the ZenJMX daemon in the background using the "${ZENHOME}/bin/zenjmx start" command and immediately follow that up with "tail -f ${ZENHOME}/log/zenjmx.log" to see what ZenJMX is doing.

2.1.9. Defining Custom JMX Data Sources

Custom JMX Data Sources allow system administrators to monitor any attribute or operation result accessible via a JMX call. ZenJMX creates a "JMX" Data Source and allows you to provide Object information, as well as authentication settings, and attribute/operation information. Determining which object and attribute names, as well as which operations to invoke, is the key to customizing ZenJMX.

Start off by creating a new Performance template at the /Device level. The performance templates associated with a device are accessible via the More->Templates link when looking at a device's page. Click the second down arrow and select "Add Template". Provide a descriptive name for the template, such as "JVM Values". After the template is created click on it. You should now be looking at the "JVM Values" performance template.

Click the down arrow next to Data Sources and select "Add Data Source". Provide a descriptive name of the JMX value you wish to retrieve. In our case we are interested in memory information so set the ID to "Heap Memory". Set the Type to JMX.

Enter the JMX Management Port (not necessarily the same as the listen port for your server) and Object Name. The Object Name is also referred to as the MBean name. Enter "java.lang:type=Memory" as the Object Name. If your JMX Agent requires authentication provide the user name and password.

Enter "HeapMemoryUsage" as the Attribute Name. Then add gauge data 425 points named "committed", "max", and "used". Click Save.

Lastly, add graphs that reference these new data points.

Please review "Interrogating an JMX Agent via JConsole" to learn how to determine the object name, attribute name, and data points that might be interesting in your application.

2.1.10. Attribute Path Input Value

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

If the result of traversing and extracting a value out of the nested open data is a single numeric value, then it is automatically mapped to the datapoint in the data source. However, if the value from the open data is another open data object, then the data point names from the datasource are used as indexes or keys to map values out of the open data.

The input value is a dot-separated string that represents a path through the object. Non-bracketed values are keys into CompositeData. Bracketed values are indexes into TabularData.

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

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

Example

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

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

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

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

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

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

2.1.11. Enabling Remote JMX Access

Each application server has a slightly different process for enabling remote JMX Access. It's best to consult with your application server for specific instructions. We've included instructions for a few commonly used configurations below.

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

2.1.11.1. Remote JMX Access for the standard JVM

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

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

export JAVA_OPTS
```

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

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

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

2.1.11.2. Remote JMX Access for Tomcat

The same JAVA_OPTS approach can be used to enable remote access to Tomcat MBeans. Set the JAVA_OPTS variable as illustrated above and then execute the "./catalina.sh start" command in ${TOMCAT_HOME}/bin.

Note that Tomcat 6.0.14's catalina.sh does not process the "stop" command properly when the JAVA_OPTS variable is set. We recommend using 2 separate shells when troubleshooting JMX problems in Tomcat: one for starting Tomcat (with the JAVA_OPTS variable set) and a different one for stopping Tomcat (where the JAVA_OPTS variable is not set).

2.1.11.3. Remote JMX Access for JBoss

JBoss uses the JAVA_OPTS approach for enabling remote access to beans. However, it requires some additional properties. To set up your JAVA_OPTS for use in JBoss see the following code segment:

```
JAVA_OPTS="-Dcom.sun.management.jmxremote.port=12345"
JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote.authenticate=false"
JAVA_OPTS="${JAVA_OPTS} -Dcom.sun.management.jmxremote.ssl=false"
JAVA_OPTS="${JAVA_OPTS} -Djboss.platform.mbeanserver"
JAVA_OPTS="${JAVA_OPTS} -Djavax.management.builder.initial=org.jboss.system.server.jmx.MBeanServerBuilder
export JAVA_OPTS
```

When you start JBoss via the run.sh you must also pass the "-b 0.0.0.0" argument:

```
cd ${JBOSS_HOME}/bin
./run.sh -b 0.0.0.0
```

JMX actually uses 2 separate ports for MBean access: one is used for initial connection handling and authentication, and the other is used for RMI access. During the handshake between a JMX Client and the JMX Agent the agent tells the client the IP address and port number for the RMI registry. By default JBoss sets the IP address to 127.0.0.1. This works when the JMX client and the JMX agent reside on the same device, but it won't work in a distributed environment.

By passing the "-b 0.0.0.0" argument you instruct JBoss to bind to all available network ports, and this results in the JMX Agent's handshaking logic using a network reachable address when informing clients of the RMI registry hostname and port.

2.1.11.4. Remote JMX Access for WebLogic

JSR-160 standardized remote access to JMX Agents, and allows for any client to connect to an JMX Agent using classes packaged with the JDK. WebLogic versions prior to 9.0 required clients to use a WebLogic JMX client library that used a proprietary protocol to interact with the JMX Agent. You'll need to make sure you're running WebLogic 9.0 or higher in order to monitor it via ZenJMX.

If you're new to WebLogic and have not set up a domain and server you'll need to run the startWLS.sh script located in ${BEA_HOME}/wlserver_10.0/server/bin. If you don't have the Terminal I/O package installed you can set the JAVA_OPTIONS variable to the following value:

```
JAVA_OPTIONS="-Dweblogic.management.allowPasswordEcho=true"
export JAVA_OPTIONS
```

Provide a user name and password to start WebLogic. Note that WebLogic requires a password that is at least 8 characters long. Wait for WebLogic to generate a configuration and start up. Shut down WebLogic and restart it with remote JMX access enabled.

To enable remote JMX access set the following variable:

```
JAVA_OPTIONS="-Dcom.sun.management.jmxremote.port=12347"
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.authenticate=false"
JAVA_OPTIONS="${JAVA_OPTIONS} -Dcom.sun.management.jmxremote.ssl=false"
export JAVA_OPTIONS
```

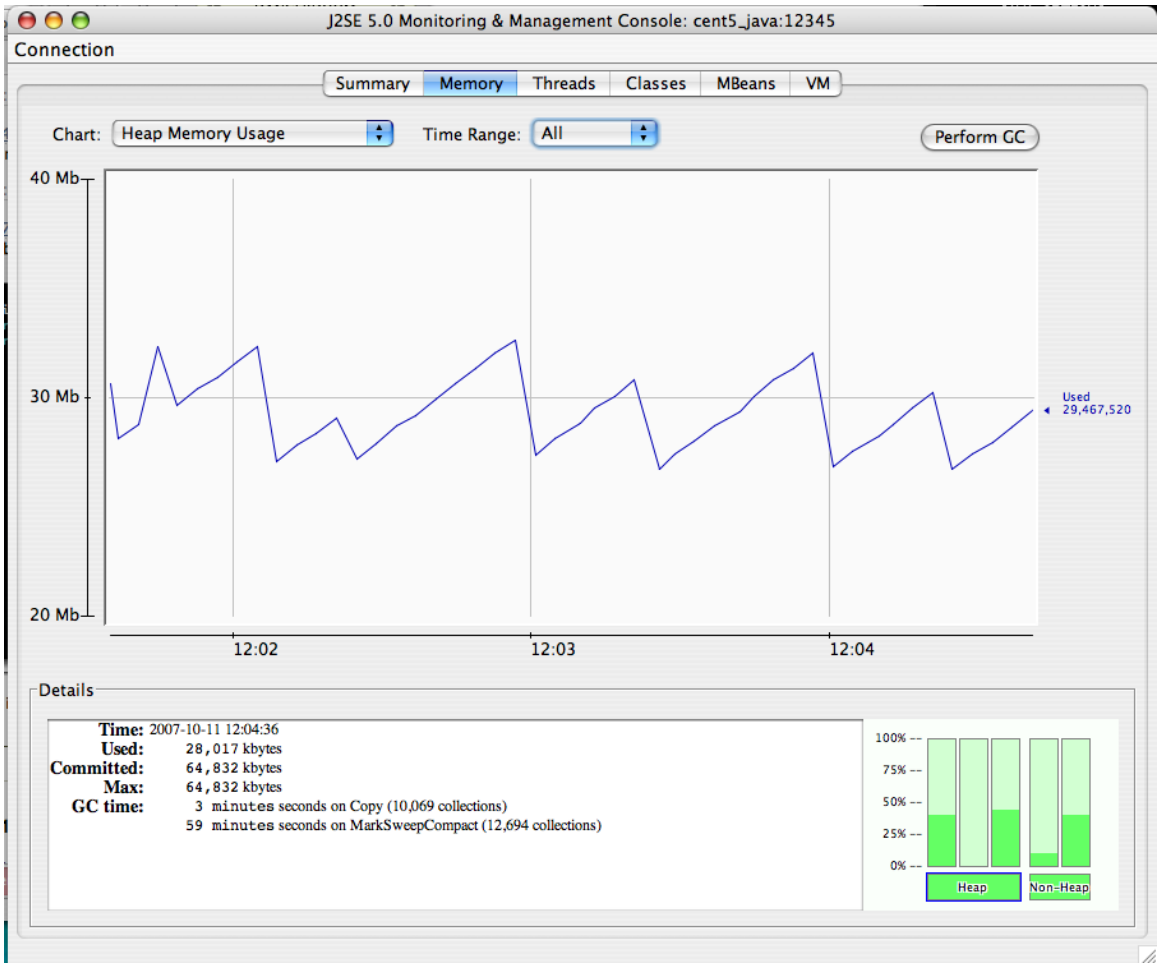
Then re-run the ./startWLS.sh script. JConsole can then communicate with the server on port 12347.

2.1.12. Interrogating an JMX Agent via JConsole

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

JConsole also provides a graph that shows memory, thread, and class usage over time. The scale of the graph can be adjusted so that a system administrator can examine a specific period of time, or can zoom out to view a longer range picture of usage. Unfortunately, JConsole can only produce graphs that show usage while JConsole was running. Administrators cannot look back in time to a point where the JVM was running but JConsole was not monitoring the JVM.

Figure 25.1. JMX Heap Graph

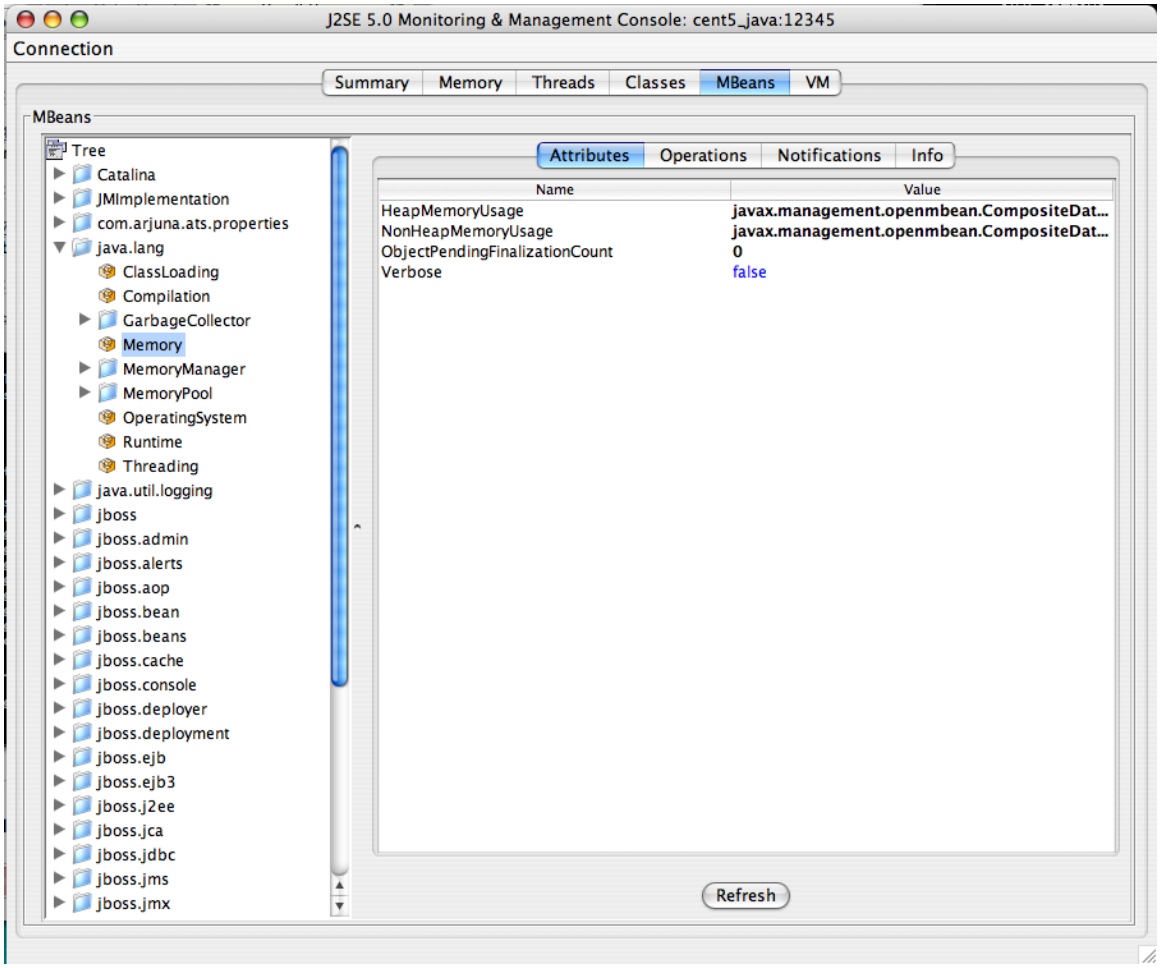


The MBean tab along the top of JConsole provides an interactive method for examining MBean values. After clicking on the MBean tab a panel will be displayed with a tree on the left hand side. The tree contains a hierarchical list of all MBeans deployed in the JVM.

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

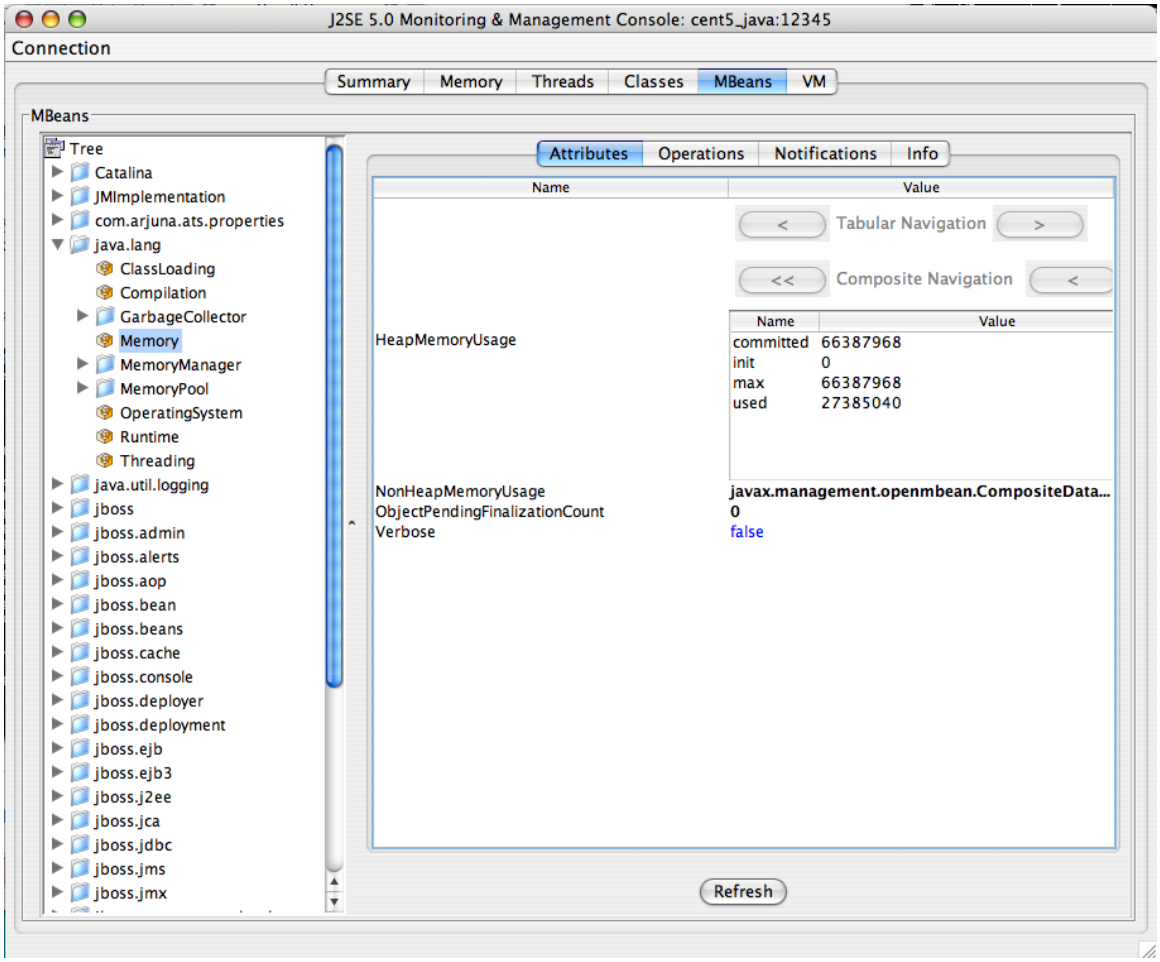
To get started expand the java.lang node in the Tree. This will expose several MBeans as well as additional folders. Click on the Memory MBean and observe how the right hand side of the panel is populated with information about the Memory MBean.

Figure 25.2. Memory MBean



MBeans can contain attributes and operations. MBeans can also fire notifications to observers, but that's beyond the scope of this document. The attributes tab lists all of the attributes in the first column and their values (or a clickable attribute type) in the second column. In the case of Memory the HeapMemoryUsage is a Composite attribute, otherwise referred to as a "complex-value attribute" in Zenoss. Double click the "javax.management.openmbean.CompositeDataSupport" type and you will see multiple attributes appear. The show the amount of committed, maximum, and used memory sizes for the heap.

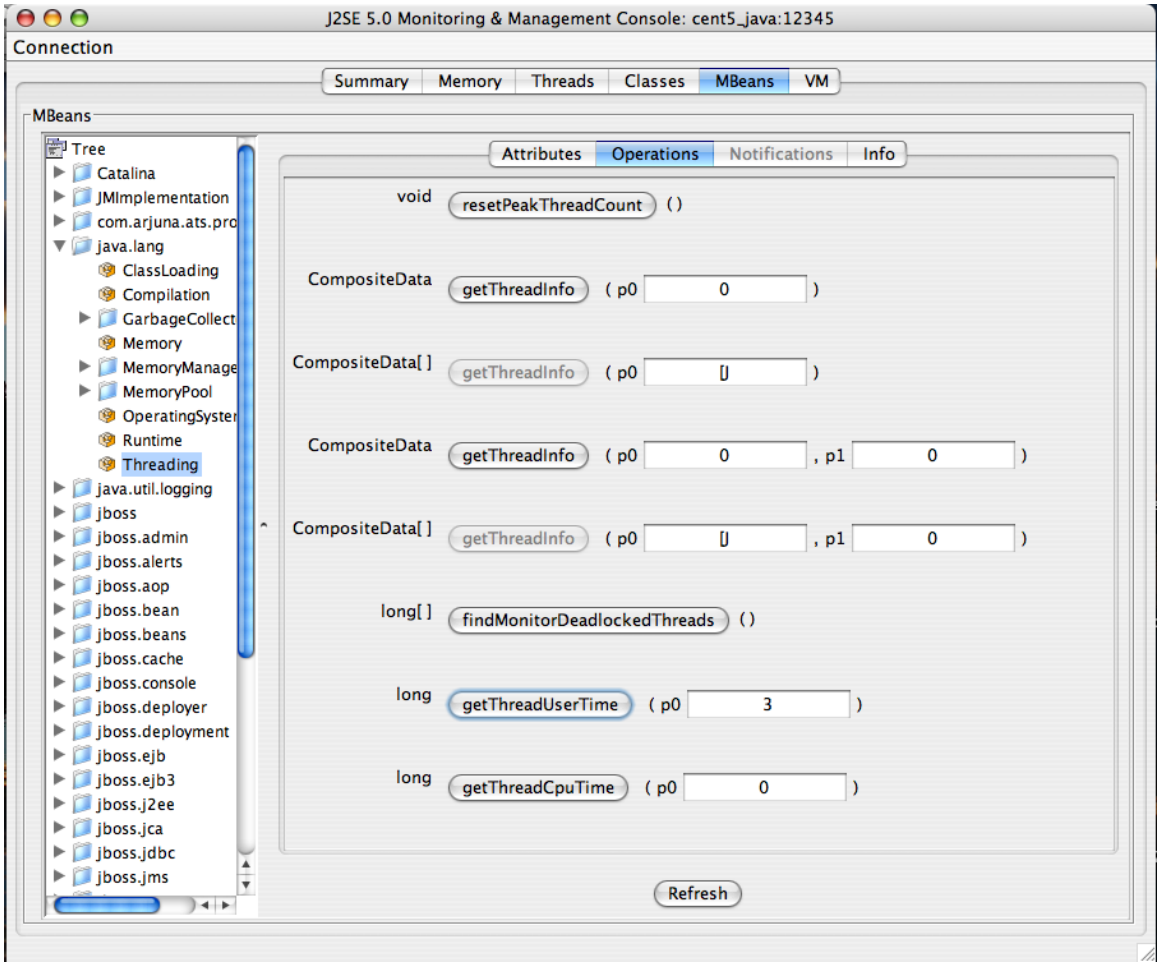
Figure 25.3. Memory MBean Expanded



The unique name of the MBean can be viewed by clicking on the Info tab. The first value is MBean Name and it's value in the case of Memory is: "java.lang:type=Memory". Note that there isn't a standardized way to name MBeans and application server vendors do it differently.

You can also examine operation information by clicking on the Operations tab. These are methods that JConsole can remotely invoke on an MBean that will result in some value being computed or some state changing in the application. The Threading MBean has several operations that can be invoked that return information. Click on the java.lang package and then click on the Threading operation. Lastly, click on the Operations tab. Methods like "getThreadUserTime" are invocable.

Figure 25.4. Operations Tab



Test the "getThreadUserTime" method by changing the p0 parameter to 1 and clicking the "getThreadUserTime" button. A dialog window will be raised that displays the amount of CPU user time thread #1 has used. Try adjusting the parameter to different values to observe the different CPU times for the threads.

2.1.13. Installing ZenJMX

Step #1: Install the ZenJMX ZenPack

ZenJMX is installed using the "zenpack" command. As the zenoss user, run the following command:

```
${ZENHOME}/bin/zenpack --install /path/to/ZenJMX-1.0.0-el5-i386.zip
```

Step #2: Install Sun Java Runtime Environment

to run correctly, ZenJMX requires Sun JRE Version 5.0 or higher. Make sure that after you install Sun's JRE you update your PATH such that the "java" executable works. You can test this using the command "which java" - if it returns a fully qualified path, then you have successfully installed Java.

2.1.13.1. Installing ZenJMX on an appliance

ZenJMX and Sun's JRE is installed using a conary command. As root, run the following command:

```
conary update --resolve group-zenjmx
```

2.2. ApacheMonitor ZenPack

ApacheMonitor provides a method for pulling performance metrics from the Apache HTTP Server (http://httpd.apache.org/) directly into Zenoss without requiring the use of an agent. This is accomplished by utilizing the standard mod_status module that comes with version 1 and 2 of the HTTP server.

The following metrics will be collected and graphed for the Apache HTTP Server.

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

Follow these steps to setup your HTTP server so that it will allow Zenoss to access the server status.

1. On the Apache server, find your httpd.conf file. This is normally located in /etc/httpd/httpd.conf or /etc/httpd/conf/httpd.conf. Other locations are possible depending on your operating system and setup.
2. Turn the ExtendedStatus option on in the httpd.conf file. This option will typically be commented out. You can enable it by uncommenting it.

   ```
   #ExtendedStatus on
   ```

   ... becomes ...

   ```
   ExtendedStatus on
   ```

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

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

   ... becomes ...

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

4. Save the httpd.conf file with these changes then restart httpd. This can be accomplished with following command.

   ```
   apachectl restart
   ```

Once your Apache HTTP Server is configured to allow Zenoss to access the extended status, you can add Apache monitoring to the device within Zenoss by simply binding the Apache template to the device.

1. Navigate to the device in the Zenoss web interface.
2. Click the device menu, choose More then Templates.
3. Click the templates menu, choose Bind Templates.
4. Ctrl-click the Apache template from /Devices/Server to choose it.
5. Click OK.

You will now be collecting the Apache HTTP Server metrics from this device.

2.3. DellMonitor ZenPack

DellMonitor provides custom modeling of devices running the Dell OpenManage agents. It also contains hardware identification for Dell proprietary hardware. The information is collected through the SNMP interface.

The following information is modeled:

- Hardware Model 11
- Hardware Serial Number
- Operating System
- CPU Information (socket, speed, cache, voltage)
- PCI Card Information (manufacturer, model)

2.4. HPMonitor ZenPack

HPMonitor provides custom modeling of devices running the HP/Compaq Insight Management Agents. It also contains hardware identification for HP proprietary hardware. The information is collected through the SNMP interface.

The following information is modeled.

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

2.5. MySqlMonitor ZenPack

MySqlMonitor provides a method for pulling performance metrics from the MySQL database server (http://www.mysql.com/) directly into Zenoss without requiring the use of an agent. This is accomplished by utilizing the MySQL client library to connect to the database remotely.

The following metrics will be collected and graphed for MySQL server.

- Command Statistics (SELECT, INSERT, UPDATE, DELETE)
- Select Statistics (Scan, Range Check, Range Join, Full Join)
- Handler Statistics (Keyed & Unkeyed Reads, Writes, Updates, Deletes)
- Network Traffic (Received & Sent)

Follow these steps to setup your MySQL server to allow Zenoss to read performance data from the system tables.

1. Connect to the MySQL database using the MySQL client.

   ```
   mysql -u root
   ```

   ... or if there is a MySQL root password ...

   ```
   mysql -u root -p
   ```

2. Create a user for Zenoss to use. The username "zenoss" is recommended.

   ```
   mysql> CREATE USER zenoss IDENTIFIED BY 'zenossPassword';

