



# **Zenoss Core Administration**

Release 5.0.0

Zenoss, Inc.

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

# Zenoss Core Administration

Copyright © 2015 Zenoss, Inc. All rights reserved.

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

Flash is a registered trademark of Adobe Systems Incorporated.

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

Linux is a registered trademark of Linus Torvalds.

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

Sybase is a registered trademark of Sybase, Inc.

Tomcat is a trademark of the Apache Software Foundation.

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

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

Part Number: 1011.15.030

Zenoss, Inc.  
11305 Four Points Drive  
Bldg 1 - Suite 300  
Austin, Texas 78726

# Contents

|  |               |
|--|---------------|
| <b>Chapter 1: Using Zenoss Core.....</b>                                   | <b>6</b>      |
| Interface and Navigation.....  | 6             |
| Customizing the Dashboard.....   | 11            |
| Search.....  | 13            |
| Navigating the Event Console.....  | 14            |
| Running a Command from the UI.....   | 18            |
| Running a Command from the CLI.....  | 18            |
| Working with Triggers and Notifications.....                               | 19            |
| Advanced User Interface Configuration.....                                 | 31            |
| <br><b>Chapter 2: Adding, Discovering and Modeling Devices.....</b>        | <br><b>32</b> |
| Add a Single Device.....   | 32            |
| Add Multiple Devices.....  | 33            |
| Discovering Devices.....   | 33            |
| Modeling Devices.....  | 36            |
| About Modeler Plugins.....   | 39            |
| Debugging the Modeling Process.....  | 40            |
| <br><b>Chapter 3: Working with Devices.....</b>                            | <br><b>41</b> |
| Viewing the Device List.....   | 41            |
| Working with Devices.....  | 42            |
| Managing Devices and Device Attributes.....                                | 51            |
| <br><b>Chapter 4: Configuration Properties.....</b>                        | <br><b>55</b> |
| Configuration Properties Inheritance and Override.....                     | 55            |
| Configuration Property Types.....  | 58            |
| Device Configuration Properties.....                                       | 58            |
| Event Configuration Properties.....  | 66            |
| Network Configuration Properties.....                                      | 67            |
| <br><b>Chapter 5: Monitoring templates.....</b>                            | <br><b>69</b> |
| Creating Templates.....  | 69            |
| Renaming Templates.....  | 69            |
| Template Binding.....  | 70            |
| Example: Defining Templates in the Device Hierarchy.....                   | 71            |
| Example: Applying Templates to Multiple Areas in the Device Hierarchy..... | 71            |
| <br><b>Chapter 6: Basic Monitoring.....</b>                                | <br><b>72</b> |
| Availability Monitoring.....   | 72            |
| Monitoring Using ZenCommand.....   | 82            |
| SNMP Monitoring.....   | 84            |
| Monitoring Devices Remotely Through SSH.....                               | 84            |

|  |                |
|--|----------------|
| <b>Chapter 7: Performance Monitoring.....</b>                        | <b>87</b>      |
| About Monitoring Templates.....                                      | 88             |
| Template Binding.....  | 88             |
| Data Sources.....  | 89             |
| Data Points.....   | 90             |
| Data Point Aliases.....  | 91             |
| Thresholds.....  | 93             |
| Performance Graphs.....  | 96             |
| Performance Data Retention.....                                      | 98             |
| <br><b>Chapter 8: Event Management.....</b>                          | <br><b>100</b> |
| Basic Event Fields.....  | 100            |
| Other Fields.....  | 102            |
| Details.....   | 103            |
| De-Duplication.....  | 103            |
| Auto-Clear Correlation.....  | 104            |
| Event Consoles.....  | 105            |
| Event Sources.....   | 111            |
| Creating Events Manually.....  | 112            |
| Event Classes.....   | 113            |
| Mapping and Transformation.....                                      | 114            |
| Event Life Cycle.....  | 117            |
| Capturing Email Messages as Events.....                              | 119            |
| SNMP Traps and Event Transforms.....                                 | 120            |
| <br><b>Chapter 9: Production States and Maintenance Windows.....</b> | <br><b>125</b> |
| Production States.....   | 125            |
| Maintenance Windows.....   | 126            |
| <br><b>Chapter 10: Organizers and Path Navigation.....</b>           | <br><b>129</b> |
| Classes.....   | 129            |
| Groups.....  | 132            |
| Systems.....   | 132            |
| Locations.....   | 133            |
| Inheritance.....   | 137            |
| <br><b>Chapter 11: User Commands.....</b>                            | <br><b>138</b> |
| Defining Global User Commands.....                                   | 138            |
| Running Global User Commands.....                                    | 139            |
| Defining User Commands for a Single Device.....                      | 139            |
| Running User Commands for a Single Device.....                       | 140            |
| Defining User Commands for All Devices in an Organizer.....          | 140            |
| Running User Commands for Devices in an Organizer.....               | 141            |
| User Command Example: Echo Command.....                              | 141            |
| <br><b>Chapter 12: Managing Users.....</b>                           | <br><b>143</b> |
| Creating User Accounts.....  | 143            |
| Editing User Accounts.....   | 143            |

|   |            |
|---|------------|
| User Groups.....  | 146        |
| Roles.....  | 147        |
| Device Access Control Lists.....                            | 147        |
| <b>Chapter 13: Reporting.....</b>                           | <b>151</b> |
| Device Reports.....   | 151        |
| Event Reports.....  | 153        |
| Performance Reports.....                                    | 154        |
| <b>Chapter 14: ZenPacks.....</b>                            | <b>155</b> |
| Viewing ZenPacks.....                                       | 155        |
| ZenPack Information Resources.....                          | 156        |
| Installing and Upgrading ZenPacks.....                      | 156        |
| Removing a ZenPack.....                                     | 158        |
| <b>Chapter 15: General Administration and Settings.....</b> | <b>160</b> |
| Recovering from a POSKey error during login.....            | 160        |
| Events Settings.....  | 160        |
| Rebuilding the Events Index.....                            | 161        |
| Audit Logging.....  | 162        |
| Setting Portlet Permissions.....                            | 166        |
| Working with the Job Manager.....                           | 167        |
| Host Name Changes.....                                      | 169        |
| Versions and Update Checks.....                             | 169        |
| <b>Appendix A: SNMP Device Preparation.....</b>             | <b>170</b> |
| Net-SNMP.....   | 170        |
| SNMP V3 Support.....  | 170        |
| Community Information.....                                  | 171        |
| System Contact Information.....                             | 172        |
| Extra Information.....                                      | 172        |
| <b>Appendix B: Syslog Device Preparation.....</b>           | <b>173</b> |
| Forwarding Syslog Messages from UNIX/Linux Devices.....     | 173        |
| Forwarding Syslog Messages from a Cisco IOS Router.....     | 173        |
| Forwarding Syslog Messages from a Cisco CatOS Switch.....   | 174        |
| Forwarding Syslog Messages using Syslog-ng.....             | 174        |
| <b>Appendix C: TALEX Expressions.....</b>                   | <b>176</b> |
| Examples.....   | 176        |
| TALEX Device Attributes.....                                | 177        |
| Tales Event Attributes.....                                 | 178        |
| <b>Glossary of terms.....</b>                               | <b>180</b> |

# Using Zenoss Core

---

This guide provides an overview of Zenoss Core architecture and features, as well as procedures and examples to help use and configure the system.

## **Documentation feedback**

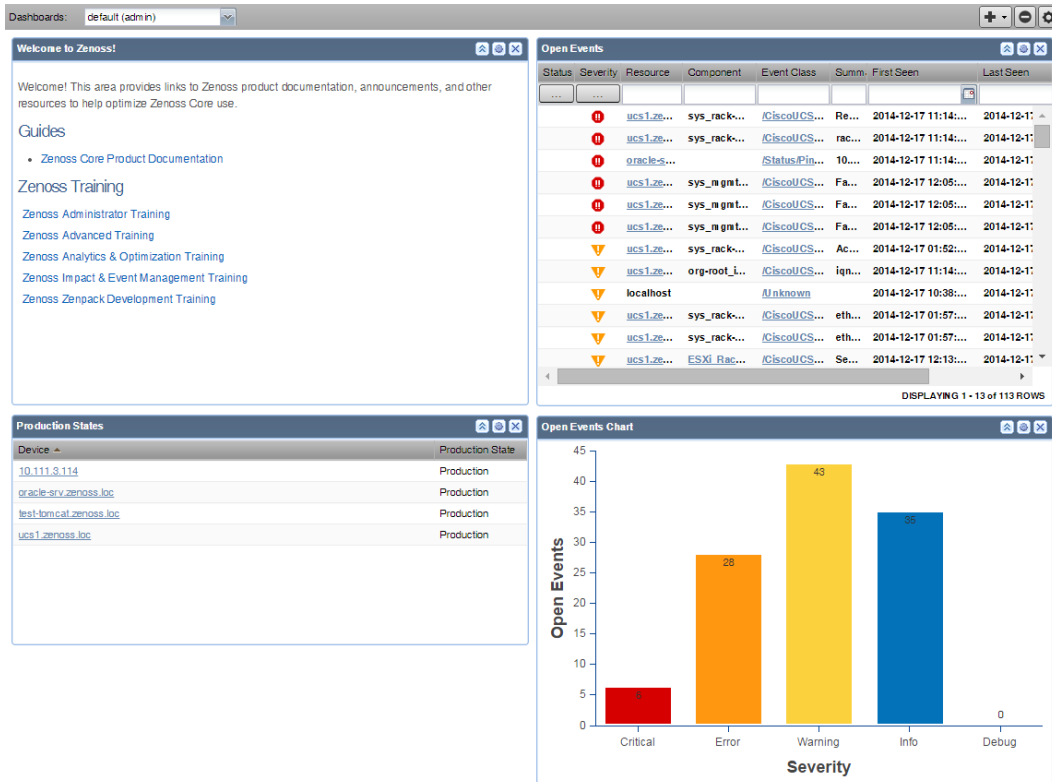
To provide technical feedback on this document, or to report an error or omission, please send your comments to [docs@zenoss.com](mailto:docs@zenoss.com). We appreciate your feedback.

## **Interface and Navigation**

---

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

Figure 1: Dashboard



The Dashboard can be customized to display a variety of information including:

- System information resources and other Web pages, such as internal portal pages
- Important error-level device events
- Geographical high-level view
- "Troubled" devices
- Devices in a certain production state

You can also create additional dashboards that can be cloned from existing dashboards and can also be restricted to certain users or user groups.

Key Dashboard and interface areas include:

- Navigation menu
- User information area
- Portlets
- System Network Map

## Navigation

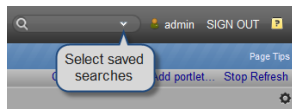
The Navigation menu lets you access major system features. In addition to the Dashboard, the menu is divided among several functional areas:

- **Events**- Guides you to the event management area, where you can monitor event status, triggers, and event transforms. You also can track changes made to events.
- **Infrastructure**- Offers access to network infrastructure, including, devices, networks, processes, and services.
- **Reports**- Allows you access to pre-defined and configurable reports.

- **Advanced**- Provides access to monitoring templates, collectors, MIBs, system settings, and tuning.

## User Information Area

**Figure 2: User Information Area**



The User information area offers information and selections:

- **Login ID**- The ID of the user currently logged in appears at the far left of this area. Click the ID to edit user settings, such as authentication information, roles, and groups. (You also can access user settings from the **Advanced > Settings > Users** page.)
- **Sign Out**- Click to log out of the system.
- **Help** icon - Click to access product documentation.

## Portlets

The main content of the Dashboard contains portlets, which provide information about the system and your infrastructure. Portlets can display:

- **Device Issues**- Displays a list of devices, associated with color-coded events of critical, error, or warning severity levels. Click a device name to view details, or click an event to go to the event console for the device.

**Figure 3: Device Issues Portlet**

| Device                      | Events                         |
|-----------------------------|--------------------------------|
| vpn-client18.zenoss.loc     | 0 Critical, 5 Warning, 0 Error |
| vpn-client12.zenoss.loc     | 1 Critical, 4 Warning, 0 Error |
| vpn-client10.zenoss.loc     | 1 Critical, 4 Warning, 0 Error |
| w2k8-dev-vm01.zenoss.loc    | 0 Critical, 4 Warning, 0 Error |
| vpn-client19.zenoss.loc     | 0 Critical, 4 Warning, 0 Error |
| vpn-client15.zenoss.loc     | 0 Critical, 4 Warning, 0 Error |
| vpn-client13.zenoss.loc     | 0 Critical, 4 Warning, 0 Error |
| test-rhel54-64-1.zenoss.loc | 0 Critical, 4 Warning, 0 Error |
| test-cent5-32-3.zenoss.loc  | 0 Critical, 4 Warning, 0 Error |

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

**Figure 4: Google Maps Portlet**



- **Daemon Processes Down**- Contains system self-monitoring information.
- **Production States**- Shows devices assigned to a particular production state. If needed, you can define multiple production states to display.

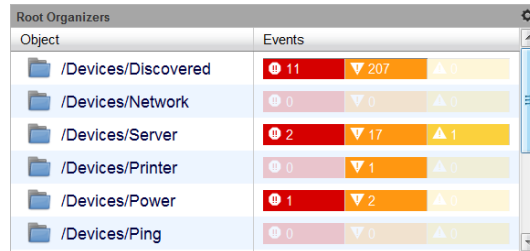


- **Site Window**- Initially provides links to resources such as product guides, forums, and training events.

You can customize the Site Window portlet to display content from any URL; however Zenoss recommends that you keep a portlet with the default URL so that you can stay up-to-date with Zenoss training and product updates. You can have multiple Site Window portlets defined on your Dashboard.

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

**Figure 5: Top Level Organizers Portlet**



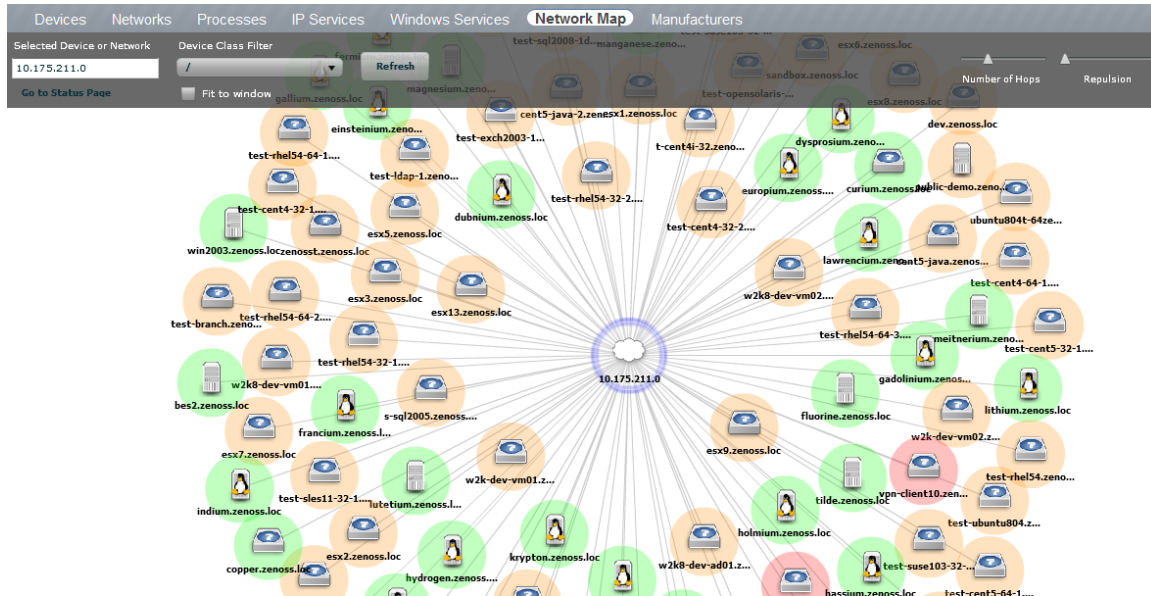
| Object              | Events  |
|---------------------|---|
| /Devices/Discovered | 11 (red circle), 207 (orange triangle down), 0 (yellow triangle up) |
| /Devices/Network    | 0 (red circle), 0 (orange triangle down), 0 (yellow triangle up)    |
| /Devices/Server     | 2 (red circle), 17 (orange triangle down), 1 (yellow triangle up)   |
| /Devices/Printer    | 0 (red circle), 1 (orange triangle down), 0 (yellow triangle up)    |
| /Devices/Power      | 1 (red circle), 2 (orange triangle down), 0 (yellow triangle up)    |
| /Devices/Ping       | 0 (red circle), 0 (orange triangle down), 0 (yellow triangle up)    |

- **Messages**- Displays system messages.
- **Object Watch List**- Allows the display of high-level status device classes, groups, systems, event classes, and locations that you select.
- **Device Chart**- Allows the display of a graph of multiple data points for a selected device class.
- **Event View**- Displays the list of events similar to the view on the Event console.
- **HTML Portlet**- Displays HTML content. You must use HTML markup in this portlet. If you want to populate a portlet with content from a given URL, use the Site Window portlet instead.
- **Network Map**- Displays a network map for a defined network being monitored. You can define the refresh interval and level of depth of your map.
- **Open Events Chart**- Displays a chart of the number of open events grouped by severity. You can define the event class to use as well as the number of past days for which to show events.
- **Watch List**- Allows the display of high-level status device classes, groups, systems, event classes, and locations that you select.

## Network Map

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

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

**Figure 6: Network Map**

### Choosing the Network to Display

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

### 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.

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

---

**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.

---

### 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 by sliding the counter.
- 3 Click **Refresh**.

### 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, then click **Refresh**.

## Adjusting Viewable Hops

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

## 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.

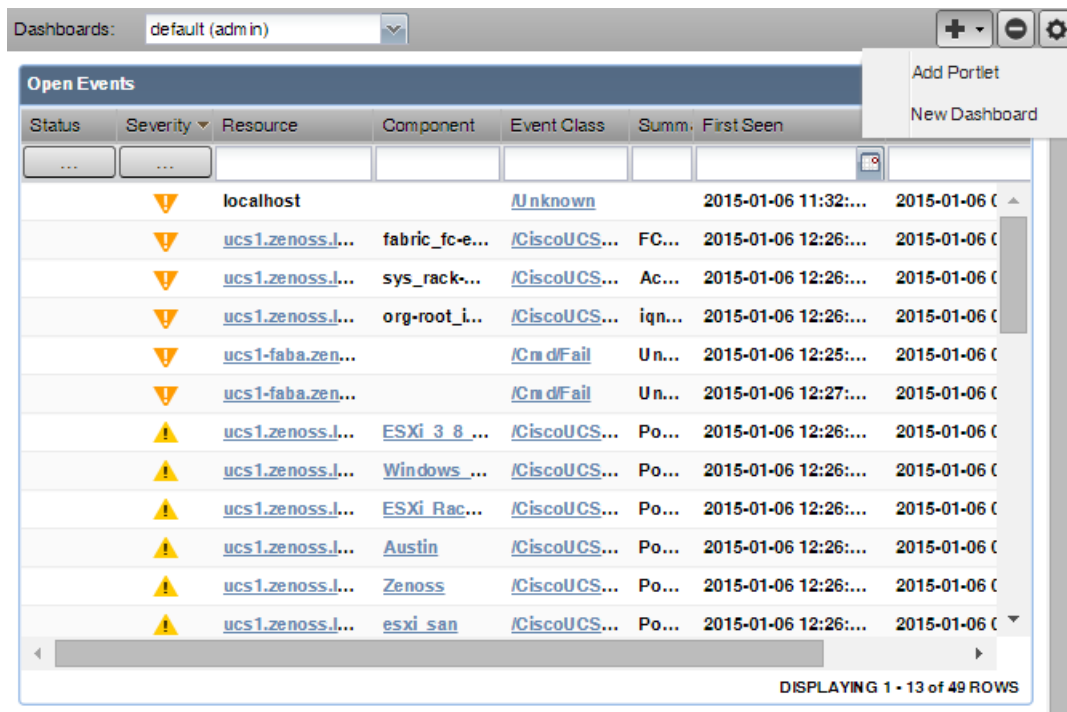
Select the **Fit to Window** option to bring all displayed icons into the viewable area.

## Customizing the Dashboard

You can customize the dashboard by:

- Creating multiple dashboards
- Selecting the portlets you want to view
- Arranging portlets
- Defining who can view the dashboard
- Changing the Dashboard column layout

The following screenshot of a sample dashboard shows the Add icon menu activated:



## Adding a New Dashboard

A default dashboard is created when you launch the system. This dashboard can be customized, but you may want to create other dashboards that display distinctive information or are targeted to a specific type of user (including just yourself). You can create as many additional dashboards as you choose. You can customize them by selecting who can view the dashboard as well as selecting and customizing portlets to display the most important information.

---

**Note** You cannot delete the default dashboard.

---

To create an additional dashboard:

- 1 From the **Add** icon on the Dashboard controls, select **New Dashboard**. The Add a New Dashboard dialog appears.

- 2 Enter a Dashboard Name. When this dashboard name is displayed in the Dashboards drop-down list, the user name who created it will be appended in parentheses as part of the name. This gives everyone who can see the dashboard an indication of who created it.
- 3 Select who can view this dashboard. If you want a User Group to view this dashboard, the group must already be created in the system and the user creating this new dashboard must be a member of the group. You cannot add a dashboard and assign it to a group you are not part of.
- 4 Select the number of columns to display in the dashboard. The default is 3.
- 5 If you want to clone the new dashboard from the previously viewed dashboard, select the check box. Otherwise, you will begin with a completely blank dashboard.
- 6 Click **Create**.

## Adding Portlets

You can customize your dashboard by adding portlets that display information you are interested in. Your dashboard can display more than one of the same portlet type. For example, you could have several Device Chart portlets with each one showing a different device class.

To add a portlet to the Dashboard:

- 1 Click the **Add** icon in the top right of the Dashboard main area and select **Add portlet**.
- 2 Select a portlet from the drop-down list.

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

## Arranging Portlets

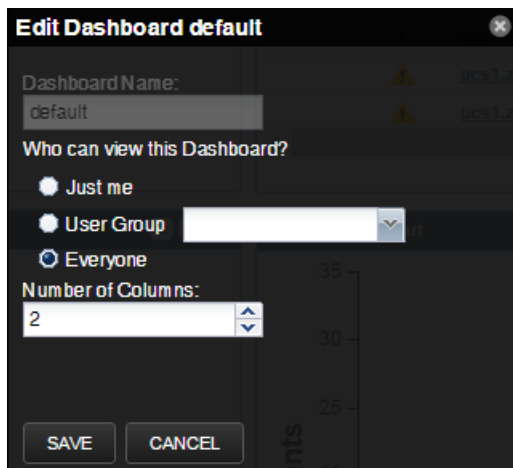
To arrange portlets, click the portlet header and drag the portlet to any location on the Dashboard. Other portlets rearrange depending on the location you drop it.

## Editing the Dashboard Settings

You can customize your dashboard to display a different number of columns or limit access to the dashboard.

To edit the dashboard settings:

- 1 Click the Action icon in the upper-right side of the Dashboard. The Edit Dashboard window appears.



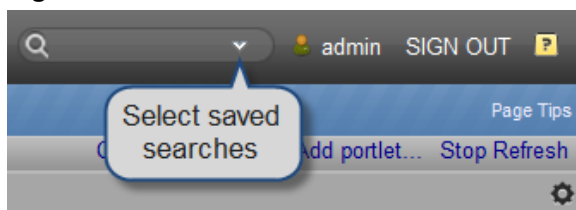
- 2 Change the value of the users who can view this dashboard, if needed. If you want a User Group to be able to view this dashboard, the group must already be defined in the system and the user editing the dashboard must be a member of the user group in order to see that group in the drop-down list.
- 3 Change the number of columns to use in this dashboard if needed.
- 4 Click **Save**.

## Search

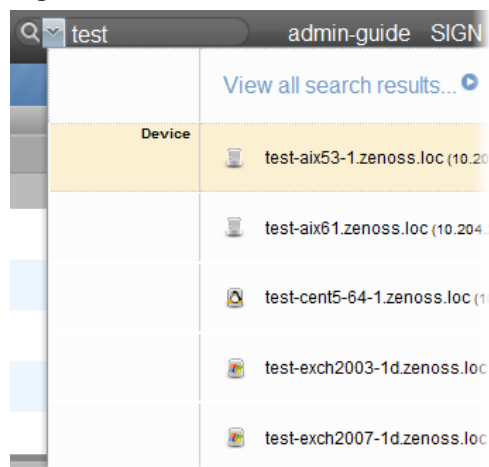
The Zenoss Core search facility supports locating devices and other system objects, as well as events and services.

In the Zenoss Core interface, the search feature is located adjacent to the user information area.

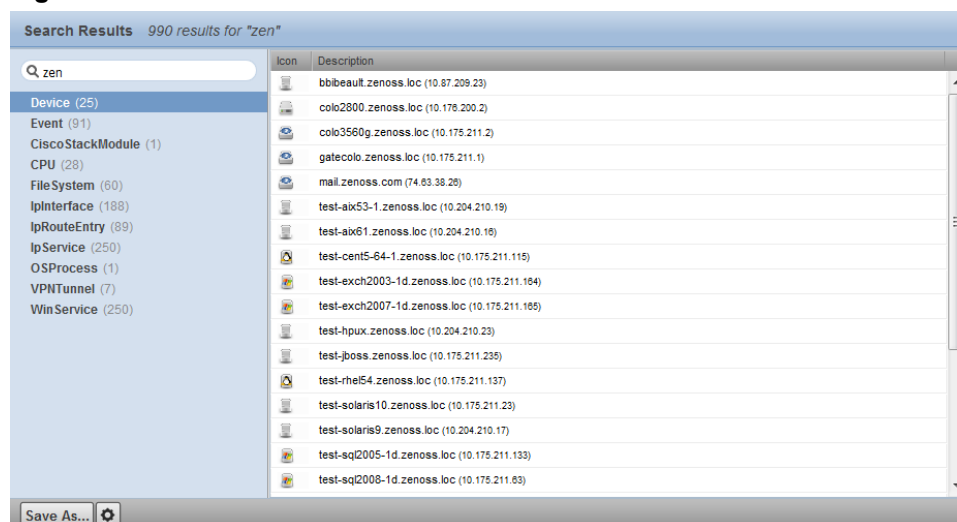
**Figure 7:** Search Field



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

**Figure 8: Search Results**

To view all search results, click the indicator at the top of the list.

**Figure 9: All Search Results**

From here, you can display search results by category. Click in the left panel to filter search results by a selection.

You can save the search to access later.

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

The Save Search As dialog appears.

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

You can access saved searches from:

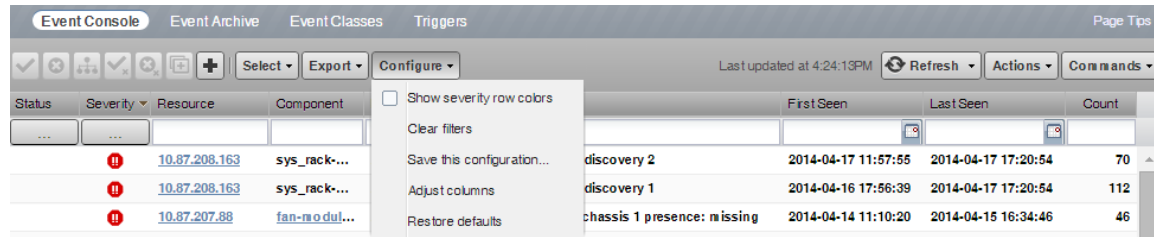
- Action menu located at the bottom of the Search Results page.
- Search box located at the top of the interface. Click the arrow, and then select **Manage Saved Searches**.

## Navigating the Event Console

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

To access the event console, click **Events** in the Navigation menu. The Event Console appears.

**Figure 10: Event Console**



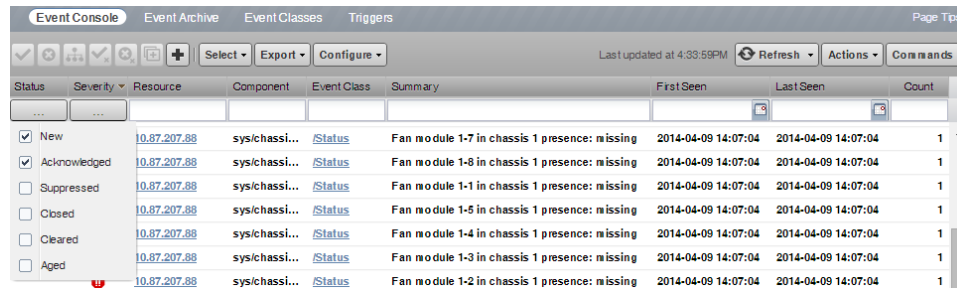
## Sorting and Filtering Events

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

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

Filter options appear below each column header. A match value can be any full string or a subset of a string, optionally with the wildcard (\*) contained in the values in that column. You can also use " | " (OR), or " ! ! " (NOT) expressions to further target your filters. For example, typing ! ! windows in the Event Class filter will return all the non-Windows device events.

**Figure 11: Event Console Filter Options**



You can filter the events that appear in the list in several ways, depending on the field type:

- **Resource** - Enter a match value to limit the list.
- **Component** - Enter a match value to limit the list.
- **Event Class** - Enter a match value to limit the list.
- **Summary** - Enter a match value to limit the list.
- **First Seen** - Enter a value or use a date selection tool to limit the list.
- **Last Seen** - Enter a value or use a date selection tool to limit the list.
- **Count** - Enter a value to filter the list, as follows:
  - *N* - Displays events with a count equal to *N*.
  - *:N* - Displays events with a count less than or equal to *N*.
  - *M:N* - Displays events with a count between *M* and *N* (inclusive).
  - *M:* - Displays events with a count greater than or equal to *M*.

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

You also can re-arrange the display order of columns in the event console. Click-and-drag column headers to change their display.

## Creating an Actionable View

For users that are not Administrators, there is an option that will filter the list of events to show only those that are not read-only for the user's permission level, and enable the action buttons above the event table header.

To turn on the actionable view, click **Configure** and select the **Only show actionable events** check box. The view is changed to show only events that can have an action performed on them based on the user's permission level. For more information, see [Managing Events](#) on page 17.

## Saving a Custom View

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

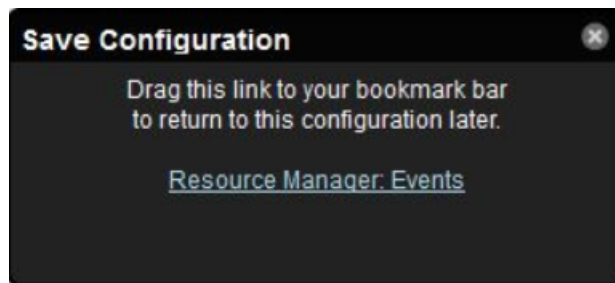
- 1 Select **Configure > Save this configuration**.

A dialog containing a link to the current view appears.

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

The system adds a link titled "Zenoss Core: Events" to your bookmarks list.

**Figure 12: Saving a Custom View (Bookmark)**




---

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

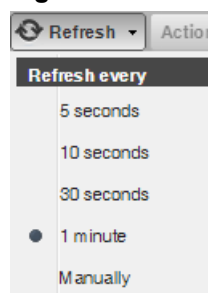
---

## Refreshing the View

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

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

**Figure 13: Automatic Refresh Selections**





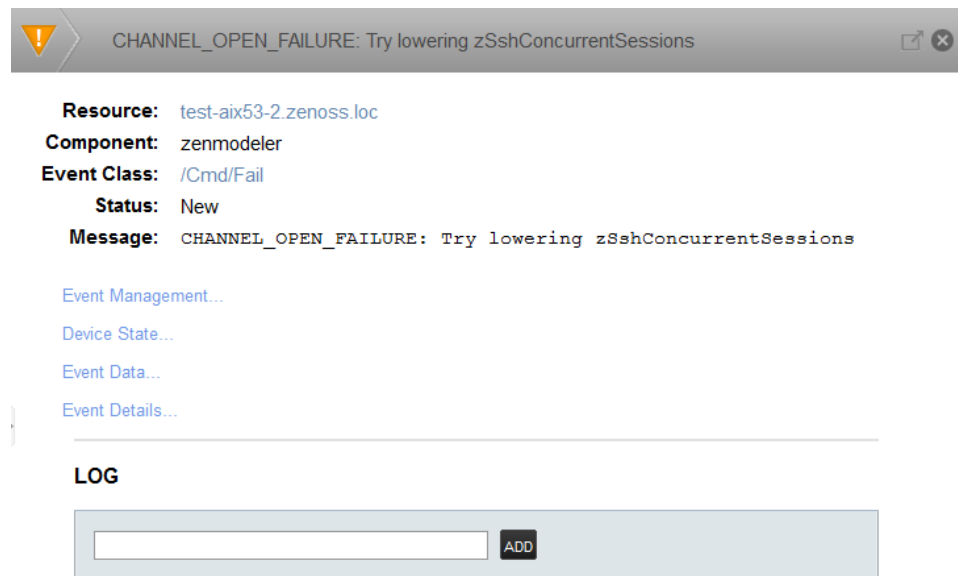
## Viewing Event Details

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

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

The Event Detail area appears.

**Figure 14: Event Detail**



To see more information about the event, click **Event Details**.

You can use the Log field (located at the bottom of the area) to add specific information about the event. Enter details, and then click **Add**.

## Selecting Events

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

- Click a row to select a single event.
- Ctrl-click rows to select multiple events, or Shift-click to select a range of events.
- Click **Select > All** to select all events.

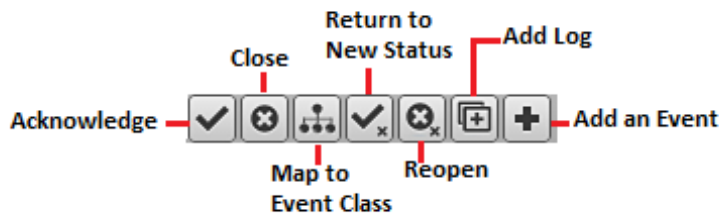
## Managing Events

You can manage events from the event console. After making a selection by clicking on the row of the event, you can:

- Acknowledge the event
- Close the event
- Reclassify the event, associating it with a specific event class
- Return the event to New status (revoke its Acknowledged status)
- Reopen the event
- Add a note to the log

You also can add an event from the event console. This feature is useful for testing a specific condition by simulating an event.

**Figure 15: Event Management Options**



## Running a Command from the UI

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

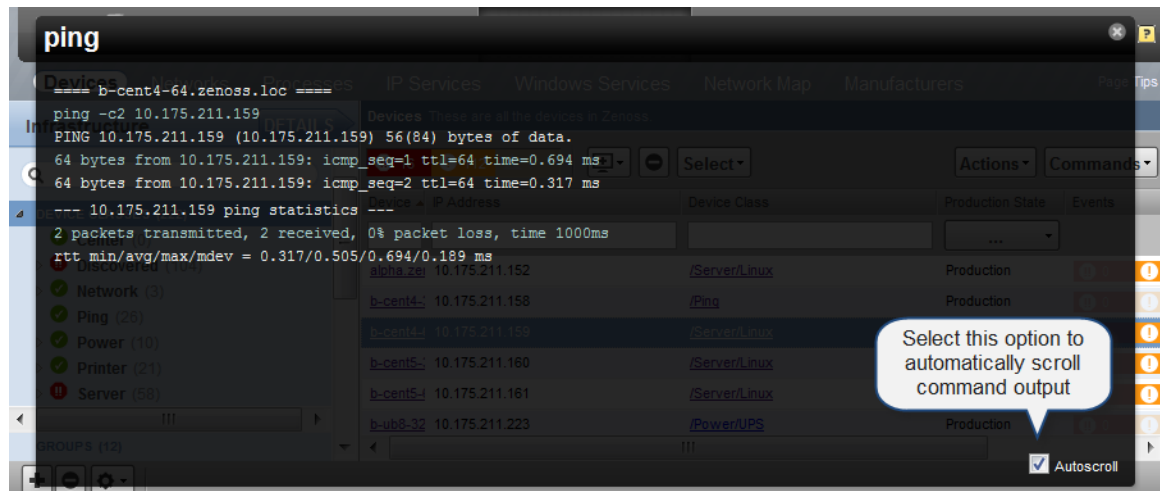
The system includes several built-in commands, such as `ping` and `traceroute`.

To run commands from the user interface:

- 1 Select one or more devices from the Devices list, which can be found under the **Infrastructure** menu. Do not click on a link within the row, just click anywhere else in the row to select the device.
- 2 Click **Commands** and select a command from the list.

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

**Figure 16: Command Output**



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

## Running a Command from the CLI

There are many commands in Zenoss Core that need to be run from the command-line interface (CLI). Since Zenoss Core uses containers to deploy the application, you must first attach to the container through the Zenoss Control Center CLI before using Zenoss Core commands that you see in this guide.

To use the Zenoss Control Center CLI, you need a Linux shell account on hosts in the Zenoss Control Center pool and the account needs to be a member of the `docker` group. Zenoss Control Center uses the `serviced` application service orchestrator to manage the container-based system. This technology is based on Docker.

The general syntax of a `serviced` command is:

```
serviced [global options] command [command options] [arguments...]
```

For more details on the `serviced` command and its syntax, see the Zenoss Control Center documentation by clicking the Help icon in that interface.

Some sample commands:

- `serviced service list` - lists all the running services (and the value of its `SERVICEID`) on the host (to avoid an error when invoking the same command on a host that is different than the master host, you can add the `--endpoint` flag.
- `serviced service stop zenhub` - stops the zenhub service
- `serviced service attach $SERVICEID bash` - attaches to a service identified by its `SERVICEID`.

## Working with Triggers and Notifications

---

You can create *notifications* to send email or pages, create SNMP traps, or execute arbitrary commands in response to an event. Notifications also can be used to notify other management systems, and to execute arbitrary commands to drive other types of integration. How and when a notification is sent is determined by a *trigger*, which specifies a rule comprising a series of one or more conditions.

To set up a notification, you must:

- Create a trigger, selecting the rules that define it
- Create a notification, selecting one or more triggers that cause it to run
- Choose appropriate options and subscribers, depending on the notification type

Read the following sections to learn about:

- Setting up triggers and trigger permissions
- Setting system SMTP settings for notifications
- Setting up notifications and notification permissions

## Working with Triggers

Setting up a trigger involves:

- Creating the trigger and the rules that define it
- Setting trigger permissions

### Creating a Trigger

To create a trigger:

- 1 Select **Events > Triggers** from the Navigation menu.

The Triggers page appears. It displays all existing triggers, indicating whether each is enabled.

- 2 Click the **Add** icon.

The Add Trigger dialog appears.

- 3 Enter a name for the trigger, and then click **Submit**. Be sure to only use letters, numbers, or the underscore character for the name. Do not use spaces or special characters.

The trigger is added to the list and is automatically enabled.

- 4 Double-click the trigger, or select the row of the trigger and click the **Action** icon to open the Edit Trigger dialog.

**Figure 17: Edit Trigger**

Enter information or make selections to define the trigger:

- **Enabled** - Select this option to enable the trigger.
- **Rule** - Define the rule comprising the trigger:
  - Select All or Any from the list to specify whether a notification will be triggered based on all, or any one, of the trigger rules.
  - Define the rule by making selections from each event field.

To add a rule to the trigger, click the **Add** icon.

Optionally, click the **Branch** icon to create a sub-branch of a given rule.

---

**Note** Be sure to take into account that a device production state may change during a maintenance window and if you haven't defined that a trigger should also fire during a production state of Maintenance, you may not get notification of a defined severity during the maintenance window.

---

## Setting Global Trigger Permissions

You can set global permissions for viewing, editing, and managing triggers. Global permissions are given to any user with "manage" permission, which includes:

- Admin, Manager, and ZenManager roles
- Trigger owner

Edit global permissions from the Users tab on the Edit Trigger dialog.

Global options are:

- **Everyone can view** - Provides global view permission.
- **Everyone can edit content** - Provides global update permission.
- **Everyone can manage users** - Provides global manage permission.

**Figure 18:** Edit Trigger - Users Tab

The screenshot shows the 'Edit Trigger' dialog box with the 'Users' tab selected. The 'Global Options' section includes three unchecked checkboxes: 'Everyone can view', 'Everyone can edit content', and 'Everyone can manage users'. The 'Users' section features a text input field, a dropdown arrow, an 'Add' button, and a minus icon. The bottom of the dialog has 'SUBMIT' and 'CANCEL' buttons.

### Setting Individual Trigger Permissions

You can grant permissions to individual users. For each user added, you can select:

- **Write** - Select this option to grant the user permission to update the trigger
- **Manage** - Select this option to grant the user permission to manage the trigger.

To set an individual's trigger permissions:

- 1 Select a user from the drop-down list in the Users section of the Edit Trigger dialog.
- 2 Click **Add**. The user is added.
- 3 Assign permissions by selecting the appropriate check box(es).
- 4 Optionally, add additional user trigger permissions by repeating this procedure.
- 5 When you are finished, click **Submit**.

To remove an individual's trigger permissions:

- 1 Select the row of the user's permissions.
- 2 Click the **Remove** icon.
- 3 Optionally, remove other user trigger permissions by repeating this procedure.
- 4 When you are finished, click **Submit**.

### Working with Notifications

Setting up a notification involves:

- Creating the notification
- Defining notification content (for email- or page-type notifications)
- Defining the SNMP trap host (for SNMP trap-type notifications)
- Defining commands to run (for command-type notifications)
- Setting notification permissions
- Setting up notification schedules

## Creating or Editing a Notification

To create or edit a notification:

- 1 Select **Events > Triggers** from the Navigation menu.
- 2 Select **Notifications** in the left panel.

The Notifications page appears.

**Figure 19: Notifications**

| Event Console | Event Archive  | Event Classes | Triggers               |             |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |
|---------------|--|---------------|------------------------|-------------|---------|--------|-------------|-----|------------|------------|---------|---|---|---------|----|-------|----|------|-----------|
| Triggers      | Notifications  |               | Notification Schedules |             |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |
| Notifications | <div><div><div>+</div><div>-</div><div>⚙</div></div></div> <table><tr><th>Enabled</th><th>ID</th><th>Trigger</th><th>Action</th><th>Subscribers</th></tr><tr><td>Yes</td><td>rancid-run</td><td>rancid-run</td><td>command</td><td>0</td></tr></table> |               | Enabled                | ID          | Trigger | Action | Subscribers | Yes | rancid-run | rancid-run | command | 0 | <div><div><div>+</div><div>-</div><div>⚙</div></div></div> <table><tr><th>Enabled</th><th>ID</th><th>Start</th></tr><tr><td>No</td><td>test</td><td>1/14/2015</td></tr></table> | Enabled | ID | Start | No | test | 1/14/2015 |
| Enabled       | ID   | Trigger       | Action                 | Subscribers |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |
| Yes           | rancid-run   | rancid-run    | command                | 0           |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |
| Enabled       | ID   | Start         |                        |             |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |
| No            | test   | 1/14/2015     |                        |             |         |        |             |     |            |            |         |   |   |         |    |       |    |      |           |

The Notifications area lists all defined notifications. For each notification, the area indicates whether the notification is enabled (Yes or No), the Action associated with the notification, and the number of notification subscribers.

To edit a notification, double-click it; or select it, and then click the **Action** icon.

To create a notification:

- a Click the **Add** icon.

The Add Notification dialog appears.

- b Enter a name for the notification.

---

**Note** Spaces are not allowed in a notification name.

---

- c Select an Action associated with the notification:

- **Command**- Allows the system to run arbitrary shell commands when events occur. Common uses of a Command notification include:
  - *Auto-remediation of events.* You can use SSH to remotely restart services on a UNIX system when they fail, or winexe to do the same for Windows services.
  - *Integration with external systems.* This includes opening tickets in your incident management system.
  - *Extending alerting mechanisms.* Zenoss Core supports email and pagers as alerting mechanisms "out of the box" through normal alerting rules.
- **Email** - Sends an HTML or text email message to authorized subscribers when an event matches a trigger rule.
- **Page** - Pages authorized subscribers when an event matches a trigger rule.
- **Syslog** - Sends a message to the syslog.
- **SNMP Trap** - Sends an SNMP trap when an event matches a trigger rule.

- d Click **Submit**.

- e Edit your newly created notification by double-click it or by selecting it and clicking the **Action** icon.

The Edit Notification dialog appears.

**Figure 20: Edit Notification**

**Edit Notification - ExampleNotification (email)**

**Notification** | Content | Subscribers

Enabled: ☐ Delay (seconds): 0

Send Clear: ☐ Repeat (seconds): 0

Send only on Initial Occurrence?: ☒

**Triggers**

Trigger:

SUBMIT CANCEL

On the Notification tab, you can select or set:

- **Enabled** - Select this option to enable the notification.
- **Send Clear** - Specify to send a notification when the problem has been resolved by a clear event.
- **Send only on Initial Occurrence** - Select this option to send the notification only on the first occurrence of the trigger.
- **Delay (seconds)** - Specify the minimum age (in seconds) of an event before the notification will be executed. You might want to set a delay to prevent notifications being sent for transient problems, or to prevent multiple notifications being sent for the same problem.

For example, if you have five events that come in and match the trigger in 45 seconds, specifying a delay of 60 seconds will ensure that only one notification is sent. Additionally, if you have an event that matches the trigger at 15 seconds and is later cleared by another event at 45 seconds, a delay of 60 seconds will prevent notifications being sent.

- **Repeat (seconds)** - Specify how often to repeat the notification until the event that triggered it is resolved.

## Defining Notification Content

To define notification content, click the **Content** tab of the notification.

For email-type notifications, you can use the default configuration for the following fields, or customize them to your needs:

- **Body Content Type** - Select HTML or text.
- **Message (Subject) Format** - Sent as the subject of the notification.
- **Body Format** - Sent in the notification.
- **Clear Message (Subject) Format** - Sent when a notification clears.
- **Body Format** - Sent when a notification clears.
- **From Address for Emails** - Sent as email address of sender
- **Various SMTP settings** - Used to define SMTP port, username, and password

**Figure 21:** Define Notification Content (Email)

**Edit Notification - ExampleServiceEmailNotification (email)**

Notification | **Content** | Subscribers

Body Content Type:

Message (Subject) Format:

Body Format:

```

Device: ${evt/device}
Component: ${evt/component}
Severity: ${evt/severity}
Time: ${evt/lastSeen}
Message:
${evt/message}
<a tal:attributes="href urls/eventUrl">Event Detail</a>
<a tal:attributes="href urls/ackUrl">Acknowledge</a>
<a tal:attributes="href urls/closeUrl">Close</a>
<a tal:attributes="href urls/eventsUrl">Device Events</a>
The following events are causes of this event:
<tal:block tal:repeat="item esa/causes">
Impact Chain: ${item/impactChain}
Device: ${item/evt/device}
Component: ${item/evt/component}
Severity: ${item/evt/severity}
Time: ${item/evt/lastSeen}
Message:
${item/evt/message}
<a tal:attributes="href item/urls/eventUrl">Event Detail</a>
<a tal:attributes="href item/urls/ackUrl">Acknowledge</a>
<a tal:attributes="href item/urls/closeUrl">Close</a>
<a tal:attributes="href item/urls/eventsUrl">Device
Events</a>
</tal:block>

```

Clear Message (Subject) Format:

Body Format:

```

Event: '${evt/summary}'
Cleared by: '${evt/id}'
At: ${evt/stateChange}
Device: ${evt/device}
Component: ${evt/component}
Severity: ${evt/severity}
Message:
${evt/message}
<a href="${urls/reopenUrl}">Reopen</a>

```

SUBMIT CANCEL

For page-type notifications, you can use the default configuration for the following fields, or customize them to your needs:

- **Message (Subject) Format** - Sent as the subject of the notification.
- **Clear Message (Subject) Format** - Sent when a notification clears.