   Query OK, 0 rows affected (0.00 sec)
   ```

3. Edit the zMySqlRootPassword zProperty for the device(s) within Zenoss that you intend to monitor MySQL on.

4. Bind the MySQL template to the same device(s).

Pay particular attention to the MySQL Version 5+ setting in the datasource. If you are monitoring pre-v5 installations of MySQL then be sure to set this value to False. If you are monitoring both pre v5 and v5+ installations then create two templates, one for MySQL installations earlier than v5 and another for those after.

2.6. NtpMonitor ZenPack

ZenPacks.zenoss.NtpMonitor monitors the offset between system time and a target ntp server's (Network Time Server) time.

2.6.1. Components

2.6.1.1. Event Classes

- /Events/Status/Ntp

2.6.1.2. Performance Templates

- NtpMonitor Datasource

2.6.1.3. Classes

- NtpMonitorDataSource

2.6.1.4. Datapoints

- offset Graphs: offset Skins: editNtpMonitorDataSource.pt

2.6.2. DataSource Class Options

The following Datasource options are available with this ZenPack:

- port - Port number to send ntp request (default: 123)

  Note

  The check_ntp plugin does not seems to use this option even though its usage notes that it exists.

- warning - Response time to result in warning status (seconds)
- critical - Response time to result in critical status (seconds)
- timeout - Seconds before connection times out (default: 10)

2.6.3. Example

The NTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.NNTPMonitor EGG.
2. Navigate to the device (or Device Class) that has a NTP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the NTPServer template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the NTPMonitor template and change the Use SSL and Port options as needed.

2.7. LDAPMonitor ZenPack

The ZenPacks.zenoss.LDAPMonitor monitors the response time of an LDAP server in milliseconds.

2.7.1. Components

2.7.1.1. Performance Templates

- LDAPServer

2.7.1.2. Thresholds

- SlowLDAP - 5000ms threshold for Warning event on response time; escalated after 6
- BrokenLDAP - 30000ms threshold for Critical event on response time

2.7.1.3. DataSource

- ldap using LDAPMonitor; creates time DataPoint

2.7.1.4. Graph Definitions

- LDAP Server Response Time

2.7.1.5. zProperties

- zLDAPBaseDN - The Base Distinquished Name to use for the response time check. Defaults to dc=zenoss,dc=com. Should be customized to fit the customer's LDAP server requirements.
- zLDAPBindDN - The Distinquished Name to use for binding to the LDAP server, if authentication is required. Defaults to nothing.
- zLDAPBindPassword - The password to use for binding to the LDAP server, if authentication is required. Defaults to nothing.

2.7.2. DataSource Class Options

- LDAP Server - defaults to ${dev/id}
- Base Distinguished Name - defaults to ${here/zLDAPBaseDN}
- Bind Password - defaults to ${here/zLDAPBindPassword}
- Use SSL - defaults to unchecked
- Port - defaults to 389

2.7.3. Additional Configuration

The three zProperties (zLDAPBaseDN, zLDAPBindDN and zLDAPBindPassword) should be configured to allow a valid connection to the LDAP server. The zLDAPBaseDN should be configured to issue a valid search to the LDAP server; normally this property would be set to the root organization within the server. If authentication is required on the server then zLDAPBaseDN and zLDAPBasePassword must also be configured.

2.7.4. Example

The NNTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.LDAPMonitor EGG.
2. Navigate to the device (or Device Class) that has a LDAP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the LDAPServer template to the device.

   The next cycle of zencommand will collect offset data.

4. Choose the zProperties option from the Server menu.

5. Change the zLDAPBaseDN zProperty to be the appropriate base distinguished name for your LDAP server. Typically this will be the organization's domain name, e.g. dc=foobar,dc=com. Check with your LDAP Administrator for more options.

6. Change the zLDAPBindDN and zLDAPBindPassword if authentication is required for the LDAP server. For example, cn=joe,dc=foobar,dc=com. Check with your LDAP Administrator for the format in your organization.

7. If your LDAP Servers require SSL or a custom port then navigate to the LDAP Server template, choose the ldap DataSource and then change the Use SSL and Port fields as needed.

8. Validate your configuration by running zencommand and observing that the check_ldap or check_ldaps command correctly connects to your LDAP server:

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

2.8. RPCMonitor ZenPack

ZenPacks.zenoss.RPCMonitor monitors the availability of an RPC server.

2.8.1. Components

2.8.1.1. Performance Templates

- RPCServer

2.8.1.2. DataSource

- rpc using RPCMonitor

2.8.2. DataSource Class Options

- RPC Server - defaults to ${dev/id}
- RPC Command - defaults to ${here/zRPCCommand}
- Port - defaults to 0 (to use PortMapper)

2.8.3. Additional Configuration

The zProperty zRPCCommand must be configured to indicate which RPC command should be tested for availability.

2.8.4. Example

The RPCMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.RPCMonitor EGG.

2. Navigate to the device that has a RPC Server that needs to be monitored (add the device if necessary).

3. Choose the Templates options from the Server menu and the the Bind Template option and bind the RPCServer template to the device.

   The next cycle of zencommand will collect offset data.

4. Choose zProperties from the Server menu and set the appropriate RPC command to test in the zRPCCommand zProperty, e.g. nfs, ypserv, etc.

5. If a custom port is being used, navigate to the RPCMonitor template and change the Port option as needed.

6. Validate your configuration by running zencommand and observing that the check_rpc command correctly connects to your RPC server:

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

2.9. IRCMonitor ZenPack

The ZenPacks.zenoss.IrcdMonitor monitors the number of users connected to an IRC server.

2.9.1. Components

2.9.1.1. Event Classes

- IRCD

2.9.1.2. Performance Templates

- IRCDMonitor

2.9.1.3. DataSource Classes

- IRCDMonitorDataSource

2.9.1.4. Graph Definitions

- number

2.9.1.5. Skins

- editIRCDMonitorDataSource.pt

2.9.2. DataSource Class Options

- hostname - Name or IP address of remote server to connect to.
- port - Port number to connect to remote IRC server (default: 6667)
- warning_num - Number of connections to
- critical_num - Response time to result in critical status (seconds)

2.9.3. Example

The IrcdMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.IrcdMonitor EGG.
2. Navigate to the device (or Device Class) that has a IRC Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the IRCDMonitor template to the device.

   The next cycle of zencommand will collect offset data.

2.10. JabberMonitor ZenPack

ZenPacks.zenoss.JabberMonitor monitors the response time of devices running a Jabber Server.

2.10.1. Components

2.10.1.1. Event Classes

- Jabber

2.10.1.2. Performance Templates

- JabberMonitor

2.10.1.3. DataSources

- JabberMonitorDataSource

2.10.1.4. DataPoints

- time

2.10.1.5. Graph Definitions

- time

2.10.2. DataSource Class Options

- Timeout (seconds) - Seconds before connection times out (default: 60)
- Port - port number default 5223
- Send String - string to send to the server : default <stream:stream to='${dev/id}' xmlns:stream='http://etherx.jabber.org/streams'>
- Expect String - string to expect in server response : default <stream

2.10.3. Example

The JabberMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.JabberMonitor EGG.
2. Navigate to the device (or Device Class) that has a Jabber Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the the Bind Template option and bind the Jabber template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the JabberMonitor template and change the Use SSL and Port options as needed.

2.11. DigMonitor ZenPack

DigMonitor monitors the response time of DNS lookups for devices running an Dig Server.

2.11.1. Components

2.11.1.1. Event Classes

- /Events/Status/DNS

2.11.1.2. Performance Templates

- NameServer

2.11.1.3. Datasource Classes

- DigMonitorDataSource

2.11.1.4. Datapoints

- time

2.11.1.5. Graph Definitions

- DNS Response Time

2.11.1.6. Skin Templates

- editDigMonitorDataSource

2.11.2. DataSource Class Options

- DNS Server: the nameserver against which the dig command should be run
- Port: The port on which the nameserver is listening
- Record Name: The name of the record you wish to look up
- Record Type: The DNS record type (e.g. A, MX, CNAME)

2.11.3. Example

The DigMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.DigMonitor EGG.
2. Navigate to the device (or Device Class) that has a Dig Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the DigMonitor template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the DigMonitor template and change the Use SSL and Port options as needed.

2.12. FTPMonitor ZenPack

The ZenPacks.zenoss.FTPMonitor ZenPack monitors connection response time to an FTP server.

2.12.1. Components

2.12.1.1. Event Classes

- Ftp

2.12.1.2. Performance Templates

- FtpMonitor

2.12.1.3. DataSource

- FtpMonitor

2.12.1.4. Datapoint Thresholds

- time

2.12.1.5. Graph Definitions

- time

2.12.2. Datasource Class Options

- Timeout - Seconds before connection times out (default: 60)
- Port - port to connect to FTP server (default 21)
- Send String - string to send to server
- Expect String - string to expect in server response
- Quit String - String to send server to initiate a clean close of the connection
- Refuse - Accept tcp refusals with states ok, warn, crit (default: crit)
- Mismatch - Accept expected string mismatches with states ok, warn, crit (default: warn)
- Max Bytes - Close connection once more than this number of bytes are received
- Delay - Seconds to wait between sending string and polling for response
- Certificate (minimum days for which a certificate is valid)
- Use SSL - Use SSL for the connection.
- Warning response time (seconds) - Response time to result in warning status (seconds)
- Critical response time (seconds) - Response time to result in critical status (seconds) What metrics (datapoints) does it collect? response time of the ftp connection

2.12.3. Example

The FTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.FTPMonitor EGG.
2. Navigate to the device (or Device Class) that has a FTP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the the Bind Template option and bind the Jabber template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the FTPMonitor template and change the Use SSL and Port options as needed.

2.13. NNTPMonitor ZenPack

The ZenPacks.zenoss.NNTPMonitor monitors the response time of an NNTP server in milliseconds.

2.13.1. Components

2.13.1.1. Performance Templates

- NNTPServer

2.13.1.2. DataSource

- nntp using NNTPMonitor; creates time

2.13.1.3. DataPoint Thresholds

- SlowNNTP - 5000ms threshold for Warning event on response time; escalated after 6 BrokenNNTP - 30000ms threshold for Critical event on response time

2.13.1.4. Graph Definitions

- NNTP Server Response Time

2.13.2. DataSource Class Options

- NNTP Server - defaults to ${dev/id}
- Port - defaults to 119

2.13.3. Example

The NNTPMonitor template must be bound to the DeviceClass or Device you wish to monitor.

1. Install the ZenPack.zenoss.NNTPMonitor EGG.
2. Navigate to the device (or Device Class) that has a NNTP Server you want to monitor (add the device if necessary).
3. Choose the Templates options from the Server menu and the Bind Template option and bind the NNTPServer template to the device.

   The next cycle of zencommand will collect offset data.
4. If a SSL or a custom port is being used, navigate to the NNTPMonitor template and change the Use SSL and Port options as needed.
5. Validate your configuration by running zencommand and observing that the check_nntp or check_nntps command correctly connects to your LDAP server:

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

Chapter 26. Zenoss Global Dashboard (ZenGlobe)

1. About Zenoss Global Dashboard (ZenGlobe)

Note

This feature is available only for Zenoss Enterprise.

The Zenoss Global Dashboard allows you to display, in one location, the data from several Zenoss instances. Each Zenoss instance monitors a number of devices on various networks and then the ZGB combines all of that data and makes for easy viewing of the data. You can view the data based on collector, network, location and report on the device in combination of the groupings you create. This enhances Zenoss' scalability and extends its performance and reporting capabilities.

ZenGlobe is a standalone Web server.

2. Additional Requirements

The Zenoss Global Dashboard requires at least Python 2.3.

3. Installation

Follow these steps to download and install the Zenoss Global Dashboard:

1. Download the latest version of the Zenoss Global Dashboard.

2. Extract the tarball and change to the created directory using the following commands:

   ```
   tar xzf zenglobe-1.0.tar.gz
   ```

   ```
   cd ZenGlobe
   ```

3. Prior to starting up the first time, ZenGlobe needs to know the port it should bind to and the Zenoss instance it should use for authentication. Run:

   ```
   sudo ./zenglobe configure
   ```

4. Enter the port to which you want ZenGlobe to bind. Make sure you have nothing else listening at that port.

When asked, enter the hostname of a running Zenoss instance that you want ZenGlobe to use for authentication. You can change this setting later, but in order to log in to ZenGlobe the first time you will need to use the username and password of a user from this Zenoss instance. Anyone with a login to this Zenoss instance will be able to view the ZenGlobe dashboard, but only the “admin” user will be able to edit settings.

1. Start ZenGlobe using the command:

   ```
   sudo ./zenglobe start
   ```

2. Check to make sure that ZenGlobe has started by accessing from your browser:

   ```
   http://[ZenGlobe machine hostname]:[port]/
   ```

   The ZenGlobe login screen will appear. If you do not get the ZenGlobe login screen retrace your steps and try again.

3.1. Setting Up Users on Remote Zenoss Instances

For security reasons, in order to gather data from remote Zenoss instances, ZenGlobe must have a login to those instances. By default, this is set to be zenglobe:zenglobe; however, it is a good idea to reconfigure ZenGlobe to use a more secure username and password combination.

You will also need to create a user on each Zenoss instance to be monitored using the same username and password.

1. In a browser, navigate to the Zenoss instance you wish to monitor, click “Settings” in the left nav pane, then select the “Users” tab..
2. From the table menu, select “Add New User.” Enter the username that you want ZenGlobe to use to log in to all Zenoss instances (e.g., “zenglobe”). You may leave the “Email” field blank. Click “OK.”
3. Repeat these steps for each Zenoss instance to be monitored, then configure ZenGlobe to log in as the remote users you have just created.

4. Setup and Configuration

1. Log in to the Zenoss Global Dashboard as the admin user.

   (Only the admin user can modify ZenGlobe options.)

2. Click the "Configure..." link in the top bar. The configuration box will slide down.

   The options in this Configuration box are as follows:

   - Zenoss Servers - The list of hostnames of the Zenoss instances ZenGlobe will monitor.
   - Remote Login - The username and password ZenGlobe will use to access the remote Zenoss instances. By default, it is set to zenglobe:zenglobe. Follow the instructions in 4.1.1 to set up matching users on each Zenoss instance to be monitored.
   - URL Template - The template ZenGlobe will use to build the URL by which it accesses monitored Zenoss instances. If you run your Zenoss instances on a different port, or serve them behind Apache with rewritten URLs, you will need to update this value to reflect that change.

3. You may also reset the port and authentication server using the same command line option you used when initially configuring ZenGlobe:

```
sudo ./zenglobe configure
```

5. Using the Zenoss Global Dashboard

The global dashboard collects event and heartbeat data from the monitored Zenoss servers and aggregates them into a single view.

- Device Issues - A list of all devices with serious events. The "Server" column displays the Zenoss server that monitors that device.
- Zenoss Sub-Systems - A list of monitored Zenoss instances. An event rainbow is displayed for each instance, showing a summary of active events.
- Zenoss Issues - A list of heartbeat issues from monitored Zenoss instances. Refer to the Zenoss admin guide for instructions on how to handle these.

6. Viewing a Zenoss Instance from the Zenoss Global Dashboard

The drop-down list on the extreme left of the top bar can be used to view monitored Zenoss instances from within ZenGlobe. Simply select the hostname of an instance from the list to load it below. For security's sake, you will

need to log in to the remote instance. You may return to the ZenGlobe dashboard at any time by selecting it from the same drop-down list.

7. Logging out of the Zenoss Global Dashboard

To logout of the Zenoss Global Dashboard, just click "Logout" in the top bar to end your ZenGlobe session.

Chapter 27. Zenoss High Availability

1. Overview

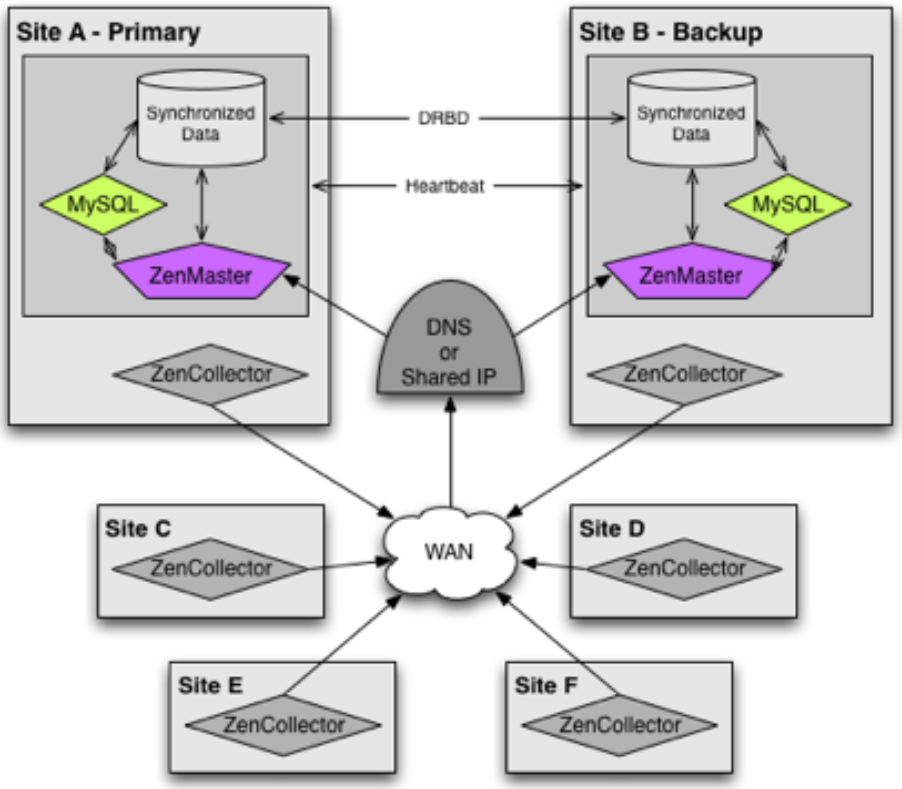
Note

This feature is available only with Zenoss Enterprise.

This chapter will describe the method for setting up two Zenoss master servers that operate in an active/passive configuration to provide a single, highly-available master. In addition, a collector will be running on each master to monitor the local network. The solution relies on DRBD (Distributed Replicated Block Device), which is a distributed storage system for the Linux platform. It consists of a kernel module, several userspace management applications and some shell scripts. DRBD bears similarities to RAID 1, except that it runs over a network.

The full deployment can conceptualized according to the following diagram. The number of non-master sites is not limited. The collectors can be configured to choose the current active master either by a dynamically router shared IP (BGP, etc.) or through a DNS lookup of a record that must be changed manually when the master site is failed over.

Figure 27.1. Zenoss High Availability



2. Conventions

The following conventions will be used to reference different elements required during the installation and configuration.

- zenoss-master: Hostname of the current Zenoss master server or IP.
- zenoss-siteA: Hostname of the Zenoss server in site A.

- 10.0.1.20: IP address of the Zenoss server in site A.
- zenoss-siteB: Hostname of the Zenoss server in site B.
- 10.0.2.20: IP address of the Zenoss server in site B.
- cluster: Highly available combination of zenoss-siteA and zenoss-siteB.
- /dev/sdb1: Block-level device that will be replicated between site A and B.
- /r0: File system mount point for the /dev/sdb1 block device.

3. Prerequisites

Prior to beginning your installation and configuration you should make sure that the following prerequisites are met.

- Architecture of both master systems is identical (i686 or x86_64)
- Operating system of both master systems is identical (RHEL5 or CentOS5)
- A spare partition that has been sized large enough to hold the MySQL and Zope database exists in an unpartitioned and unmounted state on both masters.
- A standard installation of Zenoss Enterprise has already been performed on both of the masters. This should be a clean installation that has not been further configured or used.

4. Installation of Packages

The following RPM packages must be installed to support the disk replicate and failover mechanism. These are available as part of the standard repositories for RHEL5 and CentOS5.

- drbd82
- kmod-drbd82
- heartbeat

5. Stopping Services

Stop the MySQL and Zenoss services with the following commands on zenoss-siteA and zenoss-siteB.

```
service zenoss stop
service mysqld stop
```

6. Setting up DRBD for File System Replication

We will now use those extra partitions we setup to replicate the data that needs to be shared.

1. Setup /etc/drbd.conf with the following contents on zenoss-siteA and zenoss-siteB.

```
global {
  usage-count yes;
}
common {
  protocol C;
}
resource r0 {
  syncer {
```

```
    rate 500K;
  }
  on zenoss-siteA {
    device    /dev/drbd0;
    disk      /dev/sdb1;
    address   10.0.1.20:7788;
    meta-disk internal;
  }
  on zenoss-siteB {
    device    /dev/drbd0;
    disk      /dev/sdb1;
    address   10.0.2.20:7788;
    meta-disk internal;
  }
}
```

2. Create the DRBD meta-data on zenoss-siteA and zenoss-siteB.

```
drbdadm create-md r0
```

3. Start the drbd service on zenoss-siteA and zenoss-siteB.

```
service drbd start
```

4. Seed the replicated partition on zenoss-siteA only.

```
drbdadm -- --overwrite-data-of-peer primary r0
```

5. Format the replicated disk on zenoss-siteA only.

```
mkfs.ext3 /dev/drbd0
```

7. Relocating Shared Data to the Replicated Disk

There is very little data we need replicated, but whatever we do want to be replicated should be moved/linked to the /r0 directory that will be replicated by DRBD.

1. Mount the replicate disk on zenoss-siteA.

```
mkdir /r0
mount /dev/drbd0 /r0
mkdir /r0/zenoss
chown zenoss:zenoss /r0/zenoss
```

2. Create initial MySQL data on zenoss-siteA.

```
service mysqld start
service mysqld stop
```

3. Move MySQL data to /r0 on zenoss-siteA.

```
mv /var/lib/mysql /r0/mysql
chown –R mysql:mysql /r0/mysql
```

4. Configure MySQL to find its data on /r0 on zenoss-siteA and zenoss-siteB.

```
ln –s /r0/mysql /var/lib/
```

5. Configure Zenoss to find its data on /r0 on zenoss-siteA and zenoss-siteB.

   In /opt/zenoss/etc/zeo.conf find “$INSTANCE/var/Data.fs” and change it to “/r0/zenoss/Data.fs”

6. Create initial Zenoss data on zenoss-siteA.

```
service zenoss start
service zenoss stop
```

8. Setting up Heartbeat for Resource Migration

Heartbeat is part of the Linux HA project and is responsible for performing the real work of a active/passive high-availability setup. It will mount /r0 on whichever node is currently the master as well as stopping and starting the appropriate servers such as MySQL and Zenoss.

1. Create /etc/ha.d/ha.cf on zenoss-siteA with the following contents.

```
keepalive 2
deadtime 30
warntime 10
initdead 120
bcast eth0
node zenoss-siteA
node zenoss-siteB
crm yes
```

2. Create /etc/ha.d/authkeys on zenoss-siteA with the following contents.