**Figure 22:** Edit Notification Content (Page)

### Notification Content Variables

Within the body of your email, page, and command notifications, you can specify information about the current event, in the form:

```
'${objectname/objectattribute}'
```

**Note** Do not escape event command messages and event summaries. For example, write this command as: `${evt/summary}` (rather than `echo '${evt/summary}'`).

Object names may be `evt`, `evtSummary`, or `urls`; or for clearing event context, `clearEvt` and `clearEventSummary`. For each object name, the following lists show valid attributes (for example, `${evt/DevicePriority}`):

- `evt/` and `clearEvt/`
  - `DevicePriority`
  - `agent`
  - `clearid`
  - `component`
  - `count`
  - `created`
  - `dedupid`
  - `device`
  - `eventClass`
  - `eventClassKey`
  - `eventGroup`
  - `eventKey`
  - `eventState`

- evid
- facility
- firstTime
- ipAddress
- lastTime
- manager
- message
- ntevid
- ownerid
- priority
- prodState
- severity
- stateChange
- status
- summary

---

**Note** The `message` and `summary` names are, by default, wrapped in double quotes in event commands.

---

- `eventSummary/` and `clearEventSummary/`
  - uuid
  - occurrence
  - status
  - first\_seen\_time
  - status\_change\_time
  - last\_seen\_time
  - count
  - current\_user\_uuid
  - current\_user\_name
  - cleared\_by\_event\_uuid
  - notes
  - audit\_log
  - update\_time
  - created\_time
  - fingerprint
  - event\_class
  - event\_class\_key
  - event\_class\_mapping\_uuid
  - actor
  - summary
  - message
  - severity
  - event\_key
  - event\_group
  - agent
  - syslog\_priority
  - syslog\_facility
  - nt\_event\_code
  - monitor
  - tags

- `urls/`
  - `ackUrl`
  - `closeUrl`
  - `reopenUrl`
  - `eventUrl`
  - `eventsUrl`

ZenPacks also can define additional notification actions, and can extend the context available to notifications to add additional objects or attributes.

## Defining the SNMP Trap Host

For SNMP trap-type notifications, enter information or make selections on the Content tab of the notification:

- **SNMP Trap Destination-** Specify the host name or IP address where the trap should be sent.
- **SNMP Community-** Specify the SNMP community. By default, this is public.
- **SNMP Version-** Select v2c (default) or v3.
- **SNMP Port-** Specify the SNMP port. Typically, this is 162.

SNMP traps sent as a result of this notification are defined in the `ZENOSS-MIB` file. You can find this MIB file on any Zenoss Core server at `$ZENHOME/share/mibs/site/ZENOSS-MIB.txt`.

---

**Note** In order to execute a command using `$ZENHOME (/opt/zenoss` for the zenoss user), you must be attached to the container holding the Zenoss Core master host.

---

**Figure 23:** Edit Notification Content (SNMP Trap)

The screenshot shows a web-based configuration window titled "Edit Notification - test\_trap (trap)". It has three tabs: "Notification", "Content", and "Subscribers". The "Content" tab is active. The form contains the following fields:

- SNMP Trap Destination:** A text input field containing "traphost".
- SNMP Community:** A text input field containing "public".
- SNMP Version:** A dropdown menu with "v2c" selected.
- SNMP Port (usually 162):** A spinner control with "162" selected.

At the bottom of the window are two buttons: "SUBMIT" and "CANCEL".

## Defining Commands to Run

For Command-type notifications, you must specify the command to run when configured triggers are matched. Do this on the Content tab of the notification. Configure these fields:

- **Command Timeout** - By default, 60 seconds.
- **Command** - Command to run when a trigger is matched.
- **Clear Command** - Optional command to run when the triggering event clears.
- **Environment variables** -

**Figure 24:** Edit Notification Content (Command)

The screenshot shows a web interface for editing a notification. The title bar reads 'Edit Notification - command\_notification (command)'. Below the title bar are three tabs: 'Notification', 'Content', and 'Subscribers'. The 'Content' tab is selected. The main area contains four input fields: 'Command Timeout (seconds):' with a value of 60, 'Command:', 'Clear Command:', and 'Environment variables:'. At the bottom are 'SUBMIT' and 'CANCEL' buttons.

### Global Notification Permissions

By establishing permissions, you can control which users have the ability to view, manage, and update notifications. Permissions are granted based on the user's assigned role. The following table lists account roles and their associated notification permissions:

| Role   | Permissions  |
|--|--|
| Admin, Manager, ZenManager   | Users assigned the Admin, Manager, or ZenManager roles can view, update, and manage any notification.  |
| notification owner   | When a user creates a notification, he is designated the owner of that notification. During the life of the notification, the owner can view, update, and manage it. |
| all other users (including those assigned ZenUser and ZenOperator roles) | Must be specifically granted permissions through the interface to view, edit, or manage notifications.   |

You can set global permissions for viewing, updating and managing a notification. Global permissions are given to any user with "manage" permission, which includes:

- Admin, Manager, and ZenManager roles
- Notification owner

Edit global permissions from the Subscribers tab on the Edit Notification Subscription panel.

Global options are:

- **Everyone can view** - Provides global view permission.
- **Everyone can edit content** - Provides global update permission.
- **Everyone can manage subscriptions** - Provides global manage permission.

Permission checks occur when the data is sent to the browser and when any action occurs. To determine where a user can make modifications to a particular tab, permission checks are performed on global roles, then managerial roles, and then individual roles. Any role that provides the required permission will allow that permission's associated behavior.

**Figure 25: Edit Notification**

| Type | Subscribers  | Write                    | Manage                   |
|------|--------------|--------------------------|--------------------------|
| user | admin (User) | <input type="checkbox"/> | <input type="checkbox"/> |

### Setting Individual Notification Permissions

You can grant permissions to individual users or groups. For each user or group added, you can select:

- **Write** - Select this option to grant the user or group permission to update the notification.
- **Manage** - Select this option to grant the user or group permission to manage the notification.

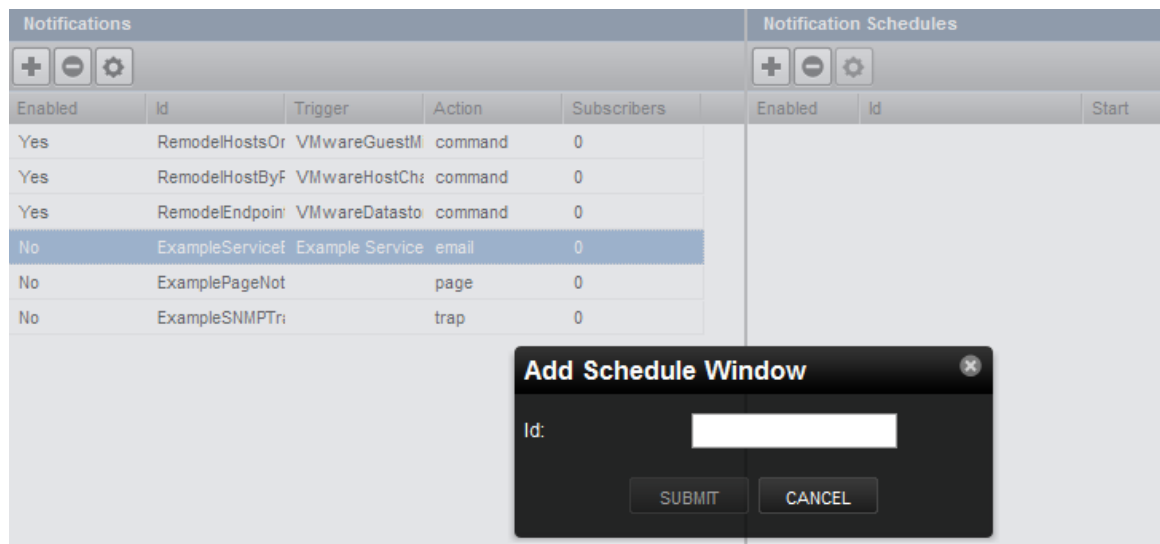
You can manually enter in the name of a user or group, or select one from the list of options.

### Setting Up Notification Schedules

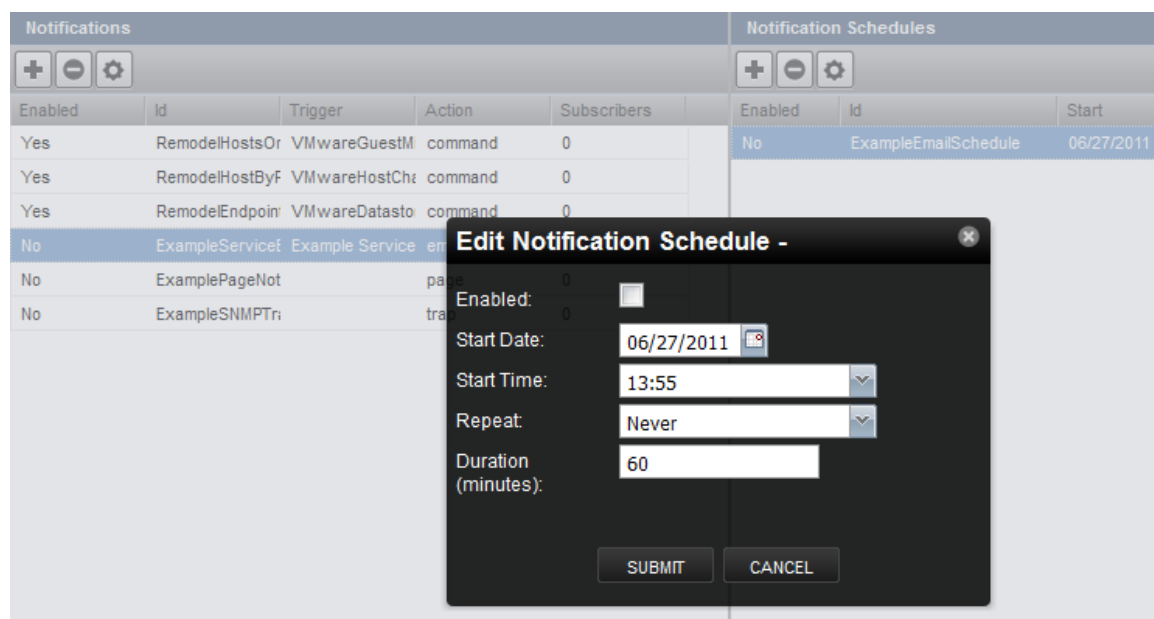
You can establish one or more notification schedules for each defined notification. To set up a schedule:

- 1 Select the notification in the Notifications area.
- 2 Click Add in the Notification Schedules area.

The Add Schedule Window dialog appears.

**Figure 26: Add Notification Schedule**

- 3 Enter a schedule ID, and then click **Submit**.
- 4 Double-click the newly added schedule to edit it. Select or enter values for the following fields:
  - **Enabled**- Select to enable the schedule. By default, this schedule is not enabled.
  - **Start Date**- Enter or select a start date for the schedule.
  - **Start Time**- Enter or Select a start time for the schedule.
  - **Repeat**- Select a schedule repeat value: Never, Daily, Every Weekday, Weekly, Monthly, or First Sunday of the Month.
  - **Duration (Minutes)**- Enter a schedule duration, which is the period of time that the notification window is active. If a notification has notification windows specified, then notifications are sent only if one of the windows is active when the notification is received.
- 5 Click **Submit**.

**Figure 27: Edit Notification Schedule**

## Advanced User Interface Configuration

---

To access advanced user interface configuration options, select **Advanced > Settings**, and then select **User Interface** in the left panel. This area lets you configure options such as how data loads, how much data is loaded, and filter and search options.

Select options or enter information:

- **Enable Infinite Grids for Events**- Disable this option to turn off infinite grids. By default, this feature is enabled.
- **Enable Live Search**- Disable this option to turn off the live search feature. By default, live search is enabled.
- **Enable Live Filters**- Disable this option to turn off the live filters feature. If you disable this feature, you will need to press enter on all search and filter inputs.
- **Enable Incremental Tree Loading on the Infrastructure Page**- Enable this option to load the infrastructure tree one node at a time. If disabled (the default), then the infrastructure tree is loaded all at once. You might enable this option if you have a complex hierarchy of organizers and device classes and want to improve your UI load response time.
- **Show Tree Event Severity Icons**- Disabling this option may speed up page loading.
- **Enable Tree Filters**- If disabled, then tree filters (the text input area that allows you to filter the information displayed) are hidden on all pages. This option is enabled by default.
- **Show Page Statistics**- Enable this option to show debugging information. By default, this option is not enabled.
- **Device Grid Buffer Size**- Specify the number of device data rows to fetch from the server for each buffer request. The default buffer size is 100 rows.
- **Component Grid Buffer Size**- Specify the number of component data rows to fetch from the server for each buffer request. The default buffer size is 50 rows.
- **Event Console Buffer Size**- Specify the number of event rows to fetch from the server for each buffer request. The default buffer size is 200 rows.
- **Device Move Job Threshold**- Specify the limit at which devices are moved immediately. If the number of devices to be moved exceeds this threshold, then the move occurs in a job; otherwise, they are moved immediately. The default value is 5 devices.
- **Job Notification Refresh Interval**- Specify the refresh interval for the job notification dialog. The default time is 10 seconds.
- **Job Grid Buffer Size**- Specify the number of job data rows to fetch from the server for each buffer request. The default buffer size is 100 rows.

When complete, click **Save** to save your changes.

## 2

## Adding, Discovering and Modeling Devices

Modeling is the process by which the system:

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

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

### Add a Single Device

Follow these steps to add a single device:

- 1 From the Navigation menu, select Infrastructure.

The Devices page appears.

- 2 Select Add a Single Device from the Add Devices icon.

The Add a Single Device dialog appears.

**Figure 28: Add a Single Device**

| IP Address    | Device Class | Production State | Priority |
|---------------|--------------|------------------|----------|
| 10.212.0.1    |              | Production       |          |
| 10.212.0.2    |              | Production       |          |
| 10.212.0.3    |              | Production       |          |
| 10.212.0.4    |              | Production       |          |
| 10.212.0.5    |              | Production       |          |
| 10.212.0.6    | PowerNet/Net | Production       |          |
| 10.212.0.7    | Printer      | Production       |          |
| 10.212.0.8    | Printer      | Production       |          |
| 10.212.0.9    | Printer      | Production       |          |
| 10.175.211.49 | Printer      | Production       |          |

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

- **Name or IP** - Enter the network (DNS) name or IP address of the device.
- **Device Class** - Select a device class to which this device will belong. For example, if the new device is a Windows server, then select /Server/Windows/WMI.
- **Collector** - By default, this is localhost. Select a collector for the device.



- **Model** - By default, this option is selected. De-select this option if you do not want the device to be modeled when it is added.
- 4 Optionally, click **More** to display additional fields. From the expanded page, you can:
    - Enter device-specific details
    - Edit SNMP settings
    - Set hardware and operating system information
    - Add device comments

---

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

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

---

- 5 Click **Add**.

---

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

---

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

## Add Multiple Devices

---

Follow these steps to manually add multiple devices:


- 1 From the Navigation menu, select **Infrastructure**.  
The **Devices** page appears.
- 2 Select **Add Multiple Devices** from the **Add Devices** icon.  
The **Add Infrastructure** page appears.
- 3 Select the category, type, and connection information for each device you want to add.
- 4 Click **Add** to add the device to the system. You will see the status at the bottom of the page. Continue to add more devices until you are done.
- 5 When you have completed adding your devices, click **Done**.

## Discovering Devices

---

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

Follow these steps to discover devices:

- 1 From the Navigation menu, select **Infrastructure**. The **Devices** page appears.
- 2 Click the **Add Devices** icon  and select **Discover Networks** from the drop-down list. The **Network Discovery** page appears.

## Network Discovery

*Devices found via Discovery will be placed in the /Discovered Device Class*

| Networks/Range   | SNMP                                       | SSH Authentication                | Windows Authentication                          |
|--|--|-----------------------------------|---|
| Enter one or more networks (such as 10.0.0.0/24) or IP ranges (such as 10.0.0.1-50):<br><input type="text"/> | Community Strings:<br><input type="text"/> | Username:<br><input type="text"/> | Administrator Username:<br><input type="text"/> |
|  | <input type="text"/>                       | Password:<br><input type="text"/> | Password:<br><input type="text"/>               |
| <input type="button" value="Discover"/>  |  |                                   |   |

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

192.0.2.0/24

or a range of IP addresses:

192.0.2.1-50

---

**Note** Trying to add a /16 or /8 network can take a very long time, and may have unintended consequences.

---

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

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

---

**Note** Zenoss Core uses Advanced Encryption Standard (AES) with a 256-bit key size to encrypt all passwords, and stores them in the Zope object database.

---

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

## Classifying Discovered Devices

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

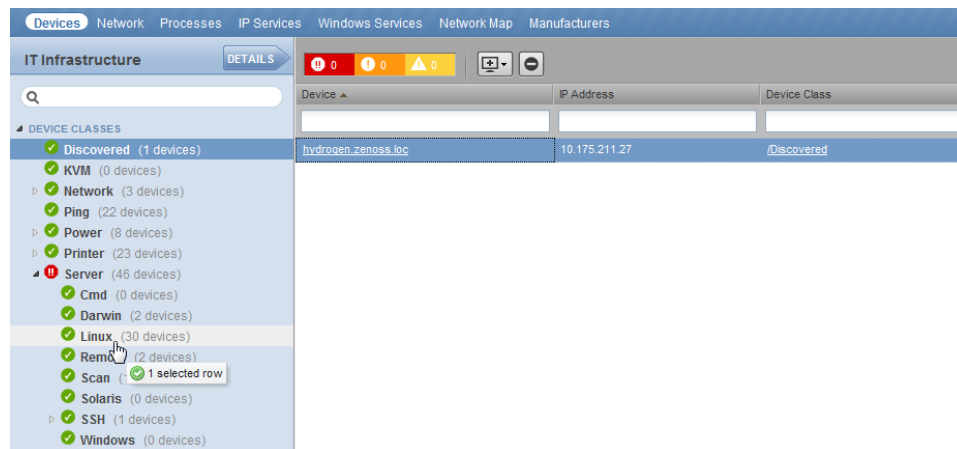
Servers are organized by operating system. If the system discovers Windows devices, for example, you might choose to relocate them to /Server/Windows. Similarly, you might choose to classify discovered Linux devices in /Server/Linux (if you want to monitor and model using SNMP), or /Server/SSH/Linux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

To classify discovered devices:

- Select one or more discovered devices (highlight one or more rows) in the device list.

- 2 Drag the selected devices to the new device class in the tree view.

**Figure 29: Classifying Discovered Devices**



The Move Devices dialog appears.

- 3 Click **OK**.

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

## Updating Device Authentication Details

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

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

- 1 Click a device name in the devices list.

The Device summary page appears.

- 2 Select Configuration Properties from the left panel.
- 3 Double-click the zWinRMUser configuration property to display the Edit Config Property dialog.
- 4 Enter your Windows user name in the Value field, and then click **Submit**.
- 5 Double-click the zWinRMPassword configuration property to display the Edit Config Property dialog.
- 6 Enter your Windows password in the Value field, and then click **Submit**.

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

---

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

---



---

**Note** Zenoss Core uses Advanced Encryption Standard (AES) with a 256-bit key size to encrypt all passwords, and stores them in the Zope object database.

---

## Adding or Editing Information on a Device Record

You may want to add or edit details about a discovered device.

To add or edit information:

- 1 Click a device name in the devices list. The Device overview page appears.
- 2 You can select values to change, or click the "edit" link adjacent to a label to edit that value. Enter or change information in one or more areas, and then click **Save** to save your changes.

## Modeling Devices

To model devices, the system can use:

- SNMP
- SSH
- WinRM
- Telnet

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

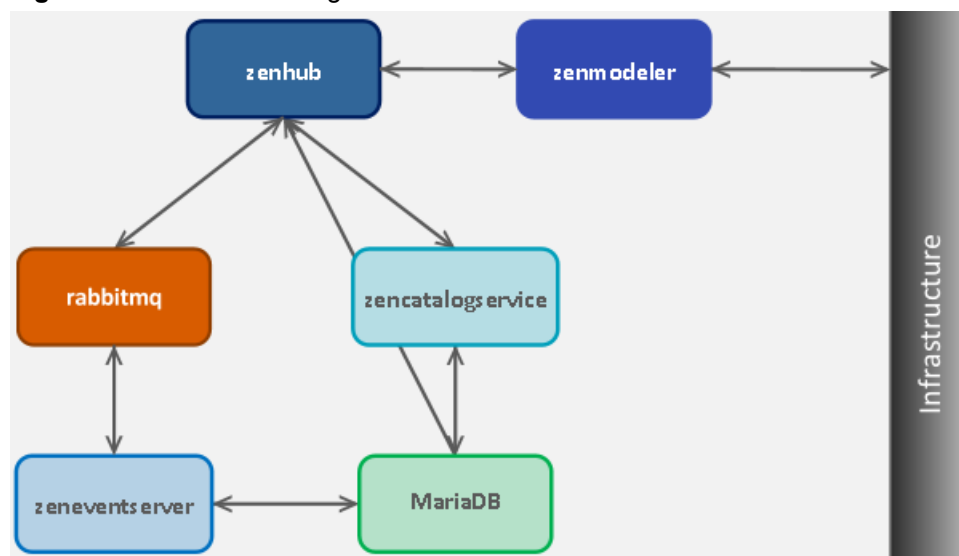
By default the system remodels each known device every 720 minutes (12 hours).

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

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

The following figure shows how the `zenmodeler` daemon fits into the modeling devices portion of the Zenoss Core architecture. Note that MariaDB has replaced ZenDS (MySQL) in version 5 and stores the object database (ZODB).

**Figure 30: Device Modeling Architecture**



## Configuring Windows Devices to Provide Data Through SNMP

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

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

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

To collect Windows event logs or log files from a Windows box using syslog, you can use the SyslogAgent Windows add-on, available from:

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

## Configuring Linux Devices to Provide Data Through SNMP


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

## Modeling Devices Using SSH/CMD

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

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

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

- 1 From the Navigation menu, select **Infrastructure**.
- 2 Click the Add Devices  icon and select **Add Device** from the drop-down list. The Add Devices window appears.

The Add a Single Device dialog appears.

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

The Device Overview page appears.

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

---

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

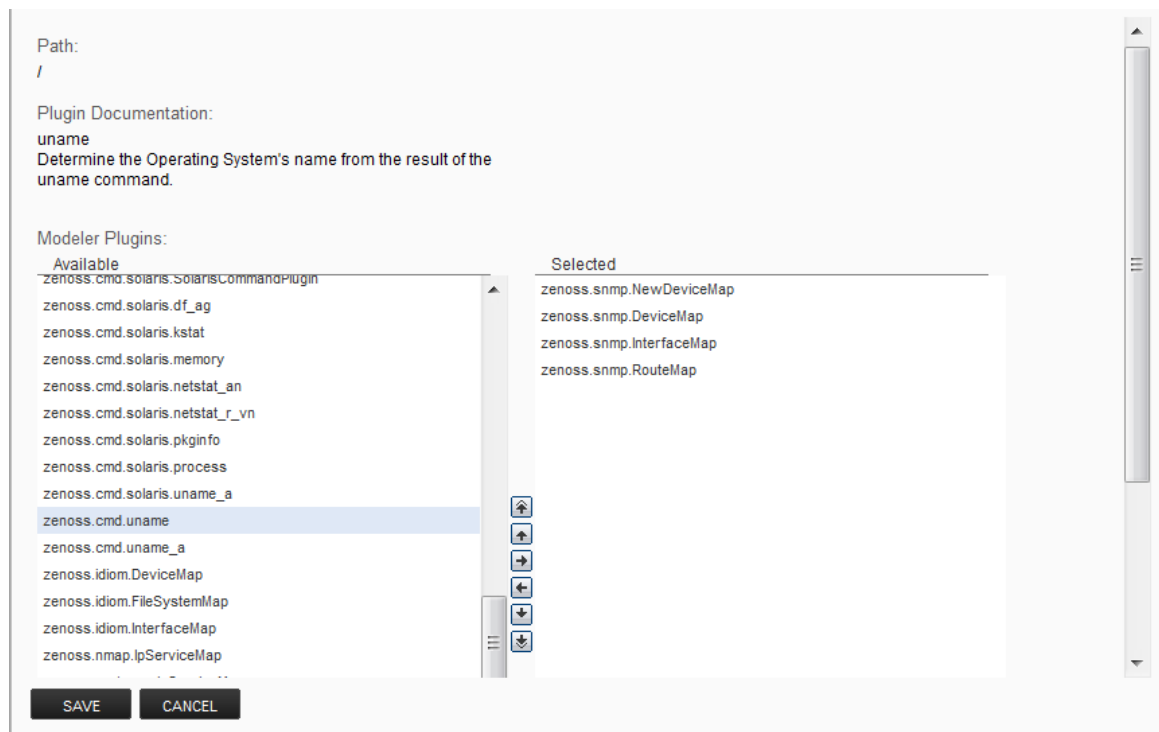
```
~/.ssh/id_rsa
```

---

- 9 In the left panel, select **Modeler Plugins**.

The list of plugins appears. The left column displays available plugins; the right column shows those currently selected.

- 10 Select `zenoss.cmd.uname` from the Available list, and then use the right arrow control to move it to the Selected list on the right. Use the controls to place it at the top of the list.

**Figure 31: Add Plugin**

- 11 Use the left arrow control to move the other Selected plugins from the Selected list to the Available list.
- 12 Click **Save**.
- 13 Model the device by clicking the **Model Device** button.

## Using Device Class to Monitor Devices Using SSH

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

## Modeling Devices Using Port Scan

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

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

```
which nmap
```

If your system returns a result similar to:

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

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

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

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

---

**Note** In order to execute a command using \$ZENHOME (/opt/zenoss for the zenoss user), you must be attached to the container holding the Zenoss Core master host.

---

To model a device using a port scan:

- 1 Select the device in the device list.
- 2 In the left panel, select **Modeler Plugins**.
- 3 Select the zenoss.nmap.ipServiceMap plugin in the list of Available plugins, and then use the right arrow control to move it to the list of Selected plugins.
- 4 Click **Save**.
- 5 Remodel the device by clicking the **Model Device** button.

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

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

## About Modeler Plugins

---

Zenoss Core uses plug-in maps to map real world information into the standard model. Input to the plug-ins can come from SNMP, SSH or Telnet. Selection of plug-ins to run against a device is done by matching the plug-in name against the zCollectorPlugins configuration property.

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

## Viewing and Editing Modeler Plugins for a Device

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

- 1 Click the device name in the devices list.
- The Device summary page appears.
- 2 In the left panel, select **Modeler Plugins**.

The Modeler Plugins page appears.

### Adding Plugins

To add a plugin to a device:

- 1 Use the right arrow control to move one or more plugins from the Available list (on the left) to the Selected list (on the right).
- 2 Click **Save**.

### Reordering Plugins

Plugins run in the order in which they are listed. To re-order plugins, use the up and down arrow controls, and then click **Save**.

### Deleting Plugins from a Device

To delete a plugin from a device, use the left arrow control to move the plugin from the Selected list to the Available list.

## Debugging the Modeling Process

---

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

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

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

If the command returns any stack traces, check the community forums for assistance.



## 3

# Working with Devices

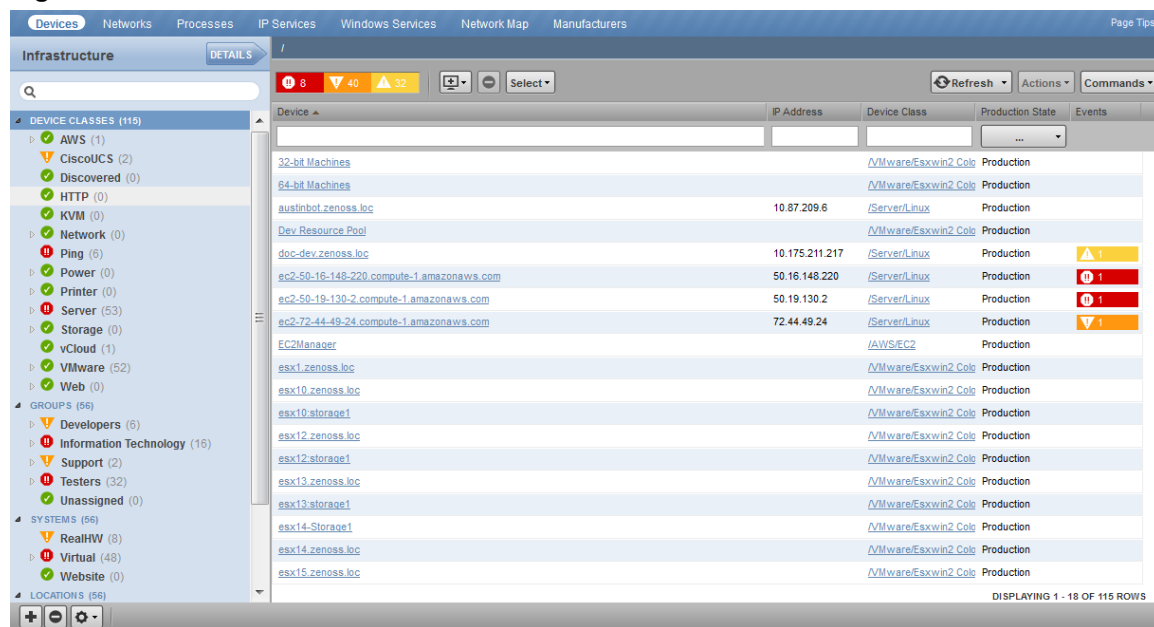
This chapter provides information and procedures for managing devices in the system.

## Viewing the Device List

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

To access the device list, select Infrastructure from the Navigation menu.

**Figure 32: Device List**



## Devices Hierarchy

Devices are organized in the tree view by:

- Device class
- Group
- System

- Location

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

## Managing Multiple Devices from the Device List

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

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

## Working with Devices

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

**Figure 33:** Device Overview

The screenshot shows the Zenoss Device Overview page for the device **ucs1.zenoss.loc** (Cisco UCS, 10.87.208.163). The top navigation bar includes tabs for Devices, Networks, Processes, IP Services, Windows Services, Network Map, and Manufacturers. The "event rainbow" at the top shows 0 errors, 14 warnings, 10 alerts, and 29 info messages. The device status is "Up", production state is "Production", and priority is "Normal".

The left sidebar contains a tree view of components and monitoring templates. The main content area is divided into several sections:

- Overview:** Includes a list of components such as Adaptor Units (4), Chassis (2), Ethernet Ports (80), Fabric Interconnects (2), Fan Modules (16), Fibre Channel Ports (16), Host Ethernet Interfaces (8), Management Interfaces (29), Memory Arrays (4), Power Supply Units (12), Processor Units (8), Server Blades (4), Service Profiles (10), Switch Cards (4), and Virtual Ethernet NICs (20).
- Device Information:** Displays fields like Device ID (10.87.208.163), Connection Information (admin 443 True), Uptime (21d:02h:59m:36s), First Seen (2015-01-07 10:39:37 am), Last Change (2015-01-07 10:40:27 am), Model Time (2015-01-07 10:40:27 am), Locking (Unlocked), and Memory/Swap (Unknown/Unknown). It also includes a "Connect to this device" link.
- Configuration Fields:** Includes input fields for Device Name (ucs1.zenoss.loc), Production State (Production), Priority (Normal), Tag, Serial Number, and Rack Slot (0). There are "Save" and "Cancel" buttons.
- Collector Information:** Lists Collector (localhost), Hardware Manufacturer (None), Hardware Model (None), OS Manufacturer (None), and OS Model (None).
- Systems and Groups:** Shows Systems (None), Groups (None), Location (None), and Links.
- Comments:** A text area for adding comments.
- SNMP Information:** Displays SNMP SysName (ucs1-faba), SNMP Location, SNMP Contact, SNMP Description, SNMP Community (public), and SNMP Version (v2c).

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

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

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

- Device title
- Production state
- Priority
- Tag
- Serial Number
- Rack Slot
- Collector
- Hardware and software manufacturer and model
- Systems
- Groups
- Location

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

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

- Events
- Components
- Graphs (Performance)
- Component Graphs
- Modeler Plugins
- Software
- Custom Properties
- Configuration Properties
- Device Administration
- Monitoring Templates

Information that appears here varies depending on device type.

## Events

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

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

## Components

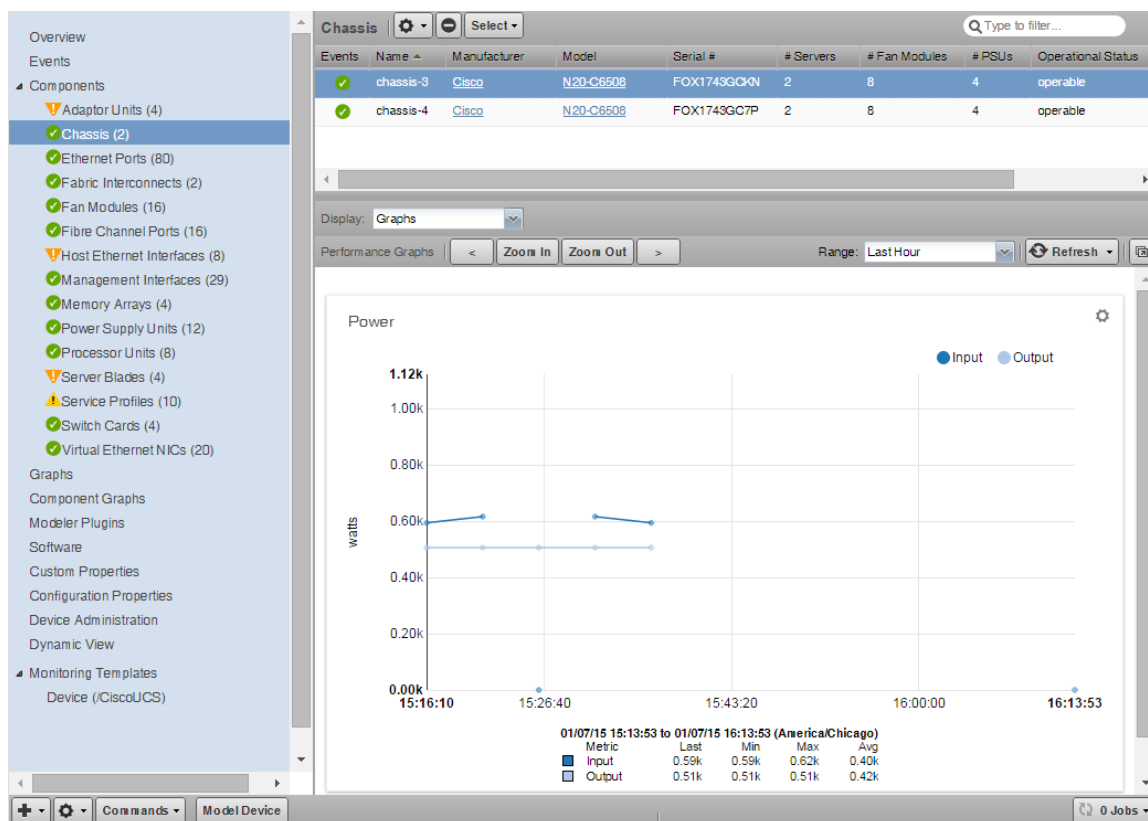
The Components view provides information about the different types of device components, including:

- IPService

- WinService
- IpRouteEntry
- IpInterface
- CPU
- FileSystem

To access components information, select Components in the left panel, and then select a component type. The components available will vary based on the type of device.

**Figure 34: Device (Components)**



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

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

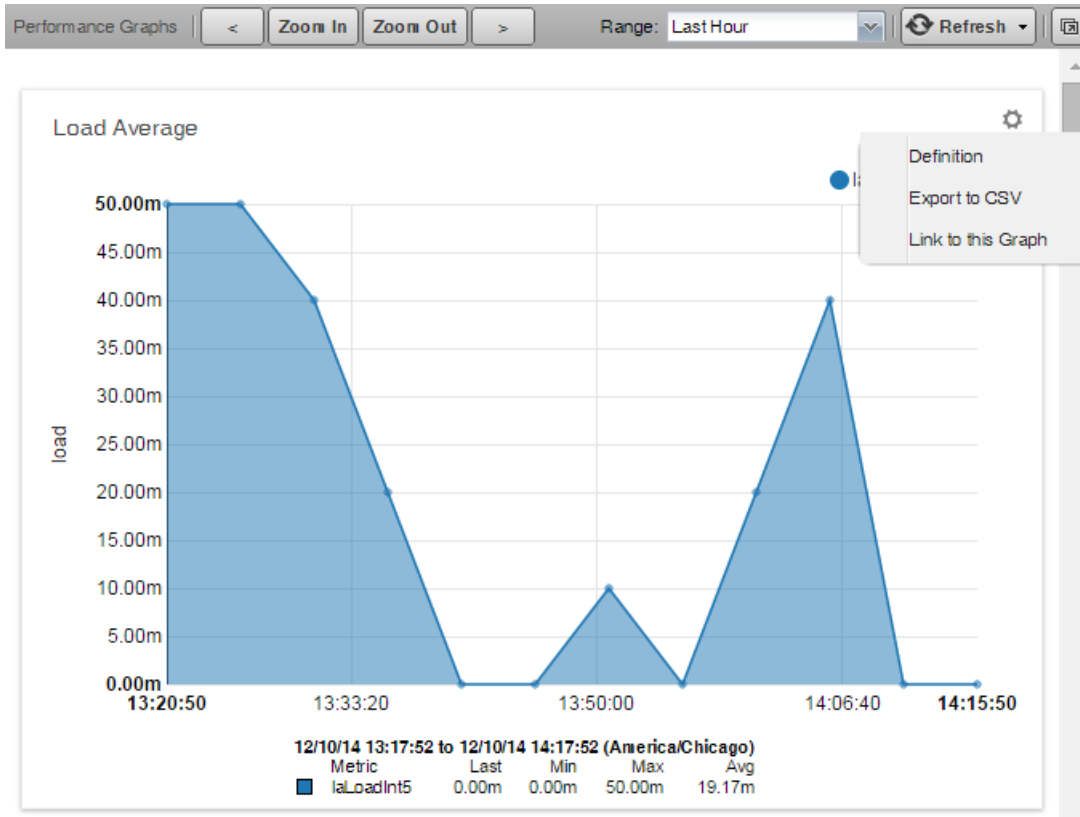
From this view, you can:

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

## Graphs (Performance)

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

**Figure 35: Performance Graph (Device)**



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

You can control these performance graph options:

- **Range** - Select the span of time displayed in the graph. You can select:
  - Last Hour
  - Yesterday
  - Last Week
  - Last 30 days
  - Last Year
  - Custom
- **Refresh** - Modify the refresh value (by default, 30 minutes) by clicking the drop-down list. Setting the refresh rate to manual requires you to click the Refresh button each time you want an updated graph.
- **Full Screen** - Click the Full Screen icon to display only the graphs in your browser.

Each graph can be further controlled by clicking the Action icon located in the upper-right corner of the graph. You can perform the following actions:

- **Definition** - View the JSON definition.
- **Export to CSV** - Export the datapoints as a .csv file for use in a spreadsheet.

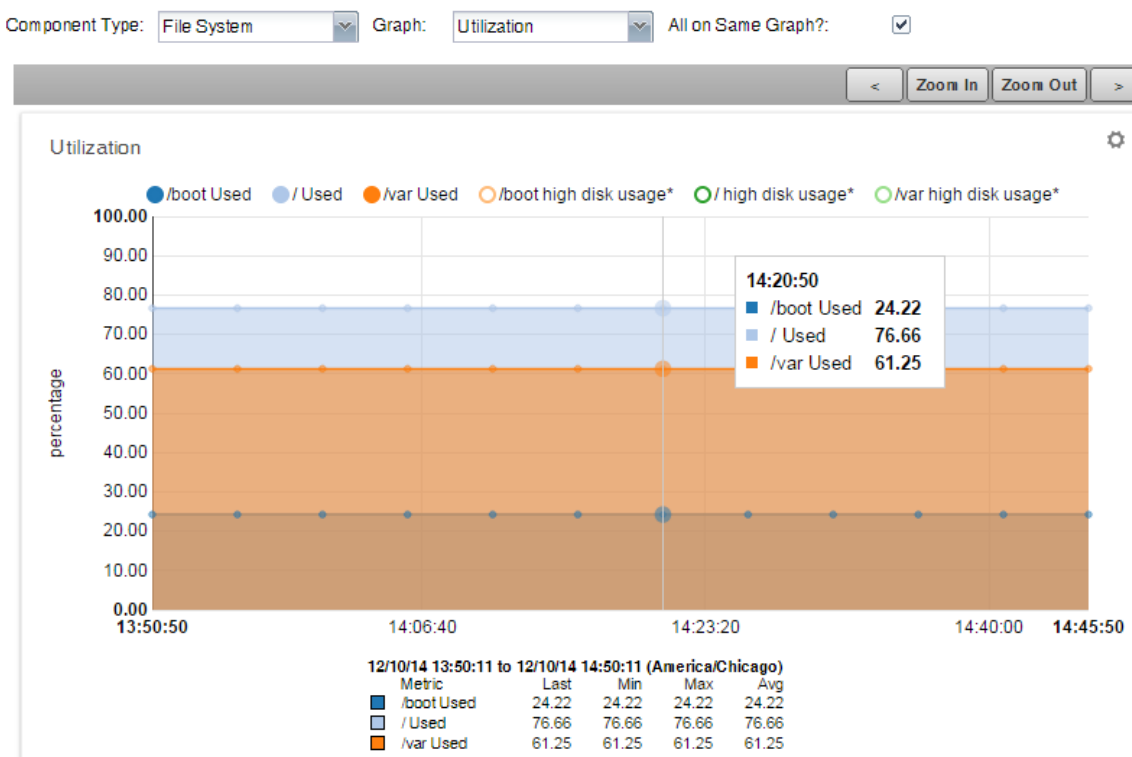
- **Link to this Graph** - Generate a link to this graph that can be saved to your browser bookmarks or use the URL to directly point to the graph in another Web page or dashboard. For example, you may want to show this graph in the Zenoss Dashboard. You can create a new Site Window portlet and insert the URL to this graph.

For more information about performance monitoring and performance graphs, see [Performance Monitoring](#) on page 87.

## Component Graphs

The Component Graphs view shows component graphs defined for the device. To access these graphs, select **Component Graphs** in the left panel. The following figure shows a file system utilization graph that has three metrics all displayed on the same graph.

**Figure 36: Component Graph (Device)**



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

You can control these component graph options:

- **Component Type** - Drop-down of available components based on the type of device being monitored.
- **Graph** - Drop-down of available graphs based on the selected component type.
- **All on Same Graph?** - Select this check box to display all the metrics on one graph. Clear this check box to have a separate graph for each metric.

Each graph can be further controlled by clicking the Action icon located in the upper-right corner of the graph. You can perform the following actions:

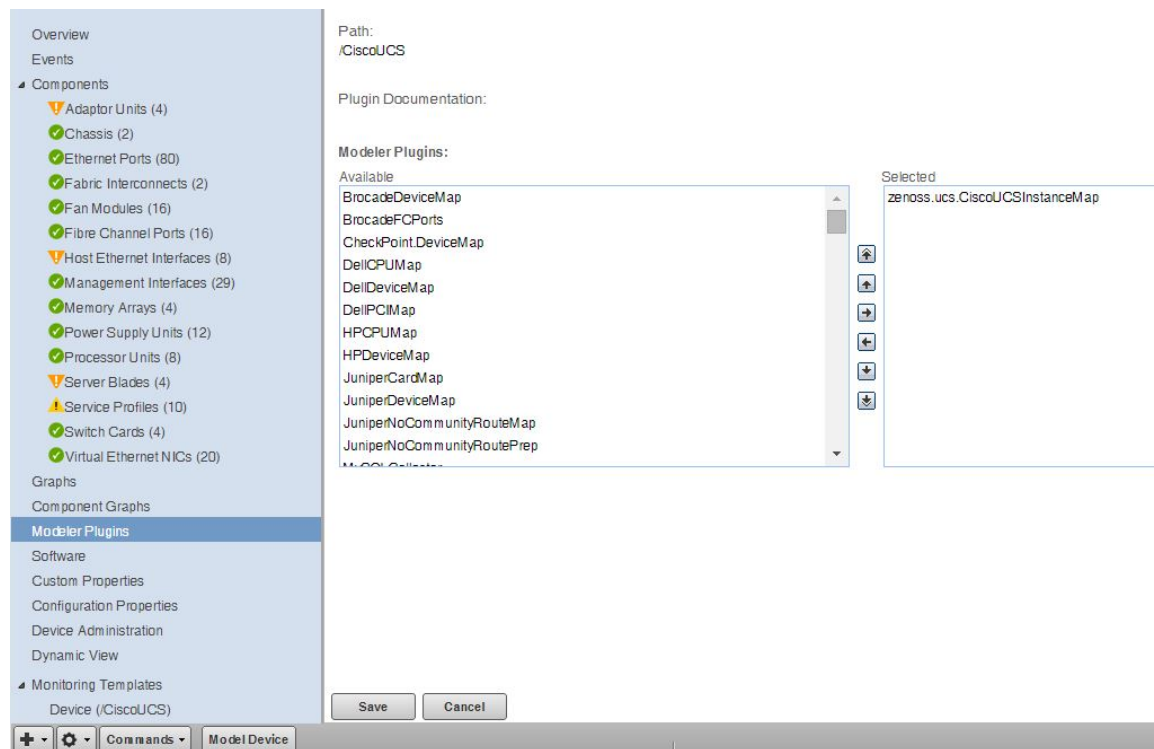
- **Definition** - View the JSON definition.
- **Export to CSV** - Export the datapoints as a .csv file for use in a spreadsheet.