```
auth 1
1 sh1 yourSecretKey
```

3. Set restrictive permissions on the authkeys file on zenoss-siteA.

```
chmod 0600 /etc/ha.d/authkeys
```

4. Copy the ha.cf and authkeys files from zenoss-siteA to zenoss-siteB.

5. Start the heartbeat service on zenoss-siteA and zenoss-siteB.

```
service heartbeat start
```

9. Setting up Heartbeat Resources

Now that we have a replicated disk and a cluster setup we need to tell heartbeat which resources we want it to control. This is all done on zenoss-siteA exclusively. The heartbeat service will replicate the changes automatically.

1. Setup constraints to make sure our replicated disk is always in place before the services that require it are started. This is done by creating a file called constraints.xml (location is irrelevant) with the following contents.

```
<constraints>
  <!-- /r0 can't be mounted until drbd is promoted -->
  <rsc_order id="r0_after_drbd"
             from="rg_zenoss" action="start"
             to="ms_zenoss" to_action="promote"
             type="after"/>

  <!-- /r0 and drbd always have to be colocated -->
  <rsc_colocation id="r0_on_drbd" to="ms_zenoss"
                  to_role="master" from="rg_zenoss"
                  score="INFINITY"/>
</constraints>
```

2. Load this file into heartbeat which will replicate it across the cluster.

```
cibadmin –U –x constraints.xml
```

3. Create a resources.xml file with the following content to setup all resources required for a Zenoss cluster.

```
<resources>
  <master_slave id="ms_zenoss">
    <meta_attributes>
      <attributes>
        <nvpair name="notify" value="yes"/>
      </attributes>
    </meta_attributes>
```

```
    <primitive class="ocf" type="drbd" provider="heartbeat"
               id="drbd_shared">
      <instance_attributes>
        <attributes>
          <nvpair name="drbd_resource" value="r0"/>
        </attributes>
      </instance_attributes>

      <operations>
        <op name="monitor" interval="29s" timeout="10s"
            role="Master"/>
        <op name="monitor" interval="30s" timeout="10s"
            role="Slave"/>
      </operations>
    </primitive>
  </master_slave>

  <group id="rg_zenoss">

    <!-- Mount /r0 -->
    <primitive class="ocf" type="Filesystem"
               provider="heartbeat" id="fs_shared">

      <instance_attributes>
        <attributes>
          <nvpair name="device" value="/dev/drbd0"/>
          <nvpair name="directory" value="/r0"/>
          <nvpair name="type" value="ext3"/>
        </attributes>
      </instance_attributes>
    </primitive>

    <!-- Take 192.168.1.229 IP -->
    <primitive class="ocf" type="IPaddr2"
               provider="heartbeat" id="ip_shared">
      <instance_attributes>
        <attributes>
          <nvpair name="ip" value="192.168.1.229"/>
          <nvpair name="nic" value="eth0"/>
        </attributes>
      </instance_attributes>
    </primitive>

    <!-- MySQL Service -->
    <primitive class="lsb" type="mysqld"
               provider="heartbeat" id="mysqld"/>

    <!-- Zenoss Service -->
    <primitive class="lsb" type="zenoss"
               provider="heartbeat" id="zenoss">
      <operations>
        <op id="zenoss_start" name="start" timeout="1m"/>
        <op id="zenoss_stop" name="stop" timeout="1m"/>
        <op id="zenoss_status" name="monitor"
            timeout="1m"/>
      </operations>
    </primitive>

  </group>
</resources>
```

4. Load this file into the cluster.

```
cibadmin –U –x resources.xml
```

10. Troubleshooting

The following are some commands that will greatly help you understand, operate and troubleshoot your cluster.

- crm_mon will show you the current state of all cluster resources and nodes.
- cat /proc/drbd will show you the current state of your replicated disk.
- crm_resource –r rg_zenoss –M –H <node-hostname> will migrate the rg_zenoss resource group to the specified node.
  - You should always run crm_resource –r rg_zenoss –U to unmigrate a resource afterwards or the resource will never be able to fail back to the node you migrated away from.

11. References

Much more in-depth information on Linux clustering can be found at the following sites.

- Linux HA (http://www.linux-ha.org/)
- DRBD Users Guide (http://www.drbd.org/users-guide/ch-configure.html)
- CentOS DRBD Howto (http://wiki.centos.org/HowTos/Ha-Drbd)

Chapter 28. Distributed Collector

1. What is Distributed Collector?

Distributed Collector, available only with Zenoss Enterprise, allows you to deploy additional performance collection and event monitoring daemons to the Zenoss server or other servers. This allows you to:

- Distribute processor, disk, and network load between multiple servers
- Collect performance and events from networks that cannot be reached by the Zenoss server
- Configure more than one set of monitoring settings and assign devices to the configuration that makes most sense

When you first install Distributed Collector, Zenoss is configured with one hub and one collector. A collector is a set of collection daemons, on the Zenoss server or another server, that share a common configuration. That configuration contains values such as number of seconds between SNMP collection cycles, default discovery networks, and maximum number of zenprocess parallel jobs.

Each collector has its own copy of each of the Zenoss collection daemons. For example, Zenoss initially contains collection daemons named zenperfsnmp, zenprocess, and zenping. If you create a new collector named My2ndCollector, then the system creates new daemons named My2ndCollector_zenperfsnmp, My2ndCollector_zenprocess, and My2ndCollector_zenping. This collector could run on the main Zenoss server or on a remote server.

You cannot delete the initial hub and collector setup by Distributed Collector (both named localhost).

In addition to collectors, Distributed Collector allows you to set up new hubs. A hub represents an instance of the zenhub daemon, which is the daemon through which all collector daemons communicate with the object database and event database. All collectors must belong to exactly one hub; however, a hub may have many collectors associated with it. All hubs (and indirectly all collectors) refer to the same object and event databases.

1.1. Restrictions and Requirements

- Servers hosting remote hubs or collectors must be the same operating system and hardware architecture as the Zenoss server. For example, if the Zenoss server if running RedHat Enterprise Linux v5 on Intel 32-bit hardware, then hubs and collectors can be deployed only to other RHEL 5 32-bit servers. If your Zenoss server is a Zenoss virtual appliance or hardware appliance, then the remote server must also be a Zenoss appliance.
- For each distributed collector you add, you must have an additional 500 MB of system memory on the machine running the collector.
- 'when you update your version of Zenoss or install additional ZenPacks, you should update your hubs and collectors.

For additional platform-specific information, refer to Platform Notes.

1.2. Navigating Existing Collectors and Hubs

When you log in as the Zenoss admin user, the Navigation pane displays a link titled Collectors in the Management area. Click this link to go to the Collectors page, which lists existing hubs and collectors in hierarchical form. Hubs are listed at the top level; collectors are nested below the hub to which they belong.

From this page, you can:

- Add a hub
- Delete a hub (which also deletes its associated collectors)

- View and edit hub settings

The Daemons tab lists the copy of the zenhub daemon that belongs to the collector. Links adjacent to the daemon name allow you to view its log, and view and edit its configuration. Use the buttons to the right of the daemon name to stop, start, and restart the daemon.

2. Typical Usage Scenarios for Distributed Monitoring

Here are a few typical setup scenarios for utilizing multiple hubs and collectors:

- ZeoDB - local hub - local collector
- ZeoDB - local hub - multiple local collectors
- ZeoDB - local hub - remote collector
- ZeoDB - local hub - multiple remote collectors
- ZeoDB - remote hub - remote collector
- ZeoDB - multiple remote hubs - multiple remote collectors
- ZeoDB - local hub - local collector(s) - remote hub(s) - remote collectors

Each of these scenarios can address the following big-picture business cases to help make your monitoring more effective, efficient and balanced.

2.1. Scalability on the Local Device

You can use distributed monitoring to have multiple local collectors that will utilize multiple CPUs. Since each collector is one thread, the use multiple collectors allows the device to use its own load balancing to properly optimize multiple CPUs. If you were to only have one hub and one collector locally, the load would be concentrated on a small portion of the potential power.

2.2. Scalability Over Multiple Remote Devices

Scaling on multiple remote devices basically enhances the same idea of single threads as with a single local device, but spreads the collection and configuration burden across multiple machines to maximize efficiency.

2.3. Collectors and or Hubs Set Up through a Firewall

If you want to have collectors and hubs on different sides of a firewall, use the ZeoDB hostname and port numbers to manage the flow of data through your firewall.

2.4. Collection Over Multiple Overlapping IP Spaces

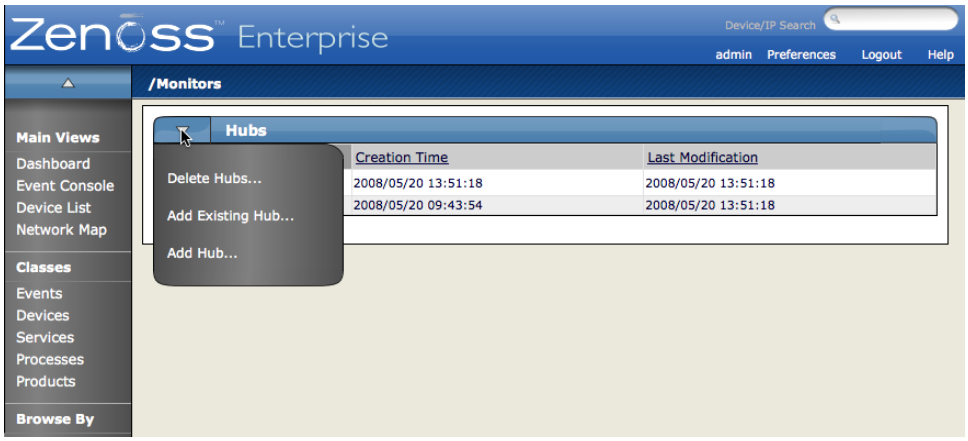
Distributed monitoring is also an essential component for enabling multiple realm IP support available to Zenoss Service Provider customers.

3. Deploying Hubs

Hubs are used to manage configuration data and pass it to the collectors. Hubs also take data from the collectors and pass it to the ZeoDB. More hubs can be a more efficient way to manage larger deployments, as they help distribute the computing resources when configuration changes are made. They further remove the potential for configuration changes to be a bottleneck to gathering and processing data.

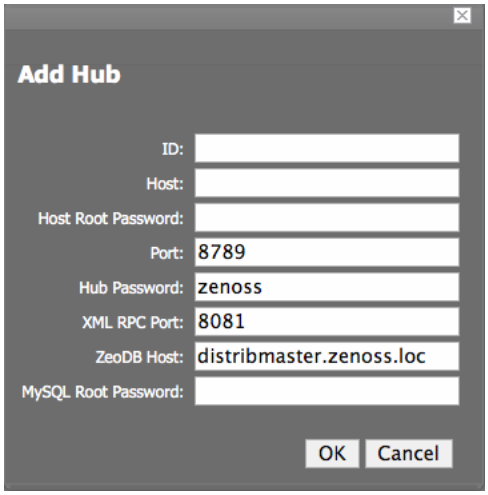
To add a hub, from the main Collectors page, select Add Hub... from the table menu.

Figure 28.1. Hubs Table menu



The Add Hub dialog appears:

Figure 28.2. Add New Hub dialog



Use the following fields to define the hub:

- ID - Enter a name for the new hub. The name can be any unique combination of letters, digits, and dashes.
- Host - Enter the fully qualified domain name, IP address, or resolvable hostname of the server on which the new hub will run.
- Host Root Password - Enter the root user password for the server you specified in the Host field.
- Port - Enter the port number on which the hub should listen for collectors. The default port is 8789.
- Hub Password - Enter the hub password that the collectors will use to log into this hub. The default password is "zenoss."
- XML RPC Port - Specify the port on which the hub should listen for xml-rpc requests from the collectors or other API clients.
- ZeoDB Host - Specify the server hosting the ZeoDB database (the object database). In most cases, this is the IP address or hostname of the main Zenoss server.

- MySQL Root Password - Enter the password for the root MySQL user. If the MySQL root user does not have a password, then leave this field blank.

Click OK. The system displays log output from the creation of the new hub. When fully configured (this may require several minutes), click the link at the bottom of the page to go to the overview page for the new hub.

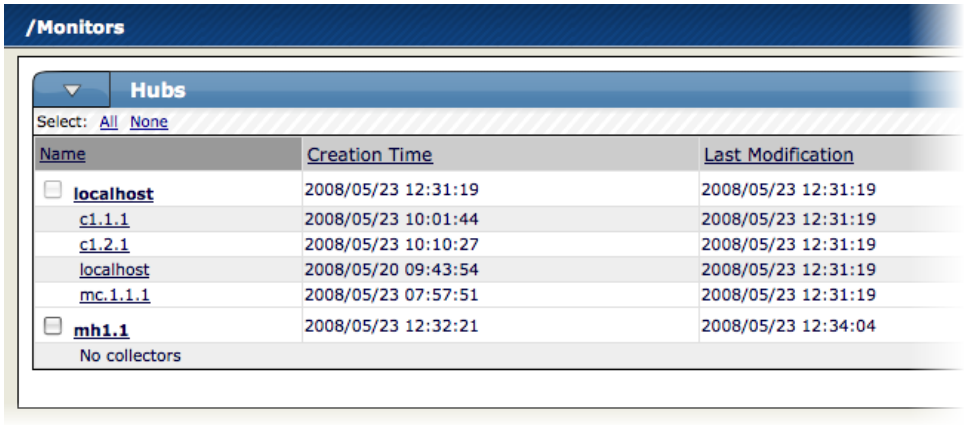
4. Deploying Collectors

To add new collector to a hub:

1. From the left navigation menu, select Collectors.

   The main Collectors page appears, showing all of the hubs and collectors.

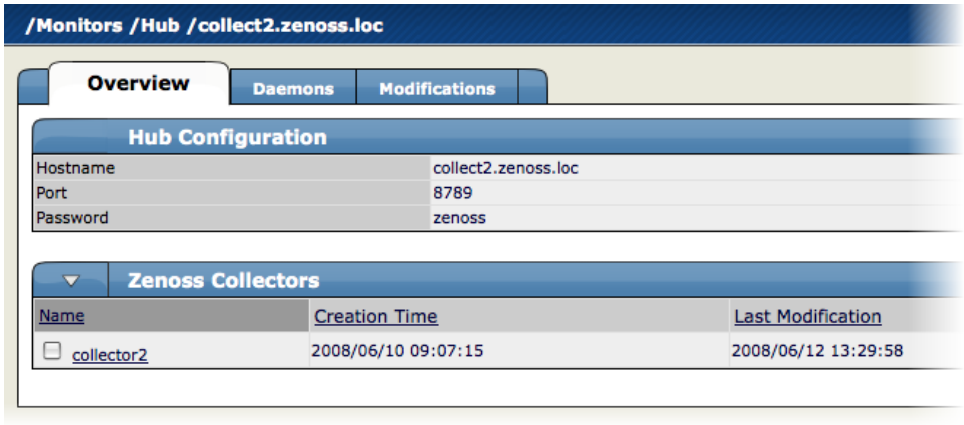
Figure 28.3. Main collectors page



2. Click the name of the hub where you want to add the collector.

   The main page for this hub appears.

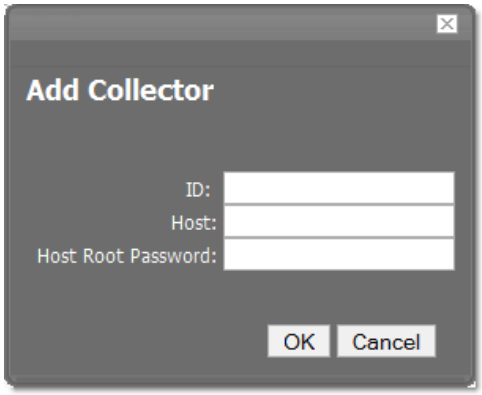
Figure 28.4. Main hub page



3. From the Zenoss Collectors table menu, choose Add Collector...

   The Add Collector dialog appears.

Figure 28.5. Add Collector Dialog



4. Add information to the dialog to define the collector.

   - ID - In the ID field, enter the name for the collector as it will be identified in Zenoss.
   - Host - Enter the name of the host for the collector. This must be a fully qualified domain name, IP address, or resolvable hostname. If you want the collector to run on the Zenoss server, then enter localhost.
   - Host Root Password - Enter the password for the root user on the Host.

5. Click OK. The system displays log output from the creation of the new collector. When fully configured (this may require several minutes), click the link at the bottom of the page to go to the overview page for the new collector.

4.1. Deleting Collectors

To delete a collector, click the name of the hub where the collector exists from the main collectors page. The Hub overview page appears. From the list of Zenoss Collectors, select the collector you want to delete. From the Zenoss Collectors table menu, select Delete Collector.

When you delete collectors using this Zenoss instance, they are not removed or "uninstalled" in anyway from the collector device. They continue to exist on the device until manually removed through the file system.

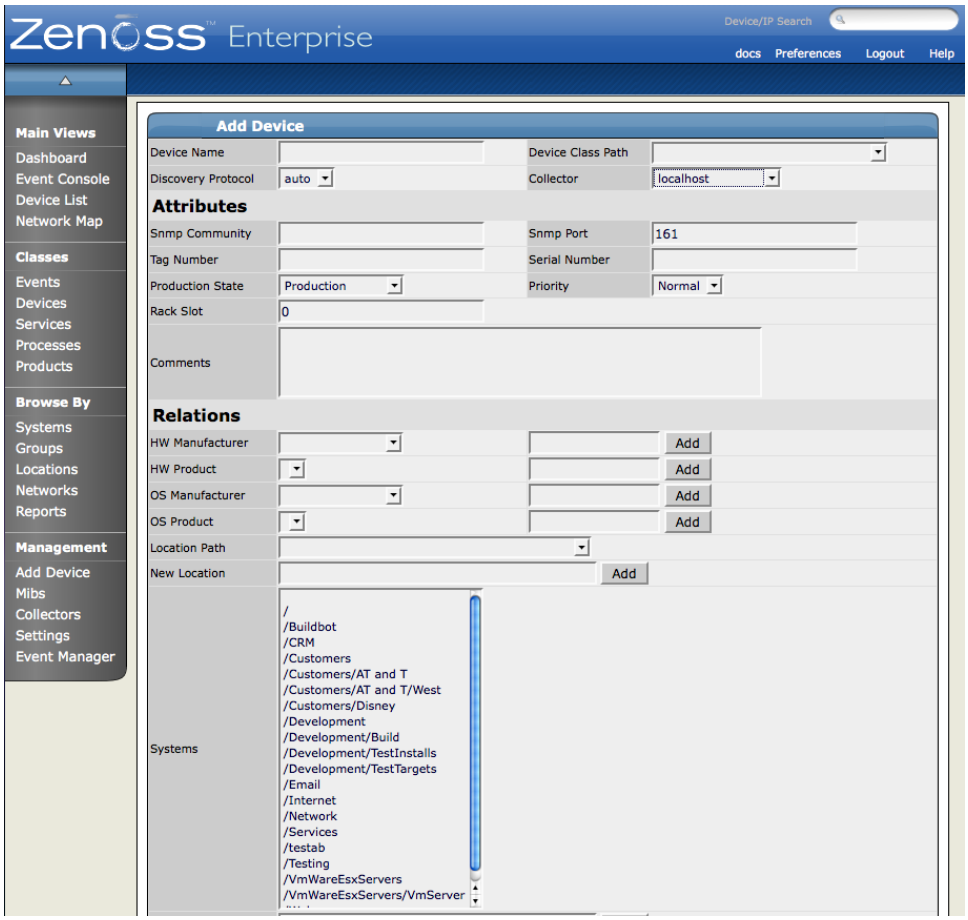
4.2. Updating a Hub or Collector

Any time you update your version of Zenoss or install additional ZenPacks, you should update any hubs or collectors. You do this by navigating to the Overview page for the hub or collector, and then choosing Update Hub or Update Collector from the page menu. This copies the most recent Zenoss code and ZenPacks to the server and restarts the daemons running there.

5. Adding Devices to Collectors

Adding devices to collectors occurs when you add the device to Zenoss. When you click Add Device from the left navigation menu you see the Add Device page.

Figure 28.6. Add Device Page



In the top right of the resulting page, you can see the Collector drop-down menu. Choose the collector you want to use to collect the data for this device. The device then appears in the Device area for that collector. You can access this page by choosing the Collectors item from the left navigation menu and then choosing the collector from the resulting list. When you click on the name of the collector, the Overview page for the collector appears and then in the Devices area at the bottom of the page you can see the Device list for this collector.

5.1. Moving Devices Between Collectors

You can move devices from one collector to another. To do this, use the Set Collector table menu item for any Device view. These are:

- Device List
- Device Tree
- Any of the Organizers device list

Follow these steps to move one or more devices to different collector:

1. From the navigation menu, select Device List.

   The Device list appears.

2. Select one or more devices that you want to move to a different collector.

3. From the Device list table menu, select Set Collector.

   The Set Collector dialog appears.

4. Select the collector from the list of options.

5. Click OK.

   Zenoss moves the device or devices to the selected collector.

5.1.1. Moving VMware Devices Between Collectors

If you move a VMware device to a different collector, you must follow one of these procedures to force the changes to take effect:

- Restart the collector daemons. To do this, go to Settings > Daemons, and then click Restart in the row for each of these daemons:
  - zenvmwaremodeler
  - zenvmwareperf
  - zenvmwareevents

  Note

  Alternatively, as user zenoss, enter the following commands to stop and then restart these Zenoss daemons:

```
zenvmwaremodeler restart
zenvmwareperf restart
zenvmwareevents restart
```

  OR

- Navigate to the page for the organizer that represents the VMware endpoint (for example, Devices/VMware/myEndpoint), and then select Manage > Push changes from the page menu.

5.2. Installation Notes

- Make sure the Zenoss server's hostname is a fully qualified domain name.
- Remember that collectors and hubs can be pushed only to servers with identical operating system versions and hardware architecture.
- Make sure that Event Manager > hostname has a fully qualified domain name (preferred) or at least a numeric address that can be reached by any server that has collectors or hubs deployed to it.
- Distributed Collector shuts down the iptables firewall on the Zenoss server. If you have any other firewalls on the Zenoss server, or on servers that host remote collectors or hubs, then you should disable their firewalls also.

5.3. Debugging

Hostname Configuration

The Zenoss server should have a properly configured hostname, preferably a fully qualified domain name. You can check the hostname from the shell:

root# hostname

You also can check by using the Python function used by Zenoss:

root# python -c 'import socket; socket.gethostname()'

MySQL Host

Hubs on remote servers need access to the MySQL events database. This setting is the Hostname field in the Connection Information section of the Event Manager page. By default this is set to localhost, but will not work for remote hubs. Distributed collector attempts to set this field to the fully qualified domain name of the Zenoss server when it is installed. If remote hubs appear to be having trouble connecting to MySQL or sending events, then check the value in this field to make sure it can be reached from the server the hub is on.

MySQL Access Privileges

Another aspect of remote hubs connecting to MySQL is privileges. For a hub to connect to the events database, the user specified in the User Name field in Event Settings must be granted privileges to connect to MySQL from the remote server. Distributed Collector attempts to grant these privileges any time a remote hub is created or updated. If a remote hub is logging error messages that indicate it is not allowed to connect to MySQL from the given host, then these privileges are likely not set up correctly. Granting of these privileges requires a fully qualified domain name for the remote server.

5.4. Firewall Notes

When a hub or collector is deployed DistributedCollector attempts to disable the iptables firewall on the Zenoss server. This enables the remote collector or hub to communicate with the daemons and databases running on the Zenoss server. You can re-enable iptables if you want to set up the proper rules to allow Zenoss traffic through. Remote hubs need to communicate with the zeo database on the Zenoss server on port XXXX. Hubs also need to communicate with the MySQL server, usually on the Zenoss server (see Event Manager > Hostname), on the port specified in Event Manager > Port (usually 3306.) Collectors communicate with their hub on the port specified when the hub was created. See the ZenHub Port field on the hub's overview page.

5.5. Platform Notes

Software Appliance and Hardware Appliance

- Hubs and collectors can be deployed only to other Zenoss software or hardware appliances.
- You must stop Zenoss on an appliance before deploying a hub or collector to it.
- You must set a password for the root user on an appliance before deploying a hub or collector to it.
- Do not use conary to update appliances being used as remote collectors or hubs.

Chapter 29. Monitoring Virtual Host Machines

1. About Monitoring Virtual Host Monitoring

Note

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

The ZenossVirtualHostMonitor ZenPack allows you to monitor virtually hosted operating systems with Zenoss. This ZenPack refers to a Virtual Machine Host as the one running on the bare metal, and Guest for those running within the virtual hardware.

This ZenPack:

- Extends Devices to support a relationship from Host to Guest.
- Extends ZenModeler to find Guest OS's and add them to Virtual Hosts
- Provides screens and templates for collecting and displaying resources allocated to Guest OSs

2. Using Zenoss Virtual Host Monitor

To Use ZenossVirtualHostMonitor:

1. Make sure you have SNMP connectivity to your ESX 3.0 servers.
2. Make sure you have SSH keys to your Xen servers (as root).
3. Create your ESX 3.0 servers using the /Servers/Virtual Hosts/ESX device class. If you have these servers modeled already, remove them and recreate them under the ESX device class. DO NOT MOVE THEM.
4. Repeat step 3 with Xen hosts (using the Xen device class).
5. Select the Guest menu and ensure that the guest hosts were found when the devices were added.
6. Using the VMware Virtual Infrastructure Client, add Zenoss to the list of destinations for SNMP traps. See the Menus Administration-> VirtualCenter Management Server Configuration ->SNMP.

Optionally, you can install the VMware ESX MIB files in Zenoss.

Additional Notes:

- There is a handy link to the VMware web interface on each ESX server Status page.
- If the name of the Guest under ESX is the same as the name of a device being monitored directly by Zenoss, a link will be provided to take you directly from to that device from the Guest list.
- The CPU monitoring on VMware shows very little relationship to actual CPU used. We have asked VMware about the problem.

Chapter 30. Predictive Thresholding (ZenHoltWinters)

1. About Predictive Thresholding

Note

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

The ZenHoltWinters ZenPack adds the ability to fire threshold events when a device exceeds cyclical predicted values. Holt-Winters is the algorithm that we use for this prediction.

2. Using Predictive Thresholding

To use ZenHoltWinters, you need to add HoltWinters Thresholds against performance data. In any template, select add a new Threshold and select the HoltWintersFailurs threshold type.

Note: Zenoss relies on the existence of Holt-Winters RRAs within an RRD file. After adding Holt-Winters thresholds, the RRD files being monitored will need to be re-created so that the new configuration can occur. At the time of Zenoss Enterprise 2.1.1, there is no conversion utility. You will have to remove any existing RRD files so that new files can be created.

Important Note: Removing RRD files will remove any historical information associated with these RRD files.

3. Customizing the Thresholds

The Holt-Winters Threshold allows you to customize some aspects of the prediction engine. The 4 values to modify are:

- Rows: The number of points to use for predictive purposes.
- Alpha: A number from 0 to 1 that controls how quickly the model adapts to unexpected values.
- Beta: A number from 0 to 1 that controls how quickly the model adapts to changes in unexpected rates changes.
- Season: The number of primary data points in a season. Note that Rows must be at least as large as Season.

Chapter 31. Authentication ZenPacks

1. Active Directory Authentication

1.1. About Active Directory Authentication

Note

The ActiveDirectory ZenPack is available only for Professional and Enterprise versions.

The ActiveDirectory ZenPack allows you to authenticate users using Microsoft Active Directory.

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

1.2. Active Directory Monitored Metrics

Use the Active Directory ZenPack to monitor these metrics:

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

- LDAP New SSL Connections/Sec
- LDAP Searches/Sec
- LDAP Successful Binds
- LDAP UDP Operations/Sec
- LDAP Writes/Sec
- NTLM Authentications

2. LDAP Authentication in Zenoss

2.1. About LDAP Authentication

Note

The LDAPAuthenticator ZenPack is available only for Professional and Enterprise versions.

The LDAPAuthenticator Enterprise ZenPack allows Zenoss to use your existing LDAP authentication infrastructure (such as Active Directory and OpenLDAP) to enable single sign-on to the Zenoss Web interface.

2.2. Setting Up LDAP Authentication

After installing the LDAPAuthenticator ZenPack, follow these steps to set up the integration.

1. Browse to this URL:

   http://`yourzenossinstallation`:8080/zport/acl_users/manage

2. Choose your authentication plugin, and then click Add.
   - Choose "ActiveDirectory Multi Plugin" for Microsoft Active Directory.
   - Choose "LDAP Multi Plugin" for any other LDAP server.
3. Complete the form with your LDAP credentials and paths:
   - ID - Enter a value similar to "ldapAuthentication."
   - Title - Leave this value blank.
   - Default User Roles - Set to ZenUser.
4. Click Add to save your changes.
5. Click plugins in the list of objects.
6. Click the Authentication Plugins link.
7. Move your ldapAuthentication plugin to the list of active plugins.

Chapter 32. Transaction Monitoring ZenPacks

1. Synthetic Transactions (ZenWebTx)

1.1. About Synthetic Transactions (ZenwebTx)

Note

The ZenWebTx ZenPack is available only for Zenoss Professional and Enterprise versions.

The ZenWebTx ZenPack allows you to test the availability and performance of Web sites. You create one or more tests that mimic a user's actions in a Web browser. Zenoss then performs these tests periodically. Zenoss creates events when a test fails or exceeds a time threshold.

Additionally, Zenoss can record data for each test run, such as:

- Time required for the test to execute
- Time taken for any portion of the test to complete
- Values extracted from Web pages during the test

1.2. Overview

Each test is represented by a single data source of type WebTx. ZenWebTx uses a scripting language called Twill to describe the steps of a test. These steps include actions such as:

- Clicking a link
- Completing form fields
- Assertions (to check for the presence or absence of text on a page)
- Descriptions of data to collect during the test

You can write Twill commands manually. You also can use a Firefox add-on called TestGen4Web to record a browser session that ZenWebTx then translates into Twill. The zenwebtx daemon processes the Twill commands periodically, recording data and creating events as appropriate.

WebTx data sources can contain many data points, depending on the Twill commands you specify. These include:

- totalTime - Total number of seconds required to execute the transaction. All WebTx data sources include this data point.
- available - Success or failure of the transaction. All WebTx data sources include this data point.
- The presence or absence of text patterns on a page. In addition, you can extract from the Web page and record the numeric values that are part of these patterns.

The ZenWebTx package includes:

- The WebTX data source, which allows you to specify a Web transaction
- The zenwebtx daemon, which processes WebTx data sources
- The /Status/Web and /Perf/Web event classes

- A sample performance template (ZenWebTx Example)

1.3. Creating a WebTx Data Source

Like data sources that use SNMP or other methods to gather data in Zenoss, you create WebTx data sources in performance templates.

To create a WebTx data source:

1. From the data sources area, select Add DataSource from the table menu.
2. In the Create Data Source dialog, enter the name of the new data source, and then select the data source type "WebTx."
3. Click OK.

   The Data Source tab appears.
4. Enter information or make selections to specify how and when this data source's Web transactions are performed, and which data should be collected:

   - Name - Displays the name of the data source that you specified in the Create Data Source dialog. This name is used in thresholds and graph definitions to refer to the data collected by this data source.
   - Source Type - Set to WebTx, indicating that this is a synthetic Web transaction data source. You cannot edit this selection.
   - Enabled - Set to True (the default) to collect information from this data source. You may want to set this value to False to disable data sources when developing the data source, or when making changes to the Web application being tested.
   - Component - Any time the Web transaction fails, Zenoss generates an event. Use this field to set the Component field of the generated event.
   - Event Class - Select the event class of the event generated by this data source. Normally, this is set to /Status/Web.
   - Timeout - Specify the number of seconds that zenwebtx will attempt to execute this data source's commands before it generates an error event.
   - Cycle Time - Specify the number of seconds that zenwebtx will wait between the start of one test run and the start of the next.
   - User Agent - Specify the text that zenwebtx will present to target Web sites to identify itself.
5. Click Save to save the specified settings.
6. Click the Script tab. From here, you will specify the details of the transaction. Information here also helps you debug Twill commands when setting up the data source.

   Enter information or make selections:

   - Initial URL - Specify the URL of the page where the transaction will start. This field frequently contains a TALES expression to refer to a device's ID or IP address, such as http://${dev/id} or http://${dev/manageIp}.

     For more information on TALES expressions, refer to the Appendix in this guide titled TALES Expressions.
   - Initial User - Specify the user name for authentication.
   - Initial Password - Specify the user password for authentication.
   - Initial Authentication Realm - Specify the basic HTTP authentication realm.

Note

If you provide values for Initial User, Initial Password, and Initial Authentication Realm, Zenoss will use these credentials before accessing the URL specified for Initial URL. All three (Initial User, Initial Password, and Initial Authentication Realm) must be present; otherwise, the values are ignored.

- TestDevice - Use this field to test and debug Twill commands. Enter the ID of a device, and then click Test Twill Commands to execute the Twill commands against the device. If you do not specify a device, then Zenoss will select a device for you.
- Upload Recording - Upload a Web session recording generated by the Firefox TestGen4Web add-on. Enter or browse to the recording location.

  If you specify a file here, and then click Save, Zenoss translates the file to Twill commands and replaces the contents of the Twill Commands field with the newly translated commands.

  ### Note

  If you select this action, then the current contents of the Twill Commands field is completely replaced. Zenoss does not save the replaced information.

- Twill Commands - Enter Twill commands that Zenoss will execute to produce values and events for the data source.

  See the section titled Creating Twill Commands for more information about Twill commands.

7. Click Save to save the data source.

1.4. Creating Twill Commands

ZenWebTx uses a language called Twill to specify the steps of a Web test. Each WebTx data source has a field that contains the Twill commands that describe a Web transaction. You can create this list of Twill commands manually, or you can record a session in a browser and use that as the basis for your data source.

Some Twill commands specify an action, such as following a specific link on a page or entering data in a form field. Other Twill commands specify a test, such as searching for specific text on a page or making sure the title does not contain specific text. The full range of available commands is described in the section titled "ZenWebTx Twill Commands Reference".

1.5. TestGen4Web

The TestGen4Web Firefox add-on allows you to record browser sessions. ZenWebTx can take these sessions and convert them to Twill, creating a starting point for developing ZenWebTx data sources.

Download the TestGen4Web add-on at:

https://addons.mozilla.org/firefox/addon/1385

Follow these general steps to record and convert a TestGen4Web session:

1. From the TestGen4Web toolbar in Firefox, use the Record and Stop buttons to record a session.
2. Use the Save button in the toolbar to save the session to a file.
3. From the Script page of a ZenWebTx data source in Zenoss, browse to and select your saved session.
4. Click Save to convert the TestGen4Web session to Twill. The newly converted commands appear in the Twill Commands field on the page, replacing any previous Twill commands in that area.

1.6. Creating Twill Commands Manually

Even if you use TestGen4Web to initially create Twill commands, you will frequently want to edit these commands manually to add data points or additional content checks. The ZenWebTx Twill Commands Reference describes in detail the commands that you can use. The Test Twill Commands button on the Script page is helpful when testing Twill commands as you create or edit them.

You also can execute Twill commands interactively by using the twill-sh program from the command line. This program lets you enter commands one at a time and then inspect the pages that come back.

Invoke twill-sh with:

```
> PYTHONPATH=$ZENHOME/Products/ZenWebTx/lib
$ZENHOME/Products/ZenWebTx/bin/twill-sh
```

Within twill-sh, use the help command to list available commands and see a command descriptions. Of particular interest are these commands:

- showforms – Lists the forms on the page and the fields within each.
- showlinks – Lists the links on the page.
- show – Lists the source HTML from the page.
- exit – Quits the twill-sh program.

Often the most convenient way to use twill-sh is to create a text file that contains your Twill commands. You can then specify that file on the command line when you invoke twill-sh. This lets you analyze problems that occur.

Invoke twill-sh with a text file as such:

```
> PYTHONPATH=$ZENHOME/Products/ZenWebTx/lib
$ZENHOME/Products/ZenWebTx/bin/twill-sh -i myTwillCommands.txt
```

The -i option instructs twill-sh to stay in the Twill shell, rather than exiting when it finishes running the commands in the myTwillCommands.txt file.

1.7. Event Generation

There are several situations for which ZenWebTx will create events in Zenoss. These events use the component and event class specified on the Data Source tab. These situations are:

- ZenWebTx is unable to retrieve a page during the transaction.
- One of the Twill commands fails, such as finding text that does not exist or following a link that does not exist.
- The timeout (specified on the Data Source tab) is exceeded.
- A threshold defined for one of the data points in this data source is exceeded. Thresholds are defined in the performance template that contains the data source.

1.8. Data Points

Data produced by any Zenoss data source are called data points. ZenWeb data sources contain two default data points:

- totalTime – Number of seconds taken to complete the entire transaction.
- success – Indicates 1 or 0, depending on whether or not the transaction succeeded.

You can create other data points by using the extract and printTimer Twill commands, which output data values when the Twill commands are run. You must create new data points with the same name you used in those commands to bring that data into Zenoss.

For more information about the extract and printTimer Twill commands, refer to the section titled "Twill Commands Reference."

ZenWebTx supports using xpath queries to extract data from XML documents. For more information about this feature, refer to the appendix in this guide titled "ZenWebTx Twill Commands Reference."

2. Mail Transactions Monitoring (ZenMailTx)

2.1. About Mail Transaction Monitoring

Note

The ZenMailTx ZenPack is available only for Zenoss Professional and Enterprise versions.

The ZenMailTX ZenPack allows you to monitor round-trip email delivery. It provides a new type of data source, MAILTX, that contains server and message information. ZenMailTX data sources are processed by the zenmailtx daemon.

2.2. Creating a ZenMailTX Data Source

To create a MailTX data source, follow these steps:

1. Navigate to a performance template, and then select New Datasource from the Data Sources table menu.
2. Enter a name for the data source.
3. Select MAILTX as the type, and then click OK.
4. Enter values for the To Address and From Address.
5. Enter the SMTP server details, including the transport security (TLS or SSL) and port (by default, port 25).
6. Enter the POP server details, including the transport security and port (by default, port 110).
7. Enter the body of the test message.
8. Click Save.

2.2.1. Testing Data Source Settings

To test the data source settings:

1. Select the Test Now tab.
2. Enter a device against which to test the data source.
3. Click Test Email.

Any of the MAILTX fields can take TAL expressions. It is often convenient to use a customer variable to define the value of the password fields. Because those fields are expected to be set via customer variables, they are in plain text fields (rather than the typical password entry field).

2.3. zenmailtx daemon

The zenmailtx daemon:

- Sends the test email message via the specified SMTP server
- Retrieves the email message from the specified POP server

- Sends the following information to Zenoss:
  - Time taken to send
  - Time taken to fetch
  - Total time

This daemon appears on the Zenoss Daemons page and can be started, stopped and restarted from there. From the command line it can be controlled via the zenoss script or by issuing commands directly to the daemon, which is symlinked at $ZENHOME/bin/zenmailtx.

3. SQL Transaction Monitoring (ZenSQLTx)

3.1. About SQL Transaction Monitoring

Note

The ZenSQLTx ZenPack is available only for Zenoss Professional and Enterprise versions.

The ZenSQLTx ZenPack allows you to test the availability and performance of MySQL and MSSQL servers. It provides the SQL data source, which contains a set of SQL queries to be executed against a server. SQL data sources are processed by the zencommand daemon.

3.2. Creating a SQL Data Source

To create a SQL data source, follow these steps:

1. Navigate to a performance template, and then select Add DataSource from the Data Sources table menu.
2. Enter a name for the data source, select SQL as the type, and then click OK.

   The Data Source page appears.
3. Enter or select these values:
   - Database Type - Select MySQL or MS SQL.
   - Host Name - Set the host name on which the database is located. This field accepts a TALES expression, such as "${here/id}" or ${here/getManageIp}".
   - Port - Set the port on which the database server is listening. For MYSQL, the default port is 3306. For MSSQL, the default port is 1433.
   - Database Name - Specify the name of the database.
   - User - Specify a user name with permission to connect to the database and run queries.
   - Password - Specify the user password.
   - SQL Queries - Specify the SQL queries that this data source should execute. A summary of MySQL syntax is available at:

     http://dev.mysql.com/doc/refman/5.0/en/sql-syntax.html

     A summary of MS SQL syntax is available in the documentation accompanying the software.
4. Click Save.

   Zenoss creates a data point that corresponds to the total query time (in milliseconds).

Chapter 33. Application Monitoring ZenPacks

1. MS SQL Monitoring (MSSQLServer)

1.1. About MS SQL Transaction Monitoring

Note

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

The MSSQLServer ZenPack monitors Microsoft SQL Server and its related services. The ZenPack enables users to view graphs based on MS SQL Server Performance Counters and to monitor processes related to MS SQL Server. This ZenPack is dependent on the ZenWinPerf ZenPack in its use of the WinPerf datasource type.

1.2. Installed Objects

1.2.1. Device Class

- /Devices/Server/Windows/MSSQLServer

1.2.2. Event Class

- /Events/Win/MSSQLServer

1.2.3. WinServices

- MSSQLSERVER
- MSSQLServerOLAPService
- MsDtsServer
- ReportServer
- SQLBrowser
- SQLSERVERAGENT
- SQLServer
- SQLWriter
- msftesql

1.2.4. IpServices

- ms-sql-s

1.2.5. Performance Templates

- MSSQLServer

1.2.6. Datasources

- ssamFullScansPerSec

- ssbmBufferCacheHitRatio
- ssbmFreePages
- ssdDataFileSSizeKB
- ssgsUserConnections
- sslkAverageWaitTimeMs
- sslkLockRequestsPerSec
- sslkNumberOfDeadlocksPerSec
- ssltLatchWaitsPerSec
- ssmmConnectionMemoryKB
- ssssBatchRequestsPerSec

1.2.7. Thresholds

- ssbmBufferCacheHitRatioLessThan90Percent

1.3. Using the MSSQLServer ZenPack

After installation you will have a device class MSSQLServer. Place any devices with MSSQLServer installed into this device class. An MSSQLServer template will automatically be bound to your devices. Make sure the zenwinperf daemon is running. On a device, click on the 'More' – 'Push Changes' menu item. This will update the collectors. To view the graphs, click on the 'Perf' tab.

2. MS Exchange Monitoring (MSExchange)

2.1. About MS Exchange Monitoring

Note

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

The MS Exchange ZenPack is an application monitoring ZenPack that monitors Micosoft Exchange and its related services. The ZenPack enables users to view graphs based on MS Exchange Performance Counters and to monitor processes related to MS Exchange. This ZenPack is dependent on the ZenWinPerf ZenPack and its use of the WinPerf datasource type.

2.2. Installed Objects

2.2.1. Device Classes

- /Devices/Server/Windows/MSExchange

2.2.2. Event Classes

- /Events/Win/Exchange

2.2.3. Event Class Keys

- MSExchangeDSAccess_2064
- MSExchangeDSAccess_2069

- MSExchangeIS_9582

2.2.4. Alerting Rules

- MSExchangeISWMTotal16MBFreeBlocks

2.2.5. WinServices

- IISADMIN
- IMAP4Svc
- MSExchangeES
- MSExchangeIS
- MSExchangeMGMT
- MSExchangeMTA
- MSExchangeSA
- MSExchangeSRS
- POP3Svc
- RESvc
- SMTPSVC
- W3SVC

2.2.6. IP Services

- imap4-ssl
- msexch-routing

2.2.7. Performance Templates

- MSExchangeIS is automatically bound to Device Class MSExchange

2.2.8. Datasources

- logicalDiskFreeMegabytes
- memoryPercentCommittedBytesInUse
- mseisMailboxFolderOpensPerSec
- mseisMailboxLocalDeliveryRate
- mseisMailboxMessageOpensPerSec
- mseisRPCAveragedLatency
- mseisRPCOperationsPerSec
- mseisRPCRequests
- mseisVMLargestBlockSize
- mseisVMTotalFreeBlocks

- mseisVMTotalLargeFreeBlockBytes
- mseisVMTotal16MBFreeBlocks

2.2.9. Thresholds

- logicalDiskFreeMegabytesUnder100
- logicalDiskFreeMegabytesUnder25
- memoryPercentCommittedBytesInUseOver75
- mseisVMLargestBlockSizeUnder16
- mseisVMLargestBlockSizeUnder32
- mseisVMTotal16MBFreeBlocksUnder3
- mseisVMTotalLargeFreeBlockBytesUnder32

2.3. Performance Counters and Thresholds

ZenPack will automatically set up performance monitoring templates for the following core metrics:

2.4. Reports, Views & Dashboards

- MSExchangeAvailability.pt
- Availability Summary
- Up/Down%, Uptime %, Up Since (Dials?),
- Current status matrix for key exchange services

2.5. Using the the MSExchange ZenPack

After installation you will have a device class MSExchange. Place any devices with MSExchange installed into this device class. An MSExchange template will automatically be bound to your devices. Make sure the zenwinperf daemon is running. On a device, click on the 'More' – 'Push Changes' menu item. This will update the collectors. To view the graphs, click on the 'Perf' tab.

3. IIS Monitoring

3.1. About IIS Monitoring

Note

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

The IISMonitor ZenPack collects key metrics from Microsoft IIS. The metrics are collected using Windows Perfmon and require no agent to be installed on the IIS server.

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

- Other: SEARCH, TRACE, LOCK, UNLOCK

3.2. Setting up IIS Monitoring

Follow these steps to setup your IIS server to have this information collected.

1. Make sure you have the ZenPack properly installed.
2. Verify that your Zenoss Windows service account has access to the IIS server. Within Zenoss this is specified using zWinUser and zWinPassword.
3. Bind the IIS performance template to the IIS server device within Zenoss.

4. JBoss Application Server Monitoring

4.1. About JBoss Application Server Monitoring

Note

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

JBossMonitor is a ZenPack that allows System Administrators to monitor JBoss Application Servers. JBossMonitor uses the JMX Remote API and accesses MBeans deployed within JBoss that contain performance information about the components that are being managed.

4.2. Overview

The collected performance information includes: pool sizes for data sources (JDBC), enterprise java beans (EJBs), message queues (JMS), threads, servlets, JSPs, and classloaders. Cache information is also accessible, providing system administrators insight into the number of hits (or misses) their cache policy has produced.

The ZenPack also aggregates individual performance metrics into higher level concepts that provide a picture of the performance of the application. Cache hits and misses are combined on the same graph to provide an overall picture of cache performance. Likewise, queue metrics are combined to show the number of messages currently on the queue, being processed, and being placed on the queue. Queue subscribers and publishers are also graphed.

Each of the individual performance metrics can be trended and predicted, and thresholds can be explicitly defined. Both the predicted thresholds and explicit thresholds inform system administrators of potential future problems before they occur. Since so much of J2EE involves "managed resources", the ability to monitor pool sizes and alert administrators prior to resources being exhausted is extremely valuable and can reduce the likelihood of a fatal outage caused by resource depletion.

Most of the metrics that are collected in JBossMonitor represent combinations of individual component metrics. For example, the Thread Pool metric represents all threads in all pools. It is possible to configure JBossMonitor to perform at higher granularity and have it monitor a Thread Pool with a particular name. However, since these names are application and specific we have chosen to configure JBossMonitor to collect at a rather coarse grained level out of the box. The installer is highly encouraged to customize and configure!

One particular performance template that screams for end-user configuration involves Servlets. If a site to be monitored is revenue generating, and credit card submissions from the website are handled via a back-end servlet it may be critically important to monitor the resources made available by the JBoss container to the servlet container. If the number of free spaces in the servlet pool dwindles to 0 it could prevent your application from making a sale.

4.3. Collected Metrics for JBoss

The following list shows al of the collected metrics for JBoss servers:

- Active Threads

- JMS Message cache memory usage
- JMS Message hits/misses
- JMS Topic/Destination queue size
- Java heap memory usage
- JCA commit, rollback, and transaction count
- JCA Connection pool in-use connections and available connections
- JCA connections created/destroyed
- JCA total connections
- JGroups cluster messages sent/received
- JGroups cluster bytes sent/received
- MBean creation/removal count
- MBean messages processed count

4.4. Setting Up the JBoss ZenPack

The JBoss Application server ZenPack does not require any additional setup after unzipping. Although you can customize the data points the same way you would customize any data point in Zenoss.

5. WebLogic Application Server Monitoring

5.1. About WebLogic Application Server Monitoring

Note

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

WebLogicMonitor is a ZenPack that allows System Administrators to monitor a WebLogic Server. WebLogicMonitor uses the JMX Remote API and and accesses MBeans deployed within JBoss that contain performance information about the components that are being managed. This performance information includes pool sizes for data sources (JDBC), threads, connections (JCA), queues (JMS), servlets, JSPs, enterprise java beans (EJB), timer queues.

Throughput is also monitored when it is available. This metric is computed by WebLogic and is based on the number of messages moving through a queue at any given time. The throughput metric gives a good picture of the health of the messaging subsystem, which is commonly used throughout many enterprise applications. Stateless, Stateful, and Entity EJB performance metrics are monitored, as are message driven bean performance.

Security realms are also monitored for potential denial of service attacks. This includes recording of authentication failures, broken out by valid accounts, invalid accounts, and accounts that are currently locked out. Application specific realms can be monitored by customizing the built in WebLogic default realm.

All of the observed values are fed into the comprehensive Zenoss thresholding and event subsystem, allowing events to be correlated.

5.2. Weblogic Collected Metrics

5.2.1. Overall Application Server Vitals

- Number of total and active JMS connections and servers

- Overall number of JTA transactions that are rolled back or abandoned
- JTA transactions rolled back due to system, app, or resource issues
- Number of JTA rollbacks that timeout
- Active and committed JTA transaction count
- Timer exceptions, executions, and scheduled triggers
- User accounts that are locked and unlocked
- Authentication failures against locked accounts and non-existent accounts
- Total sockets opened, and the current number of open sockets
- JVM Mark/Sweep and Copy garbage collector execution counts
- Number of JVM daemon threads
- JVM Heap/Non-Heap used and committed memory

5.2.2. Entity EJB, Message Driven Bean (MDB), and Session EJB Subsystem Metrics

- Rollback and commit count on a per-EJB basis
- Bean pool accesses, cache hits, and cache misses
- Number of Beans in use, idle, and destroyed
- Number of activations and passivations

5.2.3. Data Pool (JDBC) metrics

- Leaked, Total, and Active connections
- Number of requests waiting for a connection
- Number of reconnect failures

5.2.4. Queue (JMS) Metrics

- Bytes received, currently active, and pending in the queue
- Number of queue consumers
- Number of current, pending, and receives messages

6. Tomcat Application Server Monitoring

6.1. About Tomcat Application Server Monitoring

Note

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

TomcatMonitor is a ZenPack that allows System Administrators to monitor the Tomcat Application Server. Tomcat is a web application container that conforms to many parts of the J2EE Specification.

The more extensive JBoss Application Server uses Tomcat as a Web Application engine to manage web applications deployed inside enterprise applications within JBoss. As a result, the TomcatMonitor ZenPack can be used to monitor Tomcat MBeans that are active within JBoss.

This ZenPack focuses on the metrics that Tomcat updates in it's internal MBean container that is accessible via the remote JMX API.

These metrics focus on attributes that relate to the servicing of web pages and primarily include thread pool size, CPU use, available file descriptors, JSP and servlet counts, and request counts.

TomcatMonitor places much emphasis on monitoring thread status because every web request is serviced in a separate thread. Each thread requires file descriptors to be maintained, and thus those are monitored as well. The amount of CPU time spent servicing each thread is also captured and reported.

TomcatMonitor also reports on the number of times JSPs and Servlets are reloaded. This metric can be useful in highly dynamic sites where JSPs or Servlets change on the fly and need to be reloaded periodically. Monitoring of this metric can lead to the identification of small "Reloading Storms" before they cause production outages.

The amount of time Tomcat spends servicing a request is also recorded. This extremely high level metric can provide insight into downstream systems that are not monitored. If all the Tomcat resources are within normal tolerances but processing time suddenly spikes it can be an indication that a back-end service (such as a database or another web service) is misbehaving.

All metrics captured by TomcatMonitor can be predicted using the built-in Zenoss trending software. Explicit thresholds can also be defined that operate on top of the existing robust event engine.

These event based notifications can be useful at multiple granularities of monitoring, all the way from the high level "Web Processing Time" metric down to the number of times a servlet is being reloaded per minute.

6.2. Collected Metrics for Tomcat

The following metrics can be collected and graphed for Tomcat:

- Tomcat cache (accesses vs hits)
- Daemon and User thread count
- Overall CPU time
- Global Request Traffic: bytes sent/received
- Global Request Traffic: request count and error count
- Global Request processing time
- JSP/Servlet reload time
- Servlet class loading and processing time
- Servlet request and error count

6.3. Tomcat Monitoring Setup

The out-of-the-box TomcatMonitor data source configuration has been defined at the macro level, but can be configured to operate on a more granular basis. For example, the Servlet Reload Count applies to all servlets in all web applications but it could be narrowed to be Servlet /submitOrder in web application "production server".

Chapter 34. Integration ZenPacks

1. Remedy Integration in Zenoss

1.1. About Remedy Integration

Note

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

The RemedyIntegrator ZenPack provides a way to have Zenoss automatically open tickets in your Remedy system when specific events occur. The cases are opened by Zenoss sending a specially formatted email to the Remedy service's email address.

1.2. Setting Up Remedy Integration

Follow these steps to setup the integration once you have installed the RemedyIntegrator ZenPack.

1. Go to Settings/Users within the Zenoss web interface.

   You will find a new user named "Remedy."

2. Change the email address for this user to the email address that your Remedy service picks up email on.
3. Go to the Alerting Rules tab of the Remedy user.
4. Click on the "Open Ticket" alerting rule to edit it.
5. Click on the Message tab and modify the following attributes for your specific setup:
   - Server: remedy.yourdomain.com
   - Login: remedyUsername
   - Password: remedyPassword
   - Support Company: Your Company
   - Support Organization: YourSupport
   - Assigned Group: YourGroup
6. Optionally edit other attributes you want to customize.
7. Click Save then go back to the Edit tab.
8. Set the alerting rule to Enabled and adjust the event filter to your requirements.

2. RANCID Integration

2.1. About RANCID Integration

Note

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

RANCIDIntegrator is a ZenPack designed to allow integration between the popular RANCID (http://www.shrubbery.net/rancid/) configuration management tool and Zenoss. The integration points between the tools can be described as following:

- Zenoss will build the router.db file for RANCID. This allows for the centralization of administration activities and reduces the duplication of effort normally required to maintain the two tools.
- Implementation of this feature is as easy as adding a cronjob to execute $ZENHOME/bin/zenrancid as often as you'd like the router.db file to be updated.
- Zenoss will automatically run RANCID's rancid-runm tool on a single device in response to a ciscoConfigManEvent SNMP trap being sent from the device to Zenoss. Cisco devices will send this trap whenever their configuration is changed. This allows for real-time capturing of router configuration changes in your CVS repository.

2.2. Setting Up RANCID Integration

To implement this feature you must configure your Cisco routers to send their SNMP traps to the Zenoss server.

Link from Cisco router device status pages to the most recent configuration stored in your CVS repository via viewvc (http://www.viewvc.org/).

To implement this feature, change the zRancidUrl to your viewvc URL, other configuration options can be set using the following zProperties:

- zRancidRoot: File system directory where RANCID is installed. It may be NFS mounted from the real RANCID server. Default is "/opt/rancid"
- zRancidUrl: Base URL to viewvc. Default is http://rancid.mydomain.com/viewvc
- zRancidGroup: RANCID group attribute. Controls what router.db file the device is written to. Can be set at the device class or device level. Default is "router" on the /Network/Router/Cisco class.
- zRancidType: RANCID type attribute. Controls what device type is written to the router.db file. Can be set at the device class or device level. Default is "cisco" on the /Network/Router/Cis

Appendix A. Net-SNMP and Zenoss

1. Net-SNMP Configuration Settings

These are the Net-SNMP configurations settings that are needed for Zenoss. Add these lines to your snmpd.conf.

1.1. Community Information

This line will map the community name "public" into a "security name":