- **Link to this Graph** - Generate a link to this graph that can be saved to your browser bookmarks or use the URL to directly point to the graph in another Web page or dashboard. For example, you may want to show this graph in the Zenoss Dashboard. You can create a new Site Window portlet and insert the URL to this graph.

## Modeler Plugins

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

**Figure 37: Device (Modeler Plugins)**



## Configuration Properties

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

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

**Figure 38: Device (Configuration Properties)**

| Is Local | Category         | Name                        | Value              | Path |
|----------|------------------|-----------------------------|--------------------|------|
|          | Cisco            | zCiscoACEUseSSL             | true               | /    |
|          | Cisco            | zCiscoRemodelEventClassKeys |                    | /    |
|          | Cisco UCS        | zCiscoUCSCIMCEventsInterval | 60                 | /    |
|          | Cisco UCS        | zCiscoUCSCIMCOPerfInterval  | 300                | /    |
|          | Cisco UCS        | zCiscoUCSManagerPort        | 443                | /    |
|          | Cisco UCS        | zCiscoUCSManagerUseSSL      | true               | /    |
|          | Cisco UCS        | zCiscoUCSManagerUser        | admin              | /    |
|          | Modeler Controls | zCollectorClientTimeout     | 180                | /    |
|          | Modeler Controls | zCollectorDecoding          | utf-8              | /    |
|          | Misc             | zCollectorLogChanges        | false              | /    |
|          | zencommand       | zCommandCommandTimeout      | 15                 | /    |
|          | zencommand       | zCommandExistenceTest       | test -f %s         | /    |
|          | zencommand       | zCommandLoginTimeout        | 10                 | /    |
|          | zencommand       | zCommandLoginTries          | 1                  | /    |
|          | zencommand       | zCommandPassword            |                    | /    |
|          | zencommand       | zCommandPath                | \$ZENHOME/libexec  | /    |
|          | zencommand       | zCommandPort                | 22                 | /    |
|          | zencommand       | zCommandProtocol            | ssh                | /    |
|          | zencommand       | zCommandSearchPath          |                    | /    |
|          | zencommand       | zCommandUsername            |                    | /    |
|          | Control Center   | zControlCenterHost          | \$(here/managerip) | /    |
|          | Control Center   | zControlCenterModelCycle    | 3600               | /    |
|          | Control Center   | zControlCenterPassword      |                    | /    |
|          | Control Center   | zControlCenterPerfData      | on                 | /    |

For detailed information about working with configuration properties, see [Configuration Properties](#) on page 55

## Software

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

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

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

## Device Administration

Use the Device Administration view to:

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

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



**Figure 39: Device Administration**

The screenshot shows the 'Device Administration' section of a software interface. It includes three main panels:

- Maintenance Windows:** A table with columns: Enabled, Name, Start, Duration, Repeat, and State.
 

| Enabled | Name                 | Start               | Duration     | Repeat                | State       |
|---------|----------------------|---------------------|--------------|-----------------------|-------------|
| No      | 1st of Month         | 2014/12/01 02:00:00 | 01:00:00 hrs | Monthly: day of month | Maintenance |
| Yes     | Every Thursday Night | 2014/12/11 22:00:00 | 30:00 mins   | Weekly                | Maintenance |
| Yes     | One Time Testing     | 2014/12/20 08:00:00 | 04:00:00 hrs | Never                 | Test        |
- User Commands:** A table with columns: Name, Command.
 

| Name        | Command  |
|-------------|--|
| DNS forward | host \${device/id}                                       |
| DNS reverse | host \${device/managerip}                                |
| ping        | \${device/pingCommand} -c2 \${device/managerip}          |
| snmpwalk    | snmpwalk -s \${device/zsnmpVer} -c \${device/zsnmpVer}   |
| tracert     | \${device/tracertCommand} -q 1 -w 2 \${device/managerip} |
- Administrators:** A table with columns: Name, Role, Email, Pager.
 

| Name       | Role    | Email   | Pager |
|------------|---------|---------|-------|
| cgilchrist | Manager | cgil... |       |

See these topics for more information about device administration tasks:

- [Maintenance Windows](#) on page 126
- [Defining User Commands for a Single Device](#) on page 139
- [Defining User Commands for All Devices in an Organizer](#) on page 140
- [Adding Administrators](#) on page 145

## Custom Properties

Use the Custom Properties view to edit the values of custom properties associated with a device.

To display the Custom Properties view, click on a device name in the device list, and then select **Custom Properties** in the left panel. You can perform the following actions on custom properties:

- Create
- Edit
- Refresh
- Delete

**Figure 40: Device (Custom Properties)**

The screenshot shows the 'Device (Custom Properties)' view. It features a table with the following columns: Property Name, Description, Value, Path, Type, and Is Local. The table contains one row:

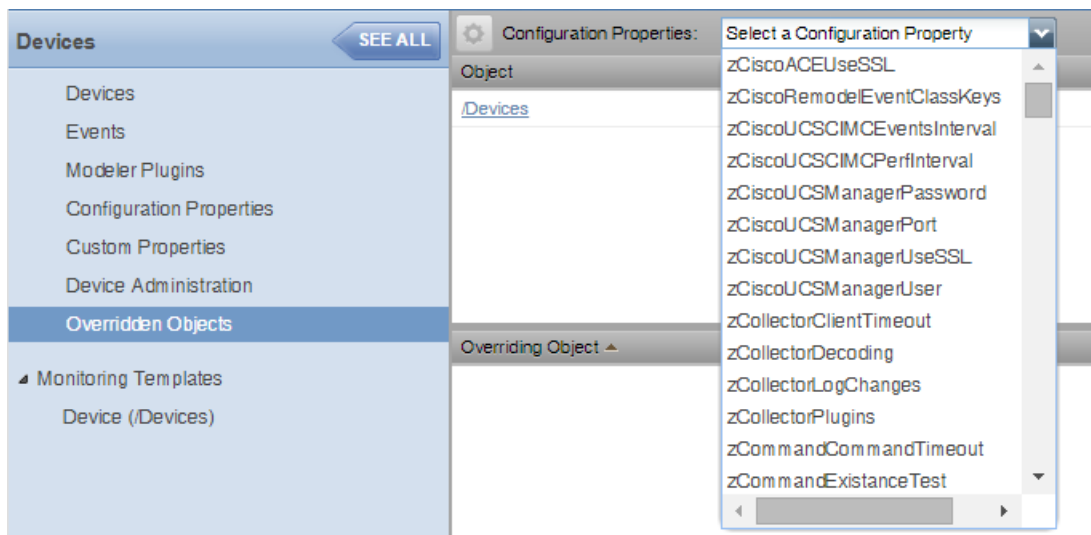
| Property Name | Description | Value                          | Path | Type | Is Local |
|---------------|-------------|--------------------------------|------|------|----------|
| cDateTest     |             | 1900/01/01 00:00:00 US/Central | /    | date |          |

**Note** The Custom Properties view allows you to edit the value of a custom property on an individual device, but not to define new custom properties for device classes. For more information about custom properties, refer to Zenoss Knowledge Base article 16052-109.

## Overridden Objects

Use the Overridden Objects view to see the objects that have overrides on their configuration properties. This view is available when looking at details of all devices.

To display the Overridden Objects view, navigate to the **Infrastructure > Devices** pages and click **Details**. Then, click **Overridden Objects** from the left-column menu.



Select a configuration property from the drop-down list to view the overridden objects for that property. Double-click the row of the overriding object to open an edit dialog box.

**Note** Do not click the link of the overriding object. You will be taken to that object's page in the infrastructure view. Instead, double-click the clear area of the row of the overriding object to view the Edit Configuration Property dialog.

## Monitoring Templates

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

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

**Figure 41: Device (Monitoring Templates)**

| Data Sources  |                           |         |      | Thresholds  |  |
|---|---------------------------|---------|------|---|--|
| <div> <div>+</div> <div>-</div> <div>⚙</div> </div> |                           |         |      | <div> <div>+</div> <div>-</div> <div>⚙</div> </div> |  |
| Data Points by Data Source ▴                        | Source                    | Enabled | Type | Name  |  |
| ▸ laLoadInt1  | 1.3.6.1.4.1.2021.10.1.5.1 | true    | SNMP | high load   |  |
| ▸ laLoadInt15                                       | 1.3.6.1.4.1.2021.10.1.5.3 | true    | SNMP | low CPU idle  |  |
| ▸ laLoadInt5  | 1.3.6.1.4.1.2021.10.1.5.2 | true    | SNMP |   |  |
| ▸ memAvailReal                                      | 1.3.6.1.4.1.2021.4.6.0    | true    | SNMP |   |  |
| ▸ memAvailSwap                                      | 1.3.6.1.4.1.2021.4.4.0    | true    | SNMP |   |  |
| ▸ memBuffer   | 1.3.6.1.4.1.2021.4.14.0   | true    | SNMP |   |  |
| ▸ memCached   | 1.3.6.1.4.1.2021.4.15.0   | true    | SNMP |   |  |
| ▸ ssCpuIdle   | 1.3.6.1.4.1.2021.11.11.0  | true    | SNMP |   |  |
| ▸ ssCpuRawWait                                      | 1.3.6.1.4.1.2021.11.54.0  | true    | SNMP |   |  |
| ▸ ssCpuSystem                                       | 1.3.6.1.4.1.2021.11.10.0  | true    | SNMP |   |  |
| ▸ ssCpuUser   | 1.3.6.1.4.1.2021.11.9.0   | true    | SNMP |   |  |
| ▸ sslORawReceived                                   | 1.3.6.1.4.1.2021.11.58.0  | true    | SNMP |   |  |
| ▸ sslORawSent                                       | 1.3.6.1.4.1.2021.11.57.0  | true    | SNMP |   |  |
| ▸ sysUpTime   | 1.3.6.1.2.1.25.1.1.0      | true    | SNMP |   |  |
|   |                           |         |      | <div> <div>+</div> <div>-</div> <div>⚙</div> </div> |  |
|   |                           |         |      | Name  |  |
|   |                           |         |      | Load Average  |  |
|   |                           |         |      | CPU Utilization                                     |  |
|   |                           |         |      | Memory Utilization                                  |  |
|   |                           |         |      | IO  |  |

For detailed information about monitoring templates, go to the section titled "[Performance Monitoring](#)."

## Managing Devices and Device Attributes

---

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

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

### Clearing Heartbeat Events

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

To clear the heartbeat events associated with a device:

- 1 Navigate to Advanced > Settings.
- 2 In the left panel, select Events.
- 3 At the bottom of the Event Configuration page, click the **Clear** button in the Clear Event Heartbeats section.

The system displays a brief message banner.

### Pushing Configuration Changes to Zenoss Core

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

To push configuration changes:

- 1 Navigate to the device in the device list.
- 2 At the bottom of the device overview page, select Push Changes from the Action menu.

The Push Changes dialog appears.

- 3 Click **OK**.

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

### Locking Device Configuration

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

---

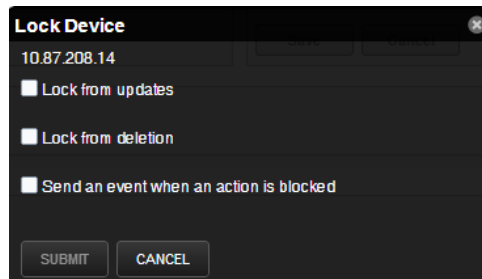
**Note** Device locking prevents changes and deletion due to remodeling. It does not prevent manual changes and deletion.

---

To edit lock selections for a device configuration:

- 1 Navigate to the device in the device list.
- 2 At the bottom of the device overview page, select Locking from the Action menu.

The Lock Device dialog appears.

**Figure 42: Lock Device Dialog**

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

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

## Renaming a Device

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

To rename a device:

- 1 Navigate to the device in the device list. Click the device name.
- 2 On the device overview page, edit the Device Name field with the new device name.
- 3 Click **Save**.

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

## Reidentifying a Device

You may want to change the Device ID in the system. This is different than changing the Device Name. If you change the ID, you will lose all graph data for this device.

To reidentify a device:

- 1 Navigate to the device in the device list. Click the device name to open the Device Overview page.
- 2 At the bottom of the Device Overview page, select **Reidentify Device** from the Action menu.
- 3 Enter a new ID for the device.
- 4 Click **Submit**.

## Remodeling a Device

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

- 1 Navigate to the device in the device list and click on the Device name.
- 2 At the bottom of the Device Overview page, click the **Model Device** button.

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

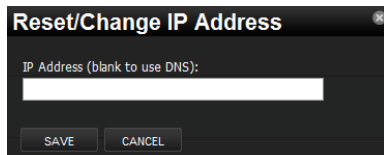
## Resetting the Device Manage IP Address

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

- 1 Navigate to the device in the device list.
- 2 At the bottom of the device overview page, select Reset/Change IP Address from the Action menu.

The Reset IP dialog appears.

**Figure 43: Reset IP Dialog**



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

The IP address for the device is reset.

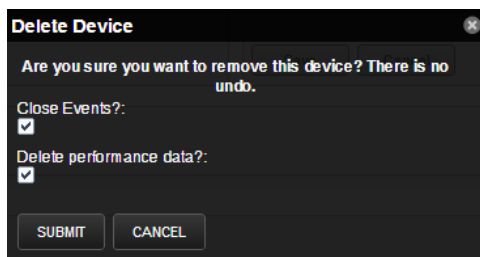
## Deleting a Device

To delete a device from the system:

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

The Delete Device dialog appears.

**Figure 44: Delete Device**



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

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

## Dumping and Loading Devices

The system allows you to export a list of your devices to a text file to import into another system instance. You can do this by using the `zenbatchdump` command.

You cannot use `zenbatchdump` on devices whose classes were defined in a ZenPack.

From the command line, use the command:

```
zenbatchdump > mydevicelist.txt
```

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

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

```
zenbatchload mydevicelist.txt
```

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

For additional ways to add and discover devices, see the "Add Devices" section of the *Zenoss Core Installation*.

## 4

# Configuration Properties

---

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

- **Devices**

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

- **Events**

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

- **Networks**

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

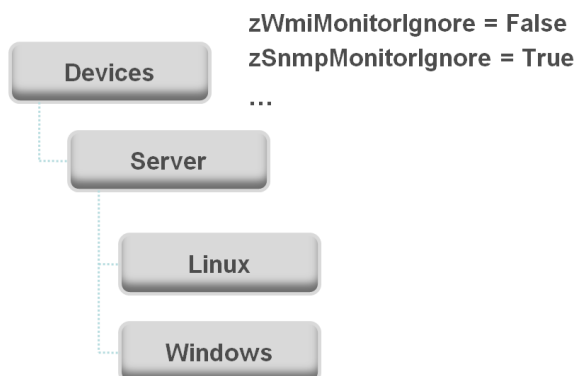
Configuration properties and values can be added to the ZenPacks you create, allowing you to customize the system when you add ZenPacks.

## Configuration Properties Inheritance and Override

---

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

**Figure 45: Device Class Hierarchy**



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

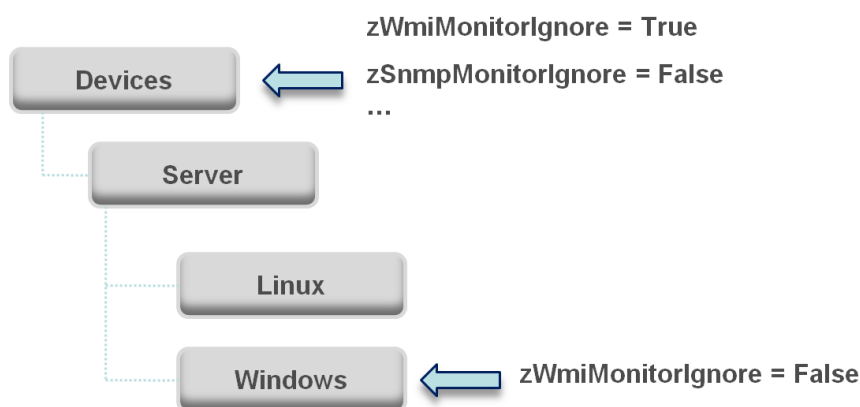
The illustration shows two defined configuration properties:

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

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

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

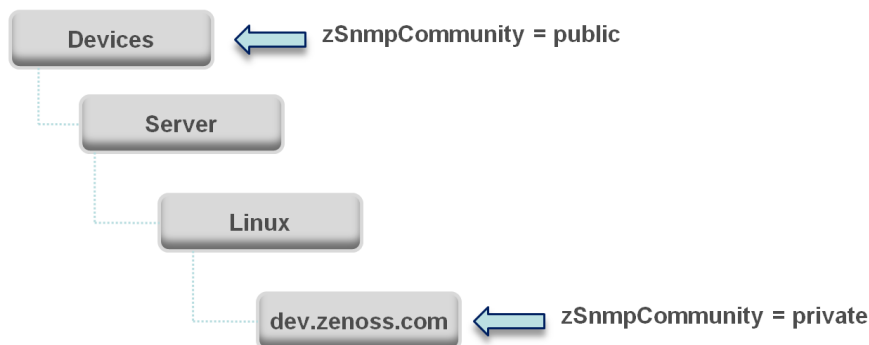
**Figure 46: Device Class Hierarchy - Locally Defined Value (Override)**



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

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

**Figure 47: Device Class Hierarchy - Value Set on Device**



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



## Viewing Properties from the User Interface

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

- 1 Select Infrastructure from the Navigation menu.

The Devices page appears.

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



**Figure 48:** Defined Device Configuration Properties - Root Level

| Configuration Properties <span>⚙️</span> <span>🔄</span> <a href="#">Delete Local Copy</a> |                  |                           |  |      |
|---|------------------|---------------------------|--|------|
| Is Local  | Category         | Name ▲                    | Value  | Path |
|   |                  |                           |  |      |
| Yes   | CiscoUCS         | zCiscoUCSManagerPassword  |  | /    |
| Yes   | CiscoUCS         | zCiscoUCSManagerPort      | 80   | /    |
| Yes   | CiscoUCS         | zCiscoUCSManagerUseSSL    | false  | /    |
| Yes   | CiscoUCS         | zCiscoUCSManagerUser      |  | /    |
| Yes   | Modeler Controls | zCollectorClientTimeout   | 180  | /    |
| Yes   | Modeler Controls | zCollectorDecoding        | latin-1  | /    |
| Yes   | Misc             | zCollectorLogChanges      | false  | /    |
| Yes   | zencommand       | zCommandCommandTimeout    | 15   | /    |
| Yes   | zencommand       | zCommandCycleTime         | 60   | /    |
| Yes   | zencommand       | zCommandExistenceTest     | test -f %s   | /    |
| Yes   | zencommand       | zCommandLoginTimeout      | 10   | /    |
| Yes   | zencommand       | zCommandLoginTries        | 1  | /    |
| Yes   | zencommand       | zCommandPassword          |  | /    |
| Yes   | zencommand       | zCommandPath              | \$ZENHOME/libexec  | /    |
| Yes   | zencommand       | zCommandPort              | 22   | /    |
| Yes   | zencommand       | zCommandProtocol          | ssh  | /    |
| Yes   | zencommand       | zCommandSearchPath        |  | /    |
| Yes   | zencommand       | zCommandUsername          |  | /    |
| Yes   | Misc             | zDeviceTemplates          | Device   | /    |
| Yes   | Misc             | zEC2Secret                |  | /    |
| Yes   | Modeler Controls | zFileSystemMapIgnoreNames |  | /    |
| Yes   | Modeler Controls | zFileSystemMapIgnoreTypes | other ram virtualMemory removableDisk floppyDisk compactDisk ramDisk f | /    |
| Yes   | Misc             | zFileSystemSizeOffset     | 1  | /    |

DISPLAYING 1 - 22 OF 127 ROWS

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

**Figure 49:** zCollectorClientTimeout Configuration Property - Local Value Set

| Configuration Properties   <a href="#">Delete Local Copy</a> |                  |                          |            |  |
|--|------------------|--------------------------|------------|--|
| Is Local   | Category         | Name ▲                   | Value      | Path                                   |
|  |                  |                          |            |  |
|  | CiscoUCS         | zCiscoUCSManagerPassword |            | /                                      |
|  | CiscoUCS         | zCiscoUCSManagerPort     | 80         | /                                      |
|  | CiscoUCS         | zCiscoUCSManagerUseSSL   | false      | /                                      |
|  | CiscoUCS         | zCiscoUCSManagerUser     |            | /                                      |
| Yes  | Modeler Controls | zCollectorClientTimeout  |            | /Server/Linux/devices/t-cent5a-64.zeno |
|  | Modeler Controls | zCollectorDecoding       |            | /                                      |
|  | Misc             | zCollectorLogChanges     |            | /                                      |
|  | zencommand       | zCommandCommandTimeout   | 15         | /                                      |
|  | zencommand       | zCommandCycleTime        | 60         | /                                      |
|  | zencommand       | zCommandExistanceTest    | test -f %s | /                                      |
|  | zencommand       | zCommandLoginTimeout     | 10         | /                                      |
|  | zencommand       | zCommandLoginTries       | 1          | /                                      |

Shows the node at which the default value is overridden

DISPLAYING 1 - 13 OF 127 ROWS

To remove the override and once again inherit the value from the root of the hierarchy:

- 1 Select the property in the list.
- 2 Click **Delete Local Copy**.

The Delete Local Property dialog appears, and requests confirmation of the action.

- 3 Click **OK**.

## Configuration Property Types

Configuration properties can be one of these types:

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

## Device Configuration Properties

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

- 1 Select Infrastructure from the Navigation menu.
- 2 In the tree view, click **Details**.
- 3 Select **Configuration Properties**.

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

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

The following table lists device configuration properties.

**Table 1: Device Configuration Properties**

| Property Name               | Property Type | Description   |
|-----------------------------|---------------|---|
| zCiscoACEUseSSL             | boolean       | Whether to use SSL when connecting to ACE XML API.  |
| zCiscoRemodelEventClassKeys | lines         | Allows you to modify the list of SNMP traps that will cause Zenoss to schedule an immediate remodeling of the device from which the trap was sent.    |
| zCiscoUCSCIMCEventsInterval | int           | Event collection interval in seconds. Default is 60.  |
| zCiscoUCSCIMCPerfInterval   | int           | Metric collection interval in seconds. Default is 300.  |
| zCiscoUCSManagerPassword    | string        |   |
| zCiscoUCSManagerPort        | int           | Port used to connect to the UCS Manager or CIMC XML APIs. Default is 443 and typically should not be changed.   |
| zCiscoUCSManagerUseSSL      | boolean       | Whether to use SSL when connecting to the UCS Manager or CIMC XML APIs. Default is true and typically should not be changed.                          |
| zCiscoUCSManagerUser        | string        |   |
| zCollectorClientTimeout     | int           | Allows you to set the timeout time of the collector client in seconds   |
| zCollectorDecoding          | string        | Converts incoming characters to Unicode.  |
| zCollectorLogChanges        | boolean       | Indicates whether to log changes.   |
| zCommandCommandTimeout      | float         | Specifies the time to wait for a command to complete.   |
| zCommandExistanceTest       | string        |   |
| zCommandLoginTimeout        | float         | Specifies the time to wait for a login prompt.  |
| zCommandLoginTries          | int           | Sets the number of times to attempt login.  |
| zCommandPassword            | string        | Specifies the password to use when performing command logins and SSH.   |
| zCommandPath                | string        | Sets the default path where ZenCommand plug-ins are installed on the local Zenoss Core box (or on a remote box where SSH is used to run the command). |
| zCommandPort                | int           | Specifies the port to connect to when performing command collection.  |
| zCommandProtocol            | string        | Establishes the protocol to use when performing command collection. Possible values are SSH and telnet.   |
| zCommandSearchPath          | lines         | Sets the path to search for any commands.   |
| zCommandUsername            | string        | Specifies the user name to use when performing command collection and SSH.  |
| zControlCenterHost          |               |   |
| ZControlCenterModelCycle    |               |   |

| Property Name                   | Property Type | Description   |
|---------------------------------|---------------|---|
| zControlCenterPassword          |               |   |
| zControlCenterPerfCycle         |               |   |
| zControlCenterPort              |               |   |
| zControlCenterUser              |               |   |
| zDBInstances                    |               | This setting is only relevant when the <code>zenoss.winrm.WinMSSQL</code> modeler plugin is enabled. Multiple instances can be specified to monitor multiple SQL Server instances per server. The default instance is <code>MSSQLSERVER</code> . Fill in the user and password to use SQL authentication. Leave the user and password blank to use Windows authentication.  |
| zDeviceTemplates                | lines         | Sets the templates associated with this device. Linked by name.   |
| zEnablePassword                 |               |   |
| zFileSystemMapIgnoreNames       | string        | Sets a regular expression of file system names to ignore.   |
| zFileSystemMapIgnoreTypes       | lines         | Do not use.   |
| zFileSystemSizeOffset           | int           | SNMP typically reports the total space available to privileged users. Zenoss Core (like the <code>df</code> command) reports capacity based on the space available to non-privileged users. The value of <code>zFileSystemSizeOffset</code> should be the fraction of the total space that is available to non-privileged users. The default reserved value is 5% of total space, so <code>zFileSystemSizeOffset</code> is preset to <code>.95</code> . If the reserved portion is different than 5%, then adjust the value of <code>zFileSystemSizeOffset</code> accordingly. The fraction should be set according to the value $(Used + Avail) / Size$ when the <code>df -Pk</code> command is run at the command line. |
| zHardDiskMapMatch               | string        | Regular expression that uses the disk ID in the <code>diskstats</code> output to filter disk activity statistics for inclusion in performance monitoring.   |
| zIcon                           | lines         | Specifies the icon to represent the device wherever device icon is shown, such as on the network map and device status page.  |
| zIdiomPassword                  | string        |   |
| zIdiomUsername                  | string        |   |
| zIfDescription                  | Boolean       | Shows the interface description field in the interface list.  |
| zInterfaceMapIgnoreDescriptions | string        | Filters out interfaces based on description.  |
| zInterfaceMapIgnoreNames        | string        | Filters out interfaces that should not be discovered.   |

| Property Name                   | Property Type | Description   |
|---------------------------------|---------------|---|
| zInterfaceMapIgnoreTypes        | string        | Filters out interface maps that should not be discovered.   |
| zIpServiceMapMaxPort            | int           | Specifies the highest port to scan. The default is 1024.  |
| zJBossJmxManagementAuthenticate |               |   |
| zJBossJmxManagementPassword     |               |   |
| zJBossJmxManagementPort         |               |   |
| zJBossJmxManagementUsername     |               |   |
| zJmxAuthenticate                |               |   |
| zJmxManagementPort              |               |   |
| zJmxPassword                    |               |   |
| zJmxUsername                    |               |   |
| zKeyPath                        | string        | Sets the path to the SSH key for device access.   |
| zLDAPBaseDN                     |               |   |
| zLDAPBindDN                     |               |   |
| zLDAPBindPassword               |               |   |
| zLDMsAutodiscover               |               |   |
| zLTMVirtualServerIgnoreNames    |               |   |
| zLinks                          | string        | Specifies a place to enter any links associated with the device.  |
| zLocalInterfaceNames            | string        | Regular expression that uses interface name to determine whether the IP addresses on an interface should be incorporated into the network map. For instance, a loopback interface "lo" might be excluded.             |
| zLocalIpAddresses               | string        | Specifies IP addresses that should be excluded from the network map (for example, 127.x addresses). If you have addresses that you reuse for connections between clustered machines they might be added here as well. |
| zMaxOIDPerRequest               | int           | Sets the maximum number of OIDs to be sent by the SNMP collection daemons when querying information. Some devices have small buffers for handling this information so the number should be lowered.                   |
| zMySQLConnectionString          |               |   |
| zMySQLPassword                  |               |   |
| zMySQLPort                      |               |   |
| zMySQLUsername                  |               |   |
| zNetAppSSL                      |               |   |

| Property Name                | Property Type | Description   |
|------------------------------|---------------|---|
| zNmapPortscanOptions         |               |   |
| zPingMonitorIgnore           | Boolean       | Whether or not to ping the device.  |
| zProdStateThreshold          | int           | Production state threshold at which Zenoss Core will begin to monitor a device.   |
| zPropertyMonitorInterval     |               |   |
| zPythonClass                 | string        | DO NOT USE  |
| zRancidGroup                 |               | RANCID group attribute. Controls what <code>router.db</code> file the device is written to. Can be set at the device class or device level. Default is <code>router</code> on the <code>/Network/Router/Cisco</code> class.   |
| zRancidRoot                  |               | File system directory where RANCID is installed. It may be NFS mounted from the RANCID server. Default is <code>/opt/rancid</code>  |
| zRancidType                  |               | RANCID type attribute. Controls what device type is written to the <code>router.db</code> file. Can be set at the device class or device level. Default is <code>cisco</code> on the <code>/Network/Router/Cisco</code> class.  |
| zRancidUrl                   |               | Base URL to viewvc  |
| zRouteMapCollectOnlyIndirect | Boolean       | Only collect routes that are indirectly connected to the device.  |
| zRouteMapCollectOnlyLocal    | Boolean       | Only collect local routes. (These usually are manually configured rather than learned through a routing protocol.)  |
| zRouteMapMaxRoutes           | int           | Sets maximum number of routes to collect. Default value is 500.   |
| zSnmpAuthPassword            | string        | The shared private key used for authentication. Must be at least 8 characters long.   |
| zSnmpAuthType                | string        | Use "MD5" or "SHA" signatures to authenticate SNMP requests   |
| zSnmpCollectionInterval      | int           | Defines, in seconds, how often the system collects performance information for each device.   |
| zSnmpCommunities             | lines         | Array of SNMP community strings that ZenModeler uses when collecting SNMP information. When you set this property, communities are tried in order; the first in the list that is successful is used as <code>zSnmpCommunity</code> . If none is successful, then the current value of <code>zSnmpCommunity</code> is used. The default value for the entire system is "public." |
| zSnmpCommunity               | string        | Community string to be used when collecting SNMP information. If it is different than what is found by ZenModeler, it will be set on the modeled device.  |
| zSnmpContext                 |               |   |

| Property Name                    | Property Type | Description   |
|----------------------------------|---------------|---|
| zSnmpDiscoveryPorts              |               |   |
| zSnmpEngineId                    |               |   |
| zSnmpMonitorIgnore               | Boolean       | Whether or not to ignore monitoring SNMP on a device.   |
| zSnmpPort                        | int           | Port that the SNMP agent listens on.  |
| zSnmpPrivPassword                | string        | The shared private key used for encrypting SNMP requests. Must be at least 8 characters long.         |
| zSnmpPrivType                    | string        | "DES" or "AES" cryptographic algorithms.  |
| zSnmpSecurityName                | string        | The Security Name (user) to use when making SNMPv3 requests.  |
| zSnmpTimeout                     | float         | Timeout time in seconds for an SNMP request   |
| zSnmpTries                       | int           | Amount of tries to collect SNMP data  |
| zSnmpVer                         | string        | SNMP version used. Valid values are v2c, v3   |
| zSshConcurrentSessions           |               |   |
| zStatusConnectTimeout            | float         | The amount of time that the zenstatus daemon should wait before marking an IP service down.           |
| zSugarCRMBase                    |               |   |
| zSugarCRMPassword                |               | Password for the zSugarCRMUsername user.  |
| zSugarCRMTestAccount             |               |   |
| zSugarCRMUsername                |               | Username allowed to log in to the Sugar CRM server.   |
| zSysedgeDiskMapIgnoreNames       |               |   |
| 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.   |
| zTomcatJ2EEApplicationName       |               |   |
| zTomcatJ2EEServerName            |               |   |
| zTomcatJmxManagementAuthenticate |               |   |
| zTomcatJmxManagementPassword     |               |   |
| zTomcatJmxManagementPort         |               |   |

| Property Name                      | Property Type | Description  |
|------------------------------------|---------------|--|
| zTomcatJmxManagementUsername       |               |  |
| zTomcatListenHost                  |               |  |
| zTomcatListenPort                  |               |  |
| zTomcatServletName                 |               |  |
| zTomcatServletUri                  |               |  |
| zTomcatWebAppUri                   |               |  |
| zVCloudPassword                    |               | Password for the zVCloudUsername user.   |
| zVCloudPort                        |               | Value of the cell port number  |
| zVCloudUsername                    |               | Username for the cell in the form username@organization. For example, if you want to define the cell administrator, enter administrator@system.      |
| zVSphereEndpointHost               |               |  |
| zVSphereEndpointPassword           |               |  |
| zVSphereEndpointUseSsl             |               |  |
| zVSphereEndpointUser               |               |  |
| zVSpherePerfDelayCollectionMinutes |               | Value of how long to lag performance data collection. Default value is 0.  |
| zVSpherePerfQueryChunkSize         |               | Value of how many performance requests to make at a time. Default value is 50.   |
| zVSpherePerfWindowSize             |               | Value of how many intervals of data to ask for in each performance request. Default value is 2.  |
| zWebLogicJmxManagementAuthenticate |               |  |
| zWebLogicJmxManagementPassword     |               |  |
| zWebLogicJmxManagementPort         |               |  |
| zWebLogicJmxManagementUsername     |               |  |
| zWebsphereAuthRealm                |               | Used for HTTP basic authentication. This field is not required, and is empty by default.   |
| zWebsphereNode                     |               | Used by the provided template to build the queries for the data to collect. You must supply a value for this field.                                  |
| zWebspherePassword                 |               | Used for HTTP basic authentication. This field is not required, and is empty by default.   |
| zWebsphereServer                   |               | Used by the provided template to build the xpath queries for the data to collect. You must supply a value for this field. There is no default value. |
| zWebsphereURLPath                  |               | Path to the PMI servlet on a WebSphere instance. The default value is the default path on a WebSphere  |



| Property Name       | Property Type | Description   |
|---------------------|---------------|---|
|                     |               | installation: wasPerTool/servlet/perfservlet  |
| zWebsphereUser      |               | Used for HTTP basic authentication. This field is not required, and is empty by default.  |
| zWinKDC             |               | IP address of a valid Windows domain controller. Must be set if domain authentication is used.  |
| zWinKeyTabFilePath  |               | This property is currently used and reserved for future use when keytab files are supported.  |
| zWinPerfmonInterval |               | Interval, in seconds, at which Windows Perfmon datapoints will be collected. Default value is 300. It is possible to override the collection interval for individual counters.  |
| zWinRMPassword      |               | Password for the user defined by zWinRMUser   |
| zWinRMPort          |               | The port on which the Windows server is listening for WinRM or WS-Management connections. Default value is 5985. It is uncommon for this to be configured as anything else.   |
| zWinRMServerName    |               | <p>This property should only be used in conjunction with domain authentication when the DNS PTR record for a monitored server's managed IP address does not resolve to the name by which the server is known in Active Directory. For example, if myserver1 is known as myserver1.ad.example.com by Active Directory and is being managed by IP address 192.51.100.21, but 192.51.100.21 resolves to www.example.com, you will have to set zWinRMServerName to myserver1.ad.example.com for domain authentication to work.</p> <p>If many Windows servers in your environment don't have DNS PTR records that match Active Directory, it is recommended that you set the name of the Zenoss device to be the fully-qualified Active Directory name and set zWinRMServerName to <code>\${here/titleOrId}</code> at the /Server/Microsoft/Windows device class. This avoids the necessity of setting zWinRMServerName on every device.</p> <p>It is recommended to leave zWinRMServerName blank if local authentication is used, or DNS PTR records match Active Directory. This allows Zenoss to not rely on DNS resolution while monitoring, and avoids the overhead of configuring zWinRMServerName.</p> |
| zWinRMUser          |               | The syntax used for zWinRMUser controls whether Zenoss will attempt Windows local authentication or domain (kerberos) authentication. If the value of zWinRMUser is <i>username</i> , local Windows   |

| Property Name | Property Type | Description   |
|---------------|---------------|---|
|               |               | authentication will be used. If zWinRMUser is username@example.com, domain authentication will be used. The zWinKDC and potentially the zWinRMServerName properties become important. |
| zWinScheme    |               | This must be set to either http or https. Default value is http.  |

## Event Configuration Properties

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

- 1 Select **Events** from the Navigation menu, and then select **Event Classes**.
- 2 In the drop-down list, select **Configuration Properties**.

To view and override event configuration properties for a specific event class:

- 1 Navigate to that class, and then select **Configuration Properties**.
- 2 Double-click the configuration property you want to change. The **Edit Config Property** dialog appears.
- 3 Make your changes and click **Submit**. You will see an indication of **Yes** in the **Is Local** column indicating that there is an overriding value. If you want to later remove the override, select the configuration property and click **Delete Local Copy**.

The following table lists event configuration properties.

**Table 2: Event Configuration Properties**

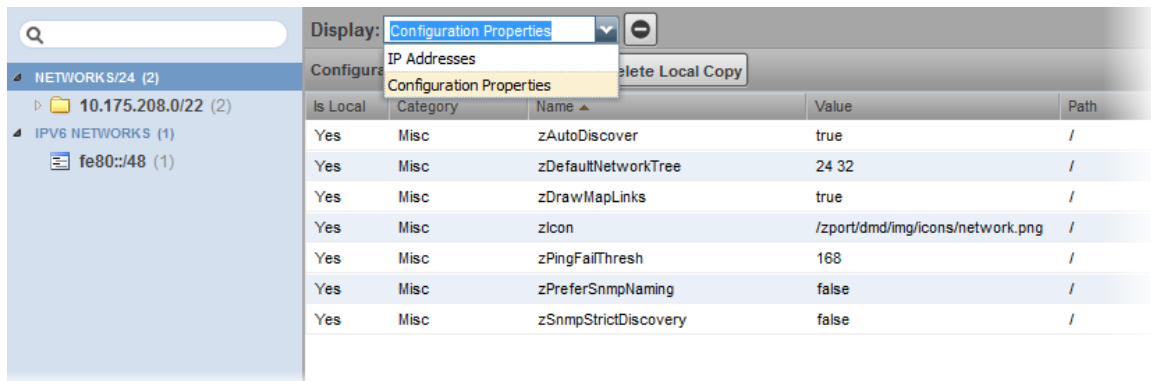
| Property Name            | Property Type | Description   |
|--------------------------|---------------|---|
| zEventAction             | string        | Specifies the database table in which an event will be stored. Possible values are: status, history and drop. Default is status, meaning the event will be an “active” event. History sends the event directly to the history table. Drop tells the system to discard the event.  |
| zEventClearClasses       | lines         | Lists classes that a clear event should clear (in addition to its own class).   |
| zEventSeverity           | int           | Overrides the severity value of events from this class. Possible values are 0-Critical, 1-Error, 2-Warning, 3-Info, 4-Debug, 5-Clear, and -1-Default.   |
| zFlappingIntervalSeconds | int           | Defines the time interval to check for event flapping (changing severity level repeatedly). Default value is 3600 seconds.  |
| zFlappingSeverity        | severity      | Drop-down list to set the severity to check for event flapping. If the severity level on an event changes from this value a certain number of times (zFlappingThreshold) within a certain time range (zFlappingIntervalSeconds) then an event flapping event is generated. Possible values include: 5-Critical, 4-Error, 3-Warning, 2-Info, 1-Debug, and 0-Clear. |

| Property Name      | Property Type | Description   |
|--------------------|---------------|---|
| zFlappingThreshold | int           | Number of times an event severity must flap within an interval. One of the parameters to define in order to generate event flapping events. |

## Network Configuration Properties

To view and edit network configuration properties, select **Infrastructure** from the Navigation menu, and then select **Networks**. The Networks page appears, listing networks by IP address.

**Figure 50: Networks**



You can view and change network configuration property values and inheritance selections. From the Display list of options, select **Configuration Properties**.

The following table lists network configuration properties.

**Table 3: Network Configuration Properties**

| Property Name       | Property Type | Description   |
|---------------------|---------------|---|
| zAutoDiscover       | Boolean       | Specifies whether zendisc should perform auto-discovery on this network. (When performing network discovery, this property specifies whether the system should discover devices and subnetworks on the network.)  |
| zDefaultNetworkTree | lines         | A network subnet is automatically created for each modeled device, based on that device's subnet mask setting. To create higher-level subnets automatically from the discovery and modeling processes, add the specific subnet mask breakpoints. For example: 8, 16. If you then model a device with, for example, an IP address of 192.0.2.0, and a subnet mask of 255.255.255.0 (corresponding to a /24 subnet), device discovery will create a 192.0.0.0/8 network containing 192.0.2.0/16, containing 192.0.2.0/24, containing your device. |
| zDrawMapLinks       | Boolean       | Calculating network links "on the fly" is resource-intensive. If you have a large number of devices that have been assigned locations, then drawing those map   |

| Property Name        | Property Type | Description   |
|----------------------|---------------|---|
|                      |               | links may take a long time. You can use this property to prevent the system from drawing links for specific networks (for example, a local network comprising many devices that you know does not span multiple locations). |
| zIcon                | string        | Use to specify device icons that appear on the device status page, Dashboard, and network map.  |
| zPingFailThresh      | int           | Specifies the number of pings sent without being returned before zendisc removes the device.  |
| zPreferSnmpNaming    | Boolean       | Specifies that when network discovery occurs, it uses the device name comes from SNMP rather than reverse DNS.  |
| zSnmpStrictDiscovery | Boolean       | Specifies that if SNMP does not exist on the device during network discovery, ignore the device.  |

## 5

## Monitoring templates

---

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

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

Templates are divided among three types:

- Device
- Component
- Interface

### Creating Templates

---

You can create a template by overriding an existing template. To override a template:

- 1 Navigate to the template you want to copy.
- 2 From the Action menu, select **Override Template Here**.

The Override Templates dialog appears.

- 3 Select the bound template to override, and then click **Submit**.

The copied template appears in the list of templates as locally defined.

### Renaming Templates

---

To rename an existing template:

- 1 Select Advanced > Monitoring Templates.
- 2 Expand the organizer containing the template to be renamed, and then class containing the template.
- 3 From the Action menu, select **View and Edit Details**.

The Edit Template Details dialog appears.

- 4 Enter a new name in the Name field.
- 5 Click **Submit**.

## Template Binding

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

### Device Templates

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

Common device templates are:

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

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

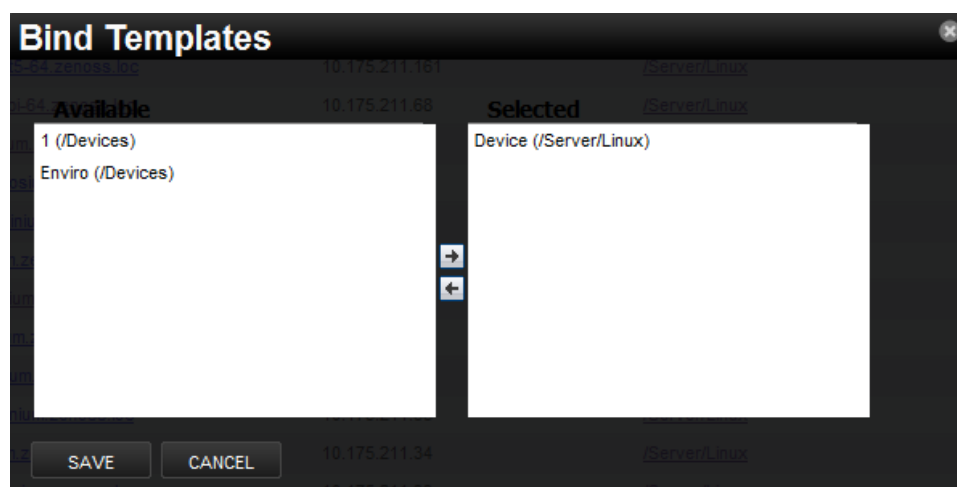
### Binding Templates

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

- 1 From the devices list, select a device class or device.
- 2 On the Overview page, select Bind Templates from the Action menu.

The Bind Templates dialog appears.

**Figure 51:** Bind Templates



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

## Resetting Bindings

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

- 1 Select **Reset Bindings** from the Action menu.

The Reset Template Bindings dialog appears.

- 2 Click **Reset Bindings** to confirm the action.

## Component Templates

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

---

**Note** Component templates should not be manually bound.

---

Common component templates are:

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

## Interface Templates

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

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

## Example: Defining Templates in the Device Hierarchy

---

You add a new device at /Devices/Server/Linux named Example1Server. You have not edited the value of its zDeviceTemplates property, so it inherits the value of "Device" from the root device class (/Devices). Zenoss Core looks to see if there is a template named Device defined on Example1Server itself. There is not, so it checks /Devices/Server/Linux. There is a template named Device defined for that device class, so that template is used for Example1Server. (There also is a template named Device defined at the root level (/Devices), but the system does not use this one because the template at /Devices/Server/Linux overrides it.)

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

---

You want to perform specific monitoring of servers running a certain Web application, but those servers are spread across several different device classes. You create a template at /Devices called WebApplication with the appropriate data sources, thresholds and graphs. You then append the name "WebApplication" to the zDeviceTemplates configuration property for the devices classes, the individual devices running this Web application, or both.

## Basic Monitoring

---

### Availability Monitoring

---

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

- Devices
- Network
- Processes
- Services

Availability monitoring is facilitated by:

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

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

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

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

### Controlling Ping Cycle Time

Follow these steps to modify the ping cycle time:

- 1 Open Zenoss Control Center and click your instance of Zenoss Core to open the application overview page.
- 2 In the Services section, click **zenping** to open the zenping Overview page
- 3 In the Configuration Files section, click **Edit** next to the `/opt/zenoss/etc/zenping.conf` file. The Edit Configuration window appears.
- 4 Uncomment the `zenhubpinginterval` line and edit the default value of 30 to your desired ping cycle time.
- 5 Click **Save**.



## Using the Predefined /Ping Device Class

The /Ping device class is a configuration for devices that you want to monitor only for availability. The system does not gather performance data for devices placed in this class. You can use the /Ping device class as a reference for your own configuration; or, if you have a device that you want to monitor solely for availability, you can place it under this class.

## Monitoring Processes

Zenoss Core provides process availability monitoring for hosts that support SNMP or SSH access. Process monitoring features include:

- Process classes, defined by Python regular expressions. Classes may generate one or more process sets, each containing one or more process instances.
- Process sets may include process instances running on multiple hosts. This captures related or redundant processes, enabling a more wholistic view of data center services.
- Process set names, to replace the often-cryptic names of process instances with descriptive labels.
- Process set locking, to maintain continuity of data collection if the members of a given process set are not running during modeling.
- A testing dialog, to discover and refine the sets a class generates.

Use the Processes page (**Infrastructure > Processes**) to create and manage process classes and process sets.