```
# sec.name source community
```

```
com2sec notConfigUser default public
```

This line will map the security name into a group name:

```
# groupName securityModel securityName
```

```
group notConfigGroup v1 notConfigUser
```

```
group notConfigGroup v2c notConfigUser
```

This line will create a view for you to let the group have rights to:

```
# Make at least snmpwalk -v 1 localhost -c public system fast again.
```

```
# name incl/excl subtree mask(optional)
```

```
view systemview included .1
```

This line will grant the group read-only access to the systemview view.

```
# group context sec.model sec.level prefix read write
notif
access notConfigGroup "" any noauth exact systemview
none none
```

1.2. System Contact Information

It is also possible to set the sysContact and sysLocation system variables through the snmpd.conf file:

```
syslocation Unknown (edit /etc/snmp/snmpd.conf)
```

```
syscontact Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
```

```
# Added for support of bcm5820 cards. pass .1 /usr/bin/ucd5820stat
```

1.3. Extra Information

See the snmpd.conf manual page, and the output of "snmpd -H".

```
trapcommunity public
```

```
trapsink default
```

Appendix B. Event Database Dictionary

1. Event Database Dictionary

Event Database Field	Description
dedupid	events will deduplicate based on the value of this field. by default: device, component, eventClass, eventKey, severity
device	name of device
component	name of component (like eth0, httpd, etc)
eclass	eventClass (if not specified maybe added by rule process if this fails will be /Unknown)
eventKey	If a component needs further deduplication specification this field maybe used
summary	message text truncated at 150 characters
message	full message text
severity	number from 0 to 5
eventState	state of event 0 = new, 1 = acknowledged, 2 = suppressed
eventClassKey	key by which rules processing begins. Often equal to component.
eventGroup	logical group of event source (syslog, ping, nteventlog etc)
stateChange	last time event changed automatically updated
firstTime	unix timestamp when event is received.
lastTime	last time an event was received
count	number of times an event has repeated
prodState	prodState of the device context
suppid	id of event that suppressed this event
manager	fqdn of the collector from which this event came
agent	collector name from which event came (zensyslog, zentrap, etc)
DeviceClass	device class from device context
Location	device location from device context
Systems	device systems from device context separated by |
DeviceGroups	device systems from device context separated by |
ipAddress	IP from which event came
facility	syslog facility of this is syslog event
priority	syslog priority of this is syslog event
ntevid	nt event id if this is nt eventlog event.

Appendix C. TALES Expressions

1. About TALES Expressions

TALES is syntax you can use to retrieve values call methods on Zenoss objects. Several fields in Zenoss accept TALES syntax, including command templates, event mapping transforms, user commands, event commands, zProperties, zLinks and others.

Commands (those associated with devices as well as those associated with events) can use TALES expressions to incorporate data from the related devices and/or events. TALES is a syntax for specifying expressions that let you access the attributes of certain objects such as a device or an event in Zenoss. For additional documentation on TALES syntax please see the TALES section at:

http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/AppendixC.stx

Depending on the context you may have access to a device and/or an event. Below is a list of the attributes and methods you may wish to use on device and event objects. The syntax for accessing device attributes and methods is ${dev/attributename}, so for example to get the manageIp of a device you would use ${dev/manageIp}. For events, the syntax is ${evt/attributename}

A command to ping a device might look like this. The ${..} is a TALES expression to get the manageIp value for the device.

```
{{{
ping -c 10 ${dev/manageIp}
}}}
```

More Examples:

DNS forward lookup

(assumes device/id is a resolvable name)

```
{{{
host ${device/id}
}}}
```

DNS reverse lookup

```
{{{
host ${device/manageIp}
}}}
```

SNMP Walk

```
{{{
snmpwalk -v1 -c${device/zSnmpCommunity} ${here/manageIp} system
}}}
```

To use these expressions effectively you need to know which objects, attributes and methods are available to you in which contexts. Usually there is a dev and/or device which allows you access the device in a particular context. Contexts related to a particular event usually have evt and/or event defined. Some available attributes for each of these classes are listed below. List items with parenthesis after them are methods and much have the parenthesis included in the TALES expression to function correctly.

2. TALES Device Attributes

Attribute	Description
getId	The primary means of identifying a device within Zenoss
getManageIp	The IP address used to contact the device in most situations

Attribute	Description
productionState	The production status of the device: Production, Pre Production, Test, Maintenance or Decommisioned. This attribute is a numeric value, use getProductionStateString for a textual representation.
getProductionStateString	Returns a textual representation of the productionState
snmpAgent	The agent returned from snmp collection
snmpDescr	The description returned by the snmp agent
snmpOid	The oid returned by the snmp agent
snmpContact	The contact returned by the snmp agent
snmpSysName	The system name returned by the snmp agent
snmpLocation	The location returned by the snmp agent
snmpLastCollection	When snmp collection was last performed on the device. This is a DateTime object.
getSnmpLastCollectionString	Textual representation of snmpLastCollection
rackSlot	The slot name/number in the rack where this physical device is installed
comments	User entered comments regarding the device
priority	A numeric value: 0 (Trivial), 1 (Lowest), 2 (Low), 3 (Normal), 4 (High), 5 (Highest)
getPriorityString	A textual representation of the priority
getHWManufacturerName	Name of the manufacturer of this hardware
getHWProductName	Name of this physical product
getHWProductKey	Used to associate this device with a hardware product class
getOSManufacturerName	Name of the manufacturer of this device's operating system.
getOSProductName	Name of the operating system running on this device.
getOSProductKey	Used to associate the operating system with a software product class
getHWSerialNumber	Serial number for this physical device
getLocationName	Name of the Location assigned to this device
getLocationLink	Link to the Zenoss page for the assigned Location
getSystemNames	A list of names of the Systems this device is associated with
getDeviceGroupNames	A list of names of the Groups this device is associated with
getOsVersion	Version of the operating system running on this device
getLastChangeString	When the last change was made to this device
getLastPollSnmpUpTime	Uptime returned from snmp
uptimeStr	Textual representation of the snmp uptime for this device
getPingStatusString	Textual representation of the ping status of the device
getSnmpStatusString	Textual representation of the SNMP status of the device

3. TALES Event Attributes

Attribute	Description
dedupid	A key used to correlate duplicate events
evid	A unique id for the event
device	The id of the associated device, if applicable
ipAddress	The IP Address of the associated device, if applicable
component	The component of the associated device, if applicable
eventClass	The event class associated with this device
eventGroup	logical group of event source (syslog, ping, nteventlog etc)
eventKey	The eventKey is the primary criteria for mapping events into event classes
facility	syslog facility of this is syslog event
severity	One of 0 (Clear), 1 (Debug), 2 (Info), 3 (Warning), 4 (Error) or 5 (Critical)
priority	syslog priority of this is syslog event
summary	Text description of the event
stateChange	When the mysql record for this event was last modified
firstTime	The first time this event was seen
lastTime	The last time this event was seen and it's count incremented
count	Number of times this event has been seen
prodState	prodState of the device context
manager	fqdn of the collector from which this event came
agent	collector name from which event came (zensyslog, zentrap, etc)

zProperties are also available for devices and events using the same syntax as above.

Appendix D. Device Preparation

1. Net-SNMP

By default Net-SNMP does not publish the full SNMP tree. Check to see if that is currently the case on a device and configure it correctly.

1. Confirm snmpd is running:

```
> snmpwalk -v1 -cpublic <your device name> system
```

2. Retrieve the IP table for the device with snmpwalk:

```
> snmpwalk -v1 -cpublic <your device name> ip
```

   Typical SNMP View:

```
view systemview included .1
view systemview included .1.3.6.1.2.1.25.1
access notConfigGroup "" any noauth exact systemview none none
```

2. SNMP V3 Support

Zenoss has initial support for SNMPv3 data collection.

The following new zProperties control the authentication and privacy of these requests:

- zSnmpAuthType: use either "MD5" or "SHA" signatures to authenticate SNMP requests
- zSnmpAuthPassword: the shared private key used for authentication. Must be at least 8 characters long.
- zSnmpPrivType: either "DES" or "AES" cryptographic algorithms.
- zSnmpPrivKey: the shared private key used for encrypting SNMP requests. Must be at least 8 characters long.
- zSnmpSecurityName: the Security Name (user) to use when making SNMPv3 requests.

If zSnmpPrivType and zSnmpPrivPassword are set, the message is sent with privacy and authentication. If only the zSnmpAuthType and zSnmpAuthPassword are set, the message is sent with Authentication but no Privacy. If neither the Priv or Auth values are set, the message is sent with no authentication or privacy. It is an error to set the PrivType and PrivPassword without also setting an AuthType and AuthPassword.

SNMPv3 encryption using the AES (Advanced Encryption Standard) algorithm is supported only if the host platform net-snmp library supports it.

At the time of this writing, RedHat 5 does not, Ubuntu 7.10 does not, but OpenSuSE 10.2 and the Zenoss Appliance do.

You can determine if your platform supports AES with the following test:

```
$ snmpwalk -x AES 2>&1 | head -1
```

If the response is:

```
"Invalid privacy protocol specified after -x flag: AES"
```

then your platform does *not* support AES encryption for SNMPv3.

If the response is:

```
"No hostname specified."
```

Then your platform *does* support AES.

SNMPv3 Traps are not supported by Zenoss in version 2.1.1.

3. Forwarding Syslog Messages from UNIX/Linux Devices

Zenoss has its own syslog server, zensyslog. Managed devices should point their syslog daemons to zenoss. To do this, edit the /etc/syslog.conf file and add an entry where 1.2.3.4 is the zensyslog IP.

1. Log on to the target device (as a super user).
2. Open /etc/syslog.conf file with a text editor (e.g VI).
3. Enter *.debug and press the Tab key. then enter the host name or IP address of the Zenoss server. See example below:

   ```
   *.debug @192.168.X.X
   ```

4. Save the file and exit the file editor program.
5. Restart the Syslog service using the command below:

   ```
   /etc.init.d/syslog restart
   ```

4. Forwarding Syslog Messages from a Cisco IOS Router

Here are some links to Cisco commands to turn on syslog. Typically, it’s easier to use syslog than SNMP traps from network devices. The most basic IOS command to send syslog messages is:

```
logging 1.2.3.4
```

4.1. Other Cisco Syslog Configurations

Here are some additional configurations for other Cisco devices. To set up these configurations follow the following steps using the configurations that follow below.

1. Log on to the target router.
2. Type the command enable at the prompt.
3. Once you are prompted for a password, enter the correct password.
4. Type the command config at the prompt.
5. Type the command terminal at the configuration prompt.
6. At the prompt, Set the Syslog forwarding mechanism. See example below:

   logging <IP address of the Zenoss server>

7. Exit out all the prompts to the main router prompt.

Catalyst

```
set logging server enable
set logging server 192.168.1.100
set logging level all 5
set logging server severity 6
```

Local Director

```
syslog output 20.5
no syslog console
syslog host 192.168.1.100
```

PIX Firewalls

```
logging on
logging standby
logging timestamp
logging trap notifications
logging facility 19
logging host inside 192.168.1.100
```

5. Forwarding Syslog Messages from a Cisco CatOS Switch

1. Log on to the target switch.
2. Type the command enable at the prompt.
3. Once you are prompted for a password, enter the correct password.
4. Set the Syslog forwarding mechanism. See example below:

   ```
   set logging server <IP address of the Zenoss server>
   ```

5. You can set the types of logging information that you want the switch to provide with the commands below as examples:

   ```
   set logging level mgmt 7 default
   set logging level sys 7 default
   set logging level filesys 7 default
   ```

6. Forwarding Syslog Messages using Syslog-ng

Here is an example for FreeBSD and Linux platforms.

1. Log on to the target device (as a super user)
2. Open /etc/syslog-ng/syslog-ng.conf file with a text editor (e.g VI).
3. Add source information to file. See example below:

   FreeBSD:

   ```
   source src { unix-dgram("/var/run/log"); internal ();};
   ```

   Linux: (will gather both system and kernel logs)

   ```
   source src {
   internal();
   unix-stream("/dev/log" keep-alive(yes) max-connections(100));
   pipe("/proc/kmsg");
   udp();
   };
   ```

4. Add destination information (in this case, the Zenoss server). See example below:

   ```
   log { source(src); destination(zenoss); };
   ```

Appendix E. ZenWebTx Twill Commands Reference

1. About Twill Commands

Twill is the language used by ZenWebTx to simulate user actions in a Web browser and to test pages retrieved by the simulation. The following sections list the Twill commands available for use in ZenWebTx data sources.

Note

For detailed information about ZenWebTX, see the section titled Synthetic Transactions (ZenWebTx).

Some Twill commands produce text output (see the section titled Display). These commands do not affect the execution of tests by ZenWebTx, and are useful in testing and debugging ZenWebTx data sources.

To see the output of commands that produce text output, click Test Twill Commands on the Script page of a ZenWebTx data source.

Twill commands are divided among the following categories:

- Browsing
- Assertions
- Display
- Forms
- Cookies
- Debugging
- Other commands

2. Browsing

- **go** `<URL>` - Visit the given URL.
- **back** - Return to the previous URL.
- **reload** - Reload the current URL.
- **follow** `<link name>` - Follow a link on the current page.

3. Assertions

- **code** `<code>` - Assert that the last page loaded had this HTTP status. For example, ``code 200`` asserts that the page loaded correctly.
- **find** `<regexp>` - Assert that the page contains this regular expression.
- **notfind** `<regexp>` - Assert that the page does not contain this regular expression.
- **url** `<regexp>` - Assert that the current URL matches the given regexp.
- **title** `<regexp>` - Assert that the title of this page matches this regular expression.

4. Display

- **echo** <string> - Echo the string to the screen.
- **redirect_output** <filename> - Append all Twill output to the given file.
- **reset_output** - Display all output to the screen.
- save_html [<filename>] - Save the current page's HTML to a file. If no filename is given, derive the filename from the URL.
- **show** - Show the current page's HTML.
- **showlinks** - Show all of the links on the current page.
- **showforms** - Show all of the forms on the current page.
- **showhistory** - Show the browser history.

5. Forms

- **submit *[**<n>**]*** - Click the nth submit button, if given; otherwise, submit via the last submission button clicked. If nothing is clicked, then use the first submit button on the form. See the section titled Details on Form Handling for more information.
- formvalue <formnum> <fieldname> <value> - Set the given field in the given form to the given value. For read-only form widgets and controls, the click may be recorded for use by submit, but the value is not changed unless the config command has changed the default behavior. See config and the section titled Details on Form Handling for more information on the formvalue command.

  For list widgets, you can use one of the following commands to select or de-select a particular value. To select a value:

  ```
  formvalue <formnum> <fieldname> +value
  ```

  To de-select a value:

  ```
  formvalue <formnum> <fieldname> -value
  ```

- fv - Abbreviation for the formvalue command.
- formaction <formnum> <action> - Change the form action URL to the given URL.
- fa - abbreviation for the fa command.
- formclear - Clear all values in the form.
- formfile <formspec> <fieldspec> <filename> [<content_type>]* - attach a file to a file upload button by filename.

6. Cookies

- save_cookies <filename> - Save the current cookie jar to a file.
- load_cookies <filename> - Replace the current cookie jar with the specified file contents.
- clear_cookies - Clear all of the current cookies.
- show_cookies - show all of the current cookies. Sometimes useful for debugging. See note under the Display heading above.

7. Debugging

debug <what> <level> - Turn on or off debugging/tracing for various functions.

Enter the command in the form:

```
debug <what> <level>
```

where <what> is one of these options:

- http - Show HTTP headers.
- equiv-refresh - Test HTTP EQUIV-REFRESH headers.
- twill - Show Twill commands.

and <level> is 0 (for off) or 1 (for on).

8. Other Commands

- tidy_ok - Check to see if the tidy command runs on this page without any errors or warnings.
- exit *[<code>]* - Exit with the given integer code, if specified. The value of <code> defaults to 0.
- run <command> - Execute the specified Python command.
- runfile <file1> [<file2> ...]* - Execute the specified files.
- agent - Set the browser's "User-agent" string.
- sleep [<seconds>] - sleep the given number of seconds. Defaults to 1 second.
- reset_browser - Reset the browser.
- extend_with <module> - Import commands from the specified Python module. This acts like ``from <module> import *`` does in Python.

  For example, a function ``fn`` in ``extmodule`` would be available as ``fn``. See *examples/extend_example.py* for an example.
- add_auth <realm> <uri> <user> <password> - Add HTTP Basic Authentication information for the given realm/URL combination.

  For example, “add_auth IdyllStuff http://www.idyll.org/ titus test” tells Twill that a request from the authentication realm "IdyllStuff" under http://www.idyll.org/ should be answered with username 'titus', password 'test'. If the 'with_default_realm' option is set to True, ignore 'realm'.
- config [<key> [<value>]] - Show/set configuration options.
- add_extra_headers <name> <value> - Add an extra HTTP header to each HTTP request.
- show_extra_headers - Show the headers being added to each HTTP request.
- clear_extra_headers - Clear the headers being added to each HTTP request.

9. Details on Form Handling

The formvalue (or fv) and submit commands rely on a certain amount of implicit cleverness to do their work. In odd situations, it is difficult to determine which form field formvalue will choose based on your field name, or which form and field submit is going to "click" on.

Example 1

Following is the pseudocode for how formvalue and submit determine which form to use (function 'twill.commands.browser.get_form')::

for each form on page:

if supplied regexp pattern matches the form name, select

if no form name, try converting to an integer N & using N-1 as

an index into the list or forms on the page (for example, form 1 is

the first form on the page).

Example 2

Following is the pseudocode for how formvalue and submit determine which form field to use (function `twill.commands.browser.get_form_field`)::

search current form for control name with exact match to fieldname;

if single (unique) match, select.

if no match, convert fieldname into a number and use as an index, if

possible.

if no match, search current form for control name with regexp match to fieldname;

if single (unique) match, select.

if *still* no match, look for exact matches to submit-button values.

if single (unique) match, select.

Example 3

Following is the pseudocode for `submit`::

if a form was _not_ previously selected by formvalue:

if there is only one form on the page, select it.

otherwise, fail.

if a field is not explicitly named:

if a submit button was "clicked" with formvalue, use it.

otherwise, use the first submit button on the form, if any.

otherwise:

find the field using the same rules as formvalue

finally, if a button has been picked, submit using it;

otherwise, submit without using a button

10. ZenWebTx Extensions to Twill

ZenWebTx adds several commands to the standard Twill vocabulary.

10.1. twilltiming

twilltiming sets timers in a set of Twill commands. If you then define a data point for this timer, you can graph and set thresholds on this timer value.

To use twilltiming, you must include the following line once, near the top of your ZenCommands:

```
extend_with_twilltiming
```

Then, use the following command to start a new timer:

```
startTimer myTimerName
```

and then, to output the value:

```
printTimer myTimerName
```

Timer values should be output only once. So, to output the time from the start of the script to more than one point in the script, you must use more than one timer. For example:

extend_with twilltiming

setTimer wwwZenossCom

setTimer bothPages

go http://www.zenoss.com

printTimer wwwZenossCom

setTimer communityPage

follow "Community"

printTimer communityPage

printTimer bothPages

To use these timers in Zenoss, create data points with the same name as the timers. In this example you could create data points named wwwZenossCom, communityPage, and bothPages. You can then use these data points in Zenoss thresholds and graph definitions.

10.2. twillextract

twillextract extracts numeric values from Web pages during the transaction. To use twillextract, include this command once near the top of your Twill commands:

```
extend_with_twillextract
```

Then, use the following command to match the given regular expression to the current page:

```
extract <dataName> <regularExpression>
```

The value 1 or 0 is assigned to dataName depending on whether the regular expression matched or not.

Additionally, you can use Python's regular expression substring-matching syntax to extract substrings of the matched text. For example, http://www.zenoss.com contains a copyright notice near the bottom that looks like "Copyright (c) 2005-2009 Zenoss, Inc." The following Twill commands use a regular expression to grab the second year from that notice:

extend_with twillextract

go http://www.zenoss.com

extract copyright "(?P<firstYear>[0-9]*)-(?P<secondYear>[0-9]*) Zenoss, Inc."

(?P<name>.....) is Python syntax for naming that particular part of the regular expression. The value extracted from that part of the matching text is given the name from the extract command, then a dash, then the name from the sub-pattern. In this example, copyright gets a value of 1 or 0 depending on whether the pattern was found on the page or not, and copyright-firstYear and copyright-secondYear get the values extracted from the matched text. To use these values in Zenoss you must create data points in the WebTx data source with the same name as those you used in the extract command. In this case you would create data points named copyright, copyright-firstYear and copyright-secondYear. You can then create graph definitions and thresholds for these data points.

10.3. twillxpathextract

Zenoss uses the twillxpathextract command to extract numeric values from XML documents. To use twillxpathextract, include the following command near the top of a Twill script:

```
extend_with_twillxpathextract
```

Note

By default, Zenoss includes twillxpathextract in each script.

Then, use the following command to match and extract data using the given xpath expression:

```
xpathextract <dataName> <xpath>
```

where xpathextract is the command name, <dataName> is the name of the data point to which the value will map, and <xpath> is the xpath used to retrieve the data.

When applied to an XML document, the xpath expression must return a numeric value. This value is then assigned to the dataName data point.

10.4. ignorescripts

ignorescripts strips javascript from visited pages before they are processed by Twill. Usually Twill ignores script tags; however, it is possible for scripts to include strings that Twill will interpret as HTML tags. Including the command extend_with ignorescripts near the top of your Twill commands will cause all script tags to be stripped, thereby avoiding this issue.

Appendix F. Basic Troubleshooting in Zenoss

1. How To Troubleshoot Zenoss Daemons

Help for figuring out why collectors are failing

If Zenoss behaves strangely or otherwise does not do what you expect it to do, you can enable debugging messages to help you diagnose problems.

To do so, edit the configuration file of the daemon where you want to enable debugging: Go to Settings/Daemons/edit config. Then change the line "level info" in the section eventlog to "level debug". Save, then restart the daemon (under tab "Daemons").

You should now see debug information in the daemon's logfile unter $ZENHOME/log.

Do not forget to turn off debugging when you are finished troubleshooting; the logfiles grow quickly in debug mode.

1.1. Identify the Problem

1.1.1. Check the logs

Your first step in troubleshooting a daemon is to look in the logs for errors. The Zenoss logs are located in in $ZENHOME/logs. You can view the logs via the web user interface under the About link.

1.1.2. Get more information

You can get additional logging by decreasing the verbose limit and running the daemon in foreground mode.

For example, this command will run zenperfsnmp:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10
```

See this page: http://www.zenoss.com/community/docs/howtos/zenoss-daemon-command-line-arguments/ for a list of command line options for the remaining Zenoss daemons.

You will get more (sometimes, a lot more) log messages, but zenperfsnmp will only perform a single scan of all the devices.

1.1.3. Find out what the program is _really_ doing

If a program is hanging, or not behaving as you expect, you can eavesdrop on what the program is asking the operating system to do on its behalf. This is a good way to determine if a helper program is failing, or system errors are not propagating up to the log file.

On Linux machines, the command to trace these system calls is "strace". On the appliance you can add strace to your system with:

```
$ conary update strace
```

Other posix-like operating systems have their own commands (truss on Solaris, dtrace on OS X). For example, you can verify that zenperfsnmp is really sending packets:

```
$ strace -f -e trace=sendto \
$ZENPERFSNMP/bin/zenperfsnmp run
```

Don't forget the largest and most complex Zenoss daemon: MySQL.

Check the version against the one needed by Zenoss. If you see "Lost Connection to Zenoss" in the dashboard, it is likely a MySQL connection problem.

1.2. Narrow the Problem

If you have managed to limit a problem to a single device, or simply suspect a device because it's running an odd configuration, or was recently added, most of the active collectors will allow you to scan a single device:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10 --device SomeDevice
```

If the problem is related to a long running server, you can ask that the server run in foreground mode, but continue with the normal endless cycle:

```
$ $ZENHOME/bin/zenperfsnmp run -v 10 --cycle
```

1.3. Look for Conflicts

Are you running more than one copy of a daemon? During debugging and before version 1.1, Zenoss could lose track of background processes.

Stop zenoss and look for stray processes:

```
$ ps auxww | grep /z
```

Are you resource limited? Is the file system full? Do you have free memory? My favorite tool for this is "top" under Linux:

```
$ top
```

This program will constantly update the display with a list of the most CPU hungry programs. You can also sort the list by memory usage.

1.4. Reproduce the Problem

The ability to reproduce a problem with a consistent set of steps will help enormously. Often the only way to find the problem is to use "binary search". You reproduce the problem and take away "half" of the configuration. Slowly you can reduce the "halves" that are causing the problem until a single element remains.

1.5. Getting Help Solving the Problem

1.5.1. Search the Zenoss Forums

Hopefully someone has seen a similar problem. Their pain can save you time and energy. Also, we are always looking for patterns across users and that will help narrow down an issue.

1.5.2. Report the Problem to Zenoss

Some problems should be reported, even in the absence of detailed information because they are almost certainly bugs.

- the python interpreter crashes (a segmentation fault, for example)
- a python trace in a log file
- a daemon regularly drops heartbeats
- a daemon's size grows over time to consume all resources