**Figure 52:** Processes page

The screenshot displays the 'Processes' page in the Zenoss Core interface. On the left, a sidebar shows a tree view of process class organizers: Apache (1), MySQL (1), Oracle (0), and Zenoss (32). Below this is a table listing process classes and their set counts, such as httpd (0), mysqld (0), rrdcached (0), zenactiond (0), zencatalogservice (0), zencommand (0), zeneventd (0), zeneventlog (0), zeneventserver (0), zenhub (0), zenjmx (0), zenjobs (0), zenjservice (0), zenmail (0), zenmailtx (0), zenmodeler (0), zenperform (0), zenping (0), zenpop3 (0), zenprocess (0), zenrender (0), zenstatus (0), zensyslog (0), zentrap (0), and zenunknown (0). The main content area is a form for configuring a process class. It includes fields for 'Process Class Name' (set to 'Processes'), 'Description', and 'Process Count Threshold' (Minimum and Maximum). There are several monitoring options with 'Set Local Value' and 'Inherit Value' radio buttons: 'Enable Monitoring? (zMonitor)' is set to 'Yes', 'Send Event on Restart? (zAlertOnRestart)' is set to 'No', 'Failure Event Severity (zFailSeverity)' is set to 'Error', 'Lock Process Components? (zModelerLock)' is set to 'Unlocked', and 'Send an event when action is blocked? (zSendEventWhenBlockedFlag)' is set to 'No'. At the bottom, there is a 'Display' dropdown set to 'Process Sets' and a table with columns: Name, Device, Min Threshold, Max Threshold, Monitored, and Status. The table currently shows 'NO RESULTS' and '0 Jobs'.

The tree view shows process class organizers (at the top) and the list of process classes in each organizer (the rest of the view). You may filter the list with the active search field, at the top of the list.

## Example: Creating a process class

This section provides an example of using process availability monitoring to create a new process class, for database processes. The database runs on a Linux host, and the following output is a partial list of the results of the `ps axho args` command on the database host. Process monitoring uses the output of that command (or its equivalent) as input for regular expression matching.

```
ora_pmon_orcl ora_psp0_orcl ora_vktm_orcl ora_gen0_orcl ora_diag_orcl
ora_dbrm_orcl ora_dia0_orcl ora_mman_orcl ora_dbw0_orcl ora_lgwr_orcl
ora_ckpt_orcl ora_smon_orcl ora_reco_orcl ora_mmon_orcl ora_mmln_orcl
ora_d000_orcl ora_s000_orcl ora_s001_orcl ora_s002_orcl ora_s003_orcl
ora_s004_orcl ora_s005_orcl ora_s006_orcl ora_s007_orcl ora_s008_orcl
ora_s009_orcl ora_p000_orcl ora_p001_orcl ora_p002_orcl ora_p003_orcl
ora_p004_orcl ora_gmnc_orcl ora_n000_orcl ora_l000_orcl ora_l001_orcl
ora_l002_orcl ora_l003_orcl
```

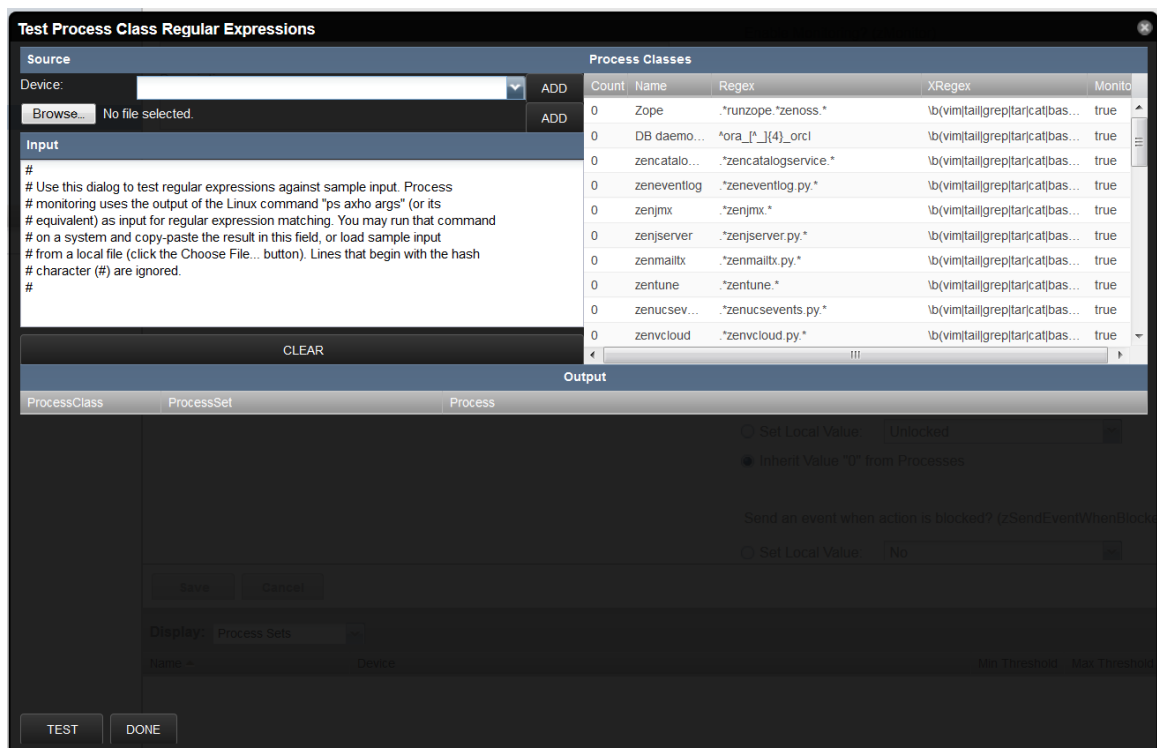
The following subsections provide procedures for creating a process class that captures a selection of the preceding process instances in process sets.

### Test existing process classes

The existing process classes may already capture the process sets you want. Follow these steps to test the existing process classes.

- 1 Log in to the Zenoss Core user interface.
- 2 Navigate to **Infrastructure > Processes**.
- 3 In the lower-left corner of the tree view, click the Action menu, and select **Test All Process Classes Regular Expressions**.

**Figure 53:** Test all process classes dialog



The top-right portion of the dialog displays all of the process classes, in the order in which they are evaluated.

- 4 Select the process names in the previous section, and copy them into a paste buffer.
- 5 In the **Test Process Class Regular Expressions** dialog, select all of the existing text in the **Input** area, and then paste the process names from the buffer. Alternatively, you may paste the process names into an empty file, save the file on the system from which your browser is launched, and then use the **Choose File** button to insert the process names.
- 6 At the bottom-left corner of the dialog, click **Test**.
- 7 If any process sets are created, they are displayed in the list area, above the **Test** button.

### Create an organizer

Complete the previous section, and then follow these steps to create a process class organizer.

- 1 Log in to the Zenoss Core user interface.
- 2 Navigate to **Infrastructure > Processes**.
- 3 In the lower-left corner of the tree view, click the **Add** menu, and select **Add Process Class Organizer**.
- 4 In the **Add Process Class Organizer** dialog, enter `Database`, and then click **Submit**.

### Create a process class

Complete the previous section, and then follow these steps to create a process class.

- 1 At the top of the tree view, double-click **Processes**, the root organizer to open it.
- 2 Select **Database**.
- 3 In the lower-left corner of the tree view, click the **Add** menu, and select **Add Process Class**.
- 4 In the **Add Process Class** dialog, enter `DB daemons`, and then click **Submit**.
- 5 At the top of the tree view, double-click **Processes** to open it, and then select **Database**.

### Define the regular expression series of a process class

Complete the previous section, and then follow these steps to create the series of regular expressions that define a process class, and generate process sets.

- 1 In the list area of the tree view, select **DB daemons**.

**Figure 54:** Process class definition page

- 2 In the **Description** field, enter Database daemons.
- 3 In the **Include processes like** field, replace DB daemons with a *Python regular expression* that selects the database processes. For example, `ora_[^_]{4}_orcl`.

The example regular expression selects all of the processes in the sample process instance list.

- 4 In the **Exclude processes like** field, enter a regular expression to remove processes from the results of the preceding regular expression. The default entry excludes common user commands.

The default entry does not exclude any of the processes in the sample process instance list.

- 5 The next two fields, **Replace command line text** and **With**, work together to simplify the names of process sets.

- In the **Replace command line text** field, enter a regular expression containing one or more pattern groups.
- In the **With** field, enter replacement text, along with the sequence number of one or more of the pattern groups defined in the previous field.

For example, to create four pattern groups for database processes, enter the following regular expression in the **Replace command line text** field:

```
^(ora_)([a-z])(.{4})(orcl)
```

To use two of the pattern groups in the replacement text, enter the following text in the **With** field:

```
DB [\4] daemons starting with [\2]
```

Each unique replacement generated by the combination of the text plus the inserted pattern sequences becomes a process set. In the case of this example, pattern group 4 does not vary, but pattern group 2 does. So the

number of process sets generated by this class will equal the number of unique alphabetic characters found in the first position after the first underscore.

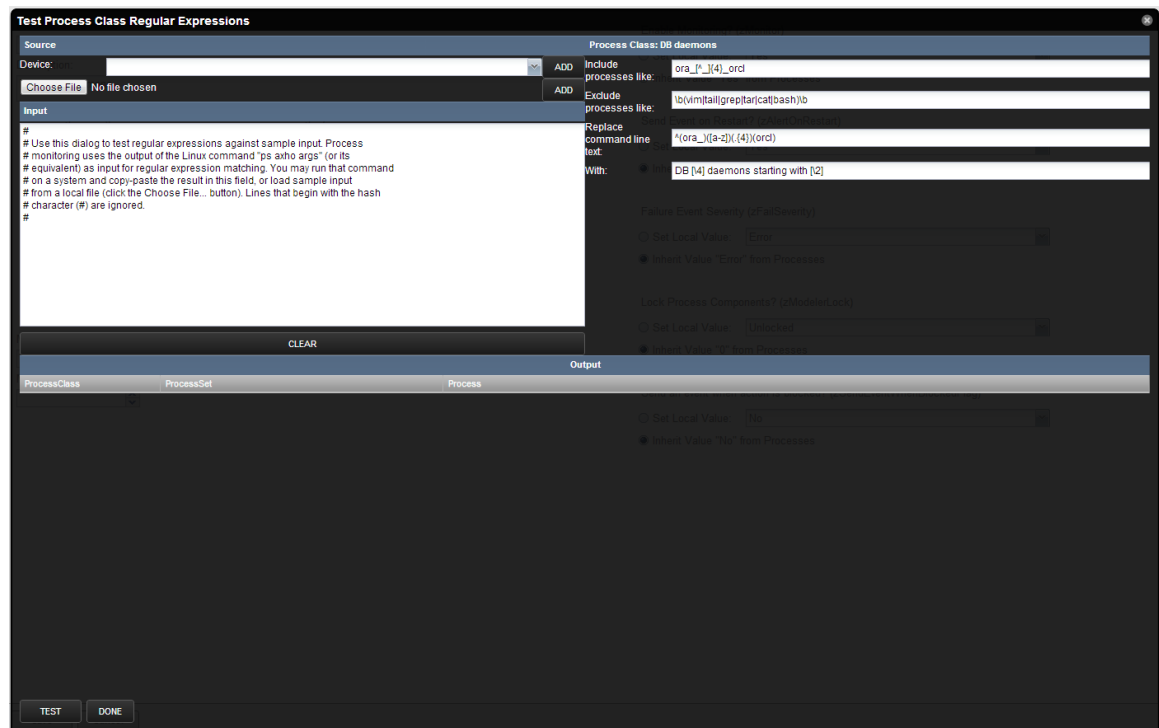
- 6 Click **Save**.

### Test a process class

Complete the previous section, and then follow these steps to test a single process class.

- 1 In the lower-left corner of the tree view, click the Action menu, and select **Test Process Class Regular Expressions**.

**Figure 55:** Test process class dialog



The top-right portion of the dialog displays the regular expression series that defines this process class.

- 2 Select the process names in the previous section, and copy them into a paste buffer.
- 3 In the **Input** area, select all of the existing text, and then paste the process names from the buffer.
- 4 At the bottom-left corner of the dialog, click **Test**.
- 5 The **Output** area displays each individual match, along with the count of unique process sets.

You may refine the regular expressions and retest as often as you like.

- 6 Click **Done**.

Changes made to regular expressions in this dialog are copied to the process class definition page. However, the changes are not saved until you click the **Save** button on that page.

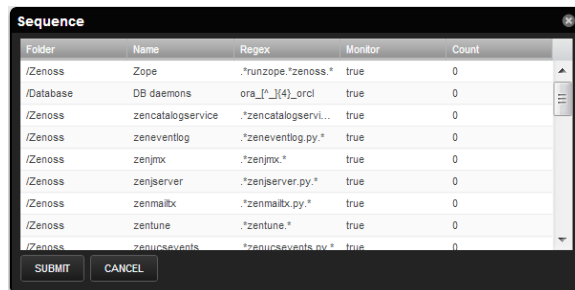
### Test and review the process class sequence

The order in which process classes are evaluated is significant. During modeling, each time a process matches a class, the matching process is put into a process set, and then removed from the list of processes that are passed on to the next class in the sequence. New process classes are inserted into the process class sequence automatically, and may not be in the appropriate position in the sequence.

Complete the previous section, and then follow these steps to test and review the process class sequence.

- 1 In the lower-left corner of the tree view, click the Action menu, and select **Test All Process Classes Regular Expressions**.
- 2 Select the process names in the previous section, and copy them into a paste buffer.
- 3 In the **Input** area, select all of the existing text, and then paste the process names from the buffer.
- 4 At the bottom-left corner of the dialog, click **Test**.
- 5 The number of processes matched and process sets created in this test should be identical to the results of testing the process class alone. If they are not, follow these steps to adjust the process class sequence.
  - a In the **Test Process Class Regular Expressions** dialog, click **Done**.
  - b From the Action menu, select **Change Sequence**.

**Figure 56:** Sequence dialog



- c Scroll through the list of process classes, and then select the class to move.
  - d Drag the class to an earlier (higher) location in the sequence.
  - e Click **Submit**.
- 6 Re-open the **Test Process Class Regular Expressions** dialog, and re-test the sequence.

### Test the process class on a host

Process sets are created during modeling. To test a process class, choose a device host that is configured for SNMP or SSH access, and model it manually.

---

**Note** For more information about device support for process monitoring, refer to the release notes.

---

Complete the previous section, and then follow these steps to test the process class on a host.

- 1 Navigate to **Infrastructure > Devices**.
- 2 Select a host that is configured for SNMP or SSH access, and is running process that match the new class.

For example, the list of processes used in this section is collected from a VirtualBox virtual appliance downloaded from the [Oracle Technology Network](#).

- 3 Open the host's **Overview** page. From the Action menu, select **Model Device**.

When modeling completes, the **OS Processes** section of the tree view is updated to include the new process sets.

- 4 Navigate to **Infrastructure > Processes**.
- 5 In the tree view, select the **DB daemons** class.

**Figure 57: Process class page with process sets**

Process Class Name:  
DB daemons

Description:  
Database daemons

Matching Rules (performance metrics will start over if changed)

Include processes like:  
ora\_[\*][4]\_orcl

Exclude processes like:  
\\vim\\tail\\grep\\tar\\cat\\bar

Replace command line text:  
^(ora\_[a-z])([4])(orcl)

With:  
DB [4] daemons starting

Process Count Threshold

Minimum:  
[ ]

Maximum:  
[ ]

Enable Monitoring? (zMonitor)

☐ Set Local Value: Yes

☒ Inherit Value "Yes" from Processes

Send Event on Restart? (zAlertOnRestart)

☐ Set Local Value: Yes

☒ Inherit Value "No" from Processes

Failure Event Severity (zFailSeverity)

☐ Set Local Value: Error

☒ Inherit Value "Error" from Processes

Lock Process Components? (zModelerLock)

☐ Set Local Value: Unlocked

☒ Inherit Value "0" from Processes

Send an event when action is blocked? (zSendEventWhenBlockedFlag)

☐ Set Local Value: No

☒ Inherit Value "No" from Processes

Save Cancel

Display: Process Sets

| Name                               | Device        | Min Threshold | Max Threshold | Monitored | Status |
|------------------------------------|---------------|---------------|---------------|-----------|--------|
| DB [orc] daemons starting with [c] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [d] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [g] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [i] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [l] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [m] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [n] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [p] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [q] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [r] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [s] | 10.87.110.103 |               |               | true      | Up     |
| DB [orc] daemons starting with [v] | 10.87.110.103 |               |               | true      | Up     |

Display: 1 - 13 of 13 Rows

The process sets found on the host are displayed in the list at the bottom of the page.

## Process class options

The process class page includes the options described in the following sections.

### Process Count Threshold

You may set minimum and maximum values for the number of process instances included in a process set. The threshold values apply to all of the process sets generated by a class. The minimum and maximum values are inclusive. That is, if the minimum is 3 and the maximum is 5, then 3, 4, and 5 are all valid process instance counts.

You may define a threshold as an exclusive range. If the minimum is 5 and the maximum is 3, then 4 is an invalid process instance count.

### Monitoring Options

#### ■ Enable Monitoring (zMonitor)

To disable monitoring for all process sets generated by this class, set the local value to No.

#### ■ Send Event on Restart (zAlertOnRestart)

To send an event when monitoring restarts, set the local value to Yes.

#### ■ Failure Event Severity (zFailSeverity)

To specify a non-default event severity for the failure of process sets generated by this class, set a local value.

## Process Set Locking

Process sets are generated at modeling time. Since modeling recurs regularly, a given modeling run may result in a missing process set, due to a transient absence of one or more process instances. To prevent this from happening, set the local value of the **Lock Process Components? (zModelerLock)** field to one of the following options.

- **Lock from Deletes**

Prevent deletion of process sets generated by this process class if modeling returns empty sets.

- **Lock from Updates**

Prevent updates to process sets generated by this process class if modeling returns new process sets.

The final option, **Send an event when action is blocked? (zSendEventWhenBlockedFlag)**, is used only when a process set is locked. If you lock the process sets of a class, you may set the local value of this field to Yes, and an event will be created when a process set would have been either deleted or updated during modeling.

## Monitoring IP Services

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

**Figure 58:** IP Services

| Name          | Port | Count |
|---------------|------|-------|
| 1ci-smcs      | 3091 | 0     |
| 3com-amp3     | 629  | 0     |
| 3com-net-mgmt | 2391 | 0     |
| 3Com-rsld     | 1742 | 0     |
| 3com-tsmux    | 106  | 0     |
| 3com-webview  | 2339 | 0     |
| 3comfaxpc     | 3446 | 0     |
| 3d-rtsd       | 2323 | 0     |
| 3ds-lm        | 1538 | 0     |
| 3i-l1         | 1511 | 0     |
| 3m-image-lm   | 1550 | 0     |

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

The details area shows:

- Service class description
- TCP port
- Associated service keys

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



The lower section of the page lists currently running services in this class (by device), and shows their monitoring status. You can also display Configuration Properties by selecting that from the drop-down list.

## Enabling IP Service Monitoring

You can choose to monitor:

- Individual services
- Service classes

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

---

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

---



---

**Note** When adding a new IP service that uses a port value higher than 1024, you need to increase the value of `zIpServiceMapMaxPort` to a number higher than the port you are monitoring.

---

To enable monitoring for a service class or service:

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

## Using the Predefined /Server/Scan Device Class

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

## Monitoring Windows Services

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

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

The details area shows:

- Service class description
- Associated service keys

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

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

## Enabling Windows Service Monitoring

You can choose to monitor:

- Individual services
- Service classes

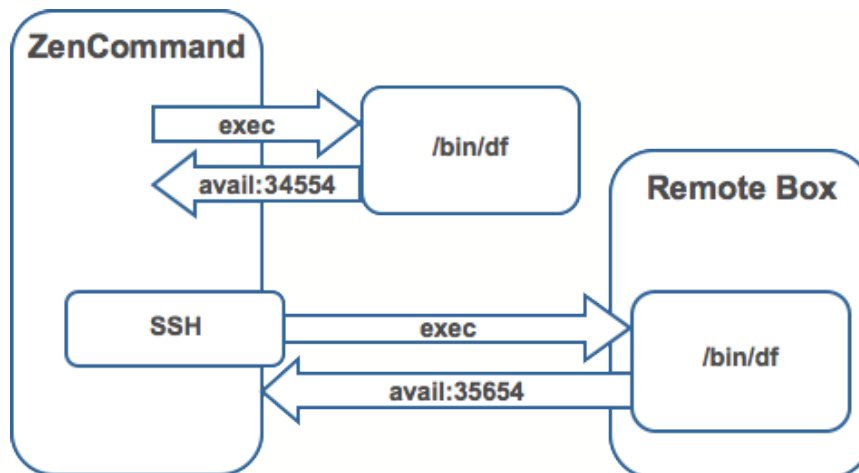
To enable monitoring for a service class or service:

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

## Monitoring Using ZenCommand

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

**Figure 59:** Running ZenCommand



## Plugin Format for ZenCommands

Nagios® plugins are configured by using a command template.

A template named “Device” will bind to all devices below the template definition. Within each template is a list of commands that will run. The commands can be any program that follows the Nagios® plug-in standard. Inputs are command line arguments; output is the first line of stdout, plus a return code.

---

**Note** Zenoss Core return codes differ from Nagios return codes, as follows:

---

| Value | Zenoss Core   | Nagios   |
|-------|---------------|----------|
| 0     | Clear         | OK       |
| 1     | Data Source   | WARNING  |
| 2     | Data Source+1 | CRITICAL |
| 3     | Data Source   | UNKNOWN  |

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

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

A Nagios® command has several fields:

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

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

- zCommandPath – Path to the zencommand plug-ins on a given box it comes from the configuration property zCommandPath. zCommandPath is automatically added to a command if a path is absent from the beginning of the command.
- devname – Device name of the device against which the command is being evaluated.
- dev – Device object against which the command is being evaluated.
- here – Context of evaluation. For a device, this is equivalent to dev for a component (such as a file system 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).

## Testing ZenCommands

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

From the command line, run:

```
zentestcommand -d
    DeviceID--datasource=
    DataSourceName
```

where DeviceID is the ID of 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.

## SNMP Monitoring

---

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

- 1 SSH to the Zenoss Core instance.

Use Username: root

Password: zenoss

- 2 Run the command `snmp get` for one of the OIDs

In this case, use the command:

```
$ snmpget -v 2c -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 commands to gather certain information.

- a Walk a basic system MIB.

```
snmpwalk -v 2c -cpublic <device name> system
```

- b Walk an interface description

```
snmpwalk -v 2c -cpublic <device name> ifDescr
```

- c Get a single value.

```
snmpget -v 2c -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
```

## Monitoring Devices Remotely Through SSH

---

You can monitor devices remotely through SSH. Follow the steps in the following sections to set up remote monitoring.

### Changing Zenoss Core to Monitor Devices Remotely Using SSH

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

- 1 Navigate to the device class path you want to monitor remotely. You can apply this monitoring for a device or a device class path.
- 2 Change the configuration properties value for the group. After selecting the device class, click **Details**, and then select Configuration Properties.

The Configuration Properties page appears.

**Figure 60: Device Class Configuration Properties**

| Discovered <span>SEE ALL</span> |                  | Delete Local Copy           |                    |      |
|---------------------------------|------------------|-----------------------------|--------------------|------|
| Is Local                        | Category         | Name                        | Value              | Path |
|                                 | Cisco            | zCiscoACEUseSSL             | true               | /    |
|                                 | Cisco            | zCiscoRemodeEventClassKeys  |                    | /    |
|                                 | Cisco UCS        | zCiscoUCSCIMCEventsInterval | 60                 | /    |
|                                 | Cisco UCS        | zCiscoUCSCIMCPerInterval    | 300                | /    |
|                                 | Cisco UCS        | zCiscoUCSMManagerPassword   | *****              | /    |
|                                 | Cisco UCS        | zCiscoUCSMManagerPort       | 443                | /    |
|                                 | Cisco UCS        | zCiscoUCSMManagerUseSSL     | true               | /    |
|                                 | Cisco UCS        | zCiscoUCSMManagerUser       | admin              | /    |
|                                 | Modeler Controls | zCollectorClientTimeout     | 180                | /    |
|                                 | Modeler Controls | zCollectorDecoding          | utf-8              | /    |
|                                 | Misc             | zCollectorLogChanges        | false              | /    |
|                                 | zencommand       | zCommandCommandTimeout      | 15                 | /    |
|                                 | zencommand       | zCommandExistenceTest       | test -f %s         | /    |
|                                 | zencommand       | zCommandLoginTimeout        | 10                 | /    |
|                                 | zencommand       | zCommandLoginTries          | 1                  | /    |
|                                 | zencommand       | zCommandPassword            |                    | /    |
|                                 | zencommand       | zCommandPath                | \$ZENHOME/libexec  | /    |
|                                 | zencommand       | zCommandPort                | 22                 | /    |
|                                 | zencommand       | zCommandProtocol            | ssh                | /    |
|                                 | zencommand       | zCommandSearchPath          |                    | /    |
|                                 | zencommand       | zCommandUsername            |                    | /    |
|                                 | Control Center   | zControlCenterHost          | \$(here/managerip) | /    |

You must make changes to the following configuration properties:

- zCollectorPlugins
- zCommandPassword
- zCommandPath
- zCommandUsername
- zSnmppMonitorIgnore

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

| Configuration Properties | Value                                     |
|--------------------------|---|
| zCollectorPlugins        | snmp   portscan                           |
| zCommandPassword         | The SSH password for the remote machine.  |
| zCommandPath             | The path to zenplugin.py                  |
| zCommandUsername         | The SSH user name for the remote machine. |
| zSnmppMonitorIgnore      | True                                      |

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

Run the command:

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

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

## Using the Predefined /Server/Cmd Device Class

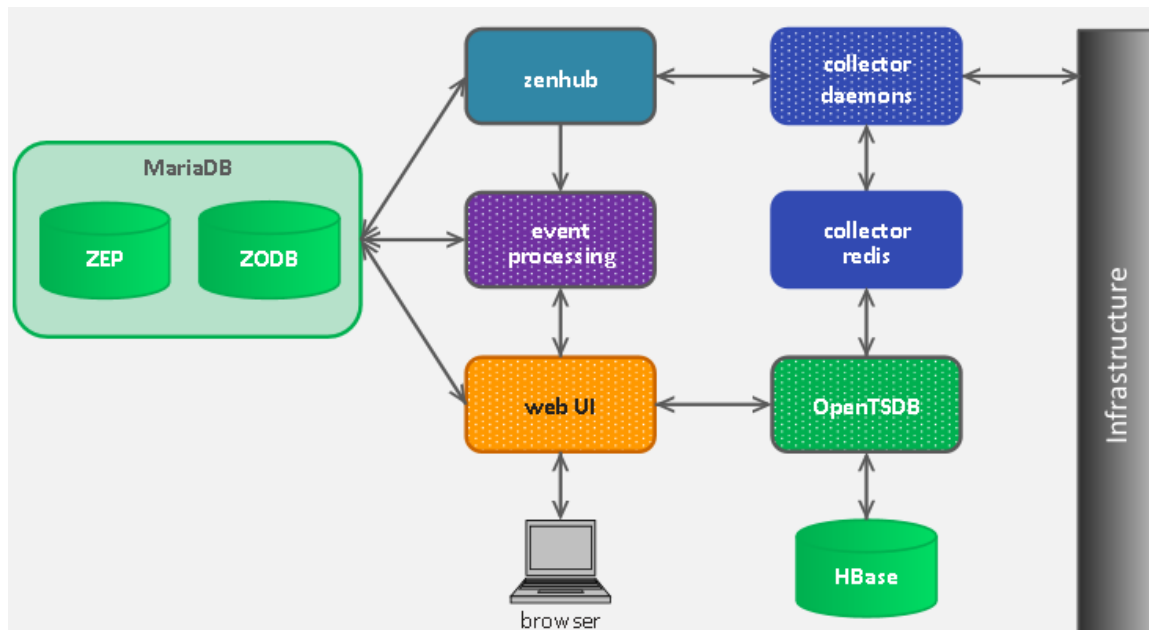
The /Server/Cmd device class is an example configuration for modeling and monitoring devices using SSH. The configuration properties have been modified (as described in the previous sections), and device, file system, and Ethernet interface templates that gather data over SSH have been created.

You can use this device class as a reference for your own configuration; or, if you have a device that needs to be modeled or monitored via SSH/Command, you can place it under this device class to use the pre-configured templates and configuration properties. You must set the `zCommandUsername` and `zCommandPassword` properties to the appropriate SSH login information for each device.

# Performance Monitoring

Zenoss Core stores device and daemon performance metrics directly into an time series database (OpenTSDB) that is run on top of an HBase instance. Writing directly into OpenTSDB has eliminated the need for RRD files to be stored on the collectors. The following image shows how the collector daemons fit into the data collection portion of the Zenoss Core architecture.

**Figure 61:** Data Collection Simplified Architecture



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

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

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

## About Monitoring Templates

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

Templates comprise three types of objects:

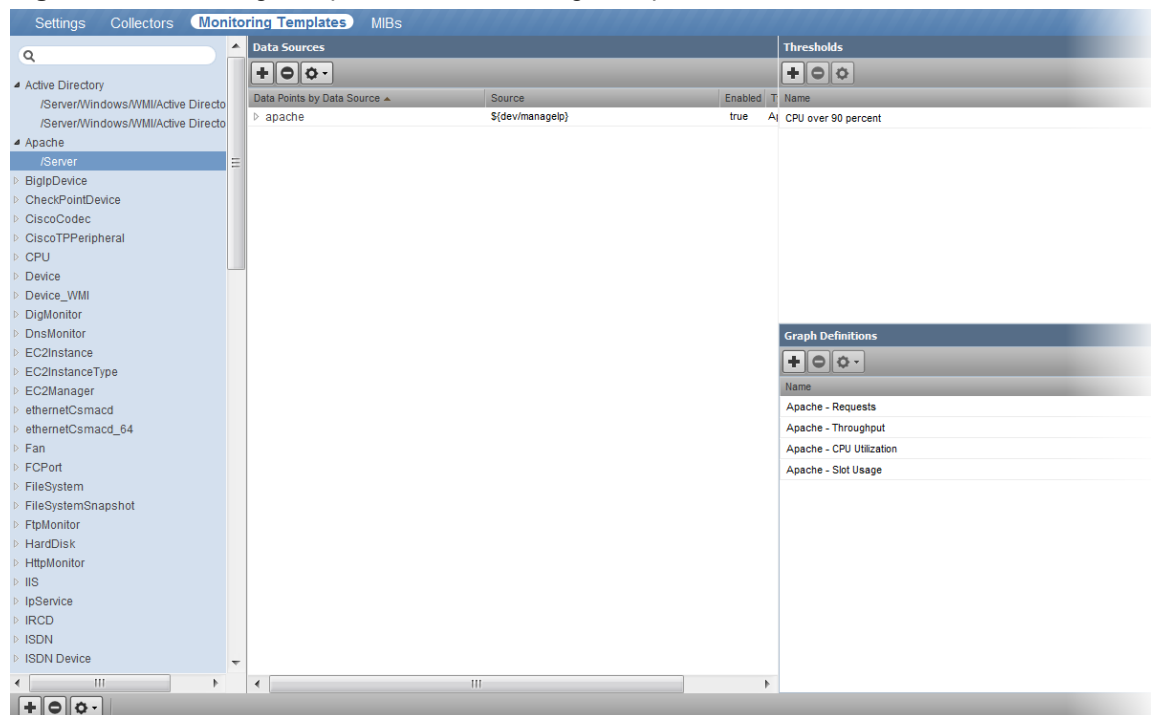
- **Data Sources**- Specify the exact data points to collect, and the method to use to collect them.
- **Thresholds**- Define expected bounds for collected data, and specify events to be created if the data does not match those bounds.
- **Graph Definitions**- Describe how to graph the collected data on the device or device components.

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

## Viewing Monitoring Templates

To view monitoring templates, select Advanced from the Navigation menu, and then select Monitoring Templates.

**Figure 62: Monitoring Template for Load Average Graph**



## Template Binding

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

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



After defining the list, the system locates templates that match the names on the list. For each name, it searches the device and then searches the device class hierarchy. Zenoss Core 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.

## Binding Templates

To edit the templates bound to a device:

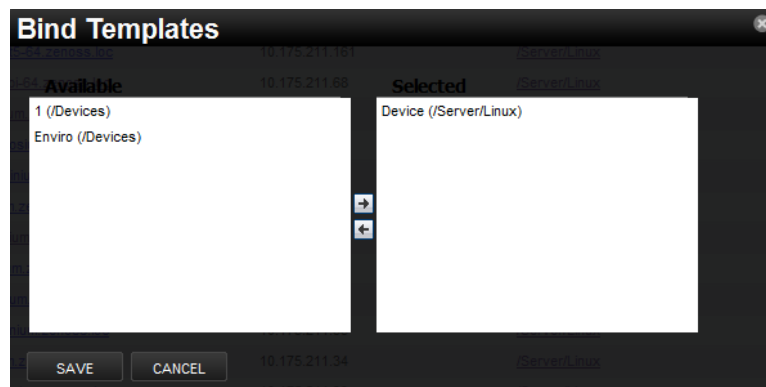
- 1 From the Navigation menu, select Infrastructure.

The device list appears.

- 2 Select a device in the device list.
- 3 Select Bind Templates from the Action menu.

The Bind Templates dialog appears.

**Figure 63:** Bind Templates



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

## Data Sources

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

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

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

## Adding a Data Source

To add a data source to a monitoring template:

- 1 Select Advanced from the Navigation menu, and then select Monitoring Templates.
- 2 In the tree view, select the monitoring template to which you want to add a data source.

- 3 In the Data Sources area, click the Add button.

The Add Data Source dialog appears.

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

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

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

The Edit Data Source dialog appears.

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

## 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.

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

To add a data point to a data source:

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

The Add Data Point dialog appears.

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

---

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

---

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

- **Name-** Displays the name you entered in the Add a New DataPoint dialog.
- **RRD Type-** Specify the RRD data source type to use for storing data for this data point. Available options are:
  - **COUNTER-** Saves the rate of change of the value over a step period. This assumes that the value is always increasing (the difference between the current and the previous value is greater than 0). Traffic counters on a router are an ideal candidate for using COUNTER.
  - **GAUGE-** Does not save the rate of change, but saves the actual value. There are no divisions or calculations. To see memory consumption in a server, for example, you might want to select this value.

---

**Note** Rather than COUNTER, you may want to define a data point using DERIVED and with a minimum of zero. This creates the same conditions as COUNTER, with one exception. Because COUNTER is a "smart" data type, it can wrap the data when a maximum number of values is reached in the system. An issue can occur when there is a loss of reporting and the system (when looking at COUNTER values) thinks it should wrap the data. This creates an artificial spike in the system and creates statistical anomalies.

---

- **DERIVE-** Same as COUNTER, but additionally allows negative values. If you want to see the rate of change in free disk space on your server, for example, then you might want to select this value.
- **ABSOLUTE-** Saves the rate of change, but assumes that the previous value is set to 0. The difference between the current and the previous value is always equal to the current value. Thus, ABSOLUTE stores the current value, divided by the step interval.

- **Create Command**- Enter an RRD expression used to create the database for this data point. If you do not enter a value, then the system uses a default applicable to most situations.

For details about the `rrdcreate` command, go to:

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

- **RRD Minimum**- Enter a value. Any value received that is less than this number is ignored.
- **RRD Maximum**- Enter a value. Any value received that is greater than this number is ignored.

6 Click **Save** to save the defined data point.

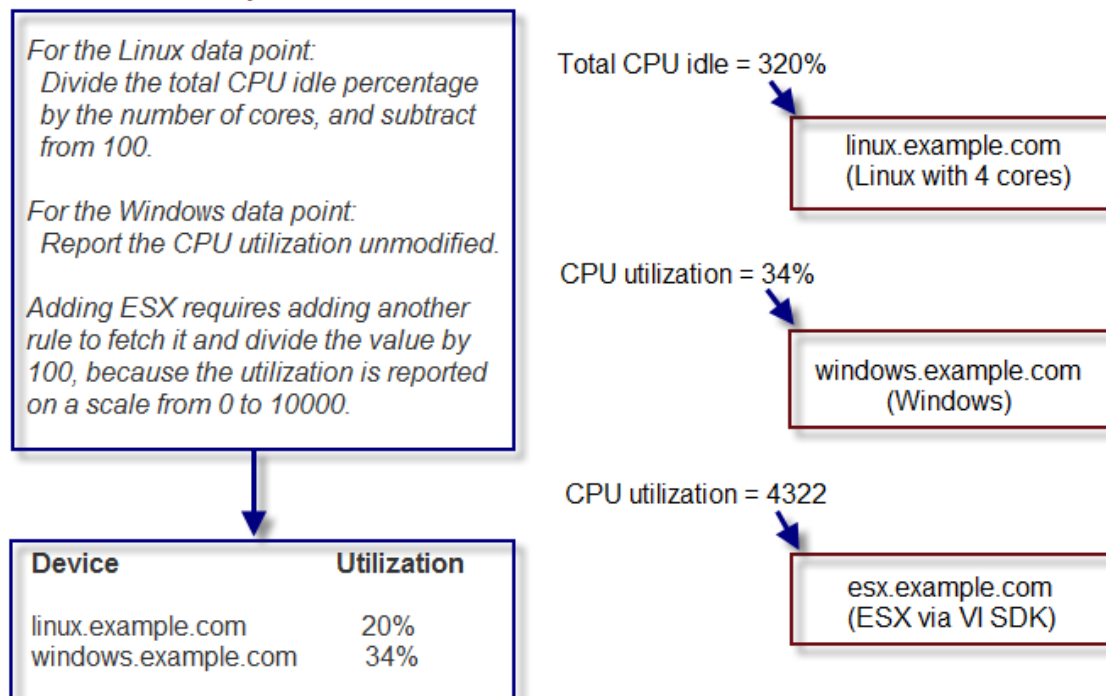
## Data Point Aliases

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

The addition of a data point requires changing the report.

**Figure 64: CPU Utilization Report**

### CPU Utilization Report



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

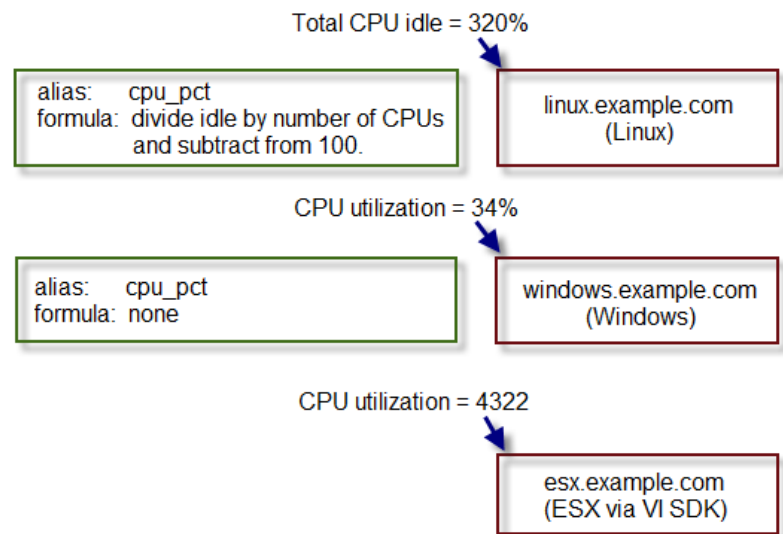
An alias-based report looks up the data points that share a common alias string, and then uses them. This approach allows you to add data points without changing the report.

**Figure 65:** Alias-Based CPU Utilization Report**CPU Utilization Report** (using aliases)

Get values for all data points with the `cpu_pct` alias.

Adding ESX to the report is now just a matter of applying the alias `cpu_pct` with a formula for dividing by 100.

| Device              | Utilization |
|---------------------|-------------|
| linux.example.com   | 20%         |
| windows.example.com | 34%         |



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

## Alias Formula Evaluation

The system evaluates the alias formula in three passes.

### Reverse Polish Notation

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

```
1024, *
```

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

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

### Using TALES Expressions in Alias Formulas

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

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

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

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

<http://wiki.zope.org/ZPT/TALESSpecification13>

## Using Python in Alias Formulas

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

```
__EVAL:
```

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

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

## Adding a Data Point Alias

To add an alias to a data point:

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

The Edit Data Point dialog appears.

- 3 Enter the alias name and the formula.

---

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

---

**Figure 66:** Add Data Point Alias

- 4 Click **Save**.

## Reports That Use Aliases

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

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

### CPU Utilization

| Alias           | Expected Units |
|-----------------|----------------|
| loadAverage5min | Processes      |
| cpu_pct         | Percent        |

## Thresholds

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

## MinMax Threshold

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

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

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

## Adding Thresholds

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

- 1 Select Advanced from the Navigation menu, and then select Monitoring Templates.
- 2 In the Thresholds area, click the **Add** icon.

The Add Threshold dialog appears.

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

The Edit Threshold dialog appears.

**Figure 67: Edit Threshold**

**Edit Threshold**

Name:

Type: MinMaxThreshold

DataPoints:

| Available                           | Selected                |
|-------------------------------------|-------------------------|
| ifInErrors_ifInErrors               | ifInOctets_ifInOctets   |
| ifInUcastPackets_ifInUcastPackets   | ifOutOctets_ifOutOctets |
| ifOperStatus_ifOperStatus           |                         |
| ifOutErrors_ifOutErrors             |                         |
| ifOutOctets_test                    |                         |
| ifOutUcastPackets_ifOutUcastPackets |                         |

Severity:

☒ Enabled

Minimum Value:

Maximum Value:

Event Class:

Escalate Count:

5 Enter or select values to define the threshold:

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

When using a Python expression, the variable *here* references the device or component for which data is being collected. For example, an 85% threshold on an interface might be specified as:

```
here.speed * .85/8
```

The division by 8 is because interface speed frequently is reported in bits/second, where the performance data is bytes/second.

- **Maximum Value**- If this field contains a value, then each time one of the selected data points goes above this value an event is triggered. This field may contain a number or a Python expression.
- **Event Class**- Select the event class of the event that will be triggered when this threshold is breached.
- **Escalate Count**- Enter the number of consecutive times this threshold can be broken before the event severity is escalated by one step.

- 6 Click **Save** to save the newly defined threshold.

## Performance Graphs

---

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

Graphs are no longer static images, but are rendered using the NVD3.js JavaScript library.

To define a graph:

- 1 Select Advanced from the Navigation menu, and then select Monitoring Templates.
- 2 In the Graph Definitions area, click the **Add** icon.

The Add Graph Definition dialog appears.

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

**Figure 68:** Graph Definition

**View and Edit Graph Definition**

Name:

Height:

Width:

Units:

Logarithmic Scale: ☐

Base 1024: ☐

Min Y:

Max Y:

Has Summary: ☒

- 5 Click **Submit** to save the graph.



## Graph Points

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

From the Graph Definitions area of the Monitoring Templates page:

- 1 Select Manage Graph Points from the Action menu.

The Manage Graph Points dialog appears.

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

The new graph point appears in the Graph Points list.

---

**Note** Thresholds are always drawn before other graph points.

---

### Re-sequencing Graph Points

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

### DataPoint Graph Points

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

#### Adding DataPoint Graph Points

To define a DataPoint graph point:

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

The Add Data Point dialog appears.

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

#### Editing DataPoint Graph Points

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

- **Name**- This is the name that appears on the Graph Definition page. By default, it appears in the graph legend.
- **Line Type**- Select Line to graph the data as a line. Select Area to fill the area between the line and the horizontal axis with the line color. Select None to use this data point for custom RRD commands and do not want it to be explicitly drawn.
- **Line Width**- Enter the pixel width of the line.
- **Stacked**- If selected, then the line or area is drawn above the previously drawn data. At any point in time on the graph, the value plotted for this data is the sum of the previously drawn data and the value of this data point now. You might set this value, for example, to asses total packets if measuring packets in and packets out.
- **Format**- Specify the RRD format to use when displaying values in the graph summary. For more information on formatting strings, go to:

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

- **RPN**- Optionally enter an RPN expression that alters the value of the data being graphed for the data point. For example, if the data is stored as bits, but you want to graph it as bytes, enter an RPN value of "8,/" to divide by 8. For more information about RPN notation, go to:

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

- **Limit**- Optionally specify a maximum value for the data being graphed.
- **Consolidation**- Specify the RRD function used to graph the data point's data to the size of the graph. Most of the time, the default value of AVERAGE is appropriate.
- **Color**- Optionally specify a color for the line or area. Enter a six-digit hexadecimal color value with an optional two-digit hex value to specify an alpha channel. An alpha channel value is only used if 'stacked' is True.
- **Legend**- Name to use for the data in the graph legend. By default, this is a TALEX expression that specifies the graph point name. The variables available in this TALEX 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.

## Editing Threshold Graph Points

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

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

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

## Editing Custom Graph Points

Custom graph points allow you to insert specific RRD graph commands into the graph definition.

For details on DEF, CDEF, and VDEF commands, go to:

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

For details on other RRD commands, go to:

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

## Performance Data Retention

---

Zenoss Core stores all performance data (a.k.a., metrics) in HBase using OpenTSDB. The default retention policy saves performance data for 90 days. To change the default, the time to live (TTL) must be adjusted on the OpenTSDB column families in HBase.

---

**Note** TTL is defined in seconds.

---

Once a TTL value is changed, the data retention will adjust on the next major HBase compaction, which by default is once per day.

## Changing the Performance Data Retention Time

To change the performance data retention time from the default value of 90 days:

- 1 Log in to the Zenoss Control Center master host as a user with `sudo` and `docker` privileges.

- 2 Stop the opentsdb writer service.

```
serviced service stop opentsdb/writer
```

- 3 Execute the following command to list all the services and their SERVICEID values. Take note of the SERVICEID for the opentsdb reader service. It will be used as an argument in the following step.

```
serviced service list
```

- 4 Execute the following command where \$id is the opentsdb reader SERVICEID and \$ttl is your desired TTL value, in seconds.

```
serviced service shell $id /opt/opentsdb/set-opentsdb-table-ttl.sh  
$ttl
```

- 5 Start the opentsdb writer service.

```
serviced service start opentsdb/writer
```

# Event Management

---

Events, and the graphs generated from performance monitoring, are the primary operational tools for understanding the state of your environment.

## Basic Event Fields

---

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

Basic event fields are:

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

## device and ipAddress Fields

The device field is a free-form text field that allows up to 255 characters. Zenoss Core accepts any value for this field. If the device field contains an IP address, then the system queries for devices with a matching address. If it finds a match, it changes the device field to the found device identifier.

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

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

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

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

## eventState Field

The eventState field defines the current state of an event. This field is often updated after an event has been created. Values for this numeric field are 0-6, defined as follows:

| Number | Name         | Description  |
|--------|--------------|--|
| 0      | New          |  |
| 1      | Acknowledged |  |
| 2      | Suppressed   |  |
| 3      | Closed       | State given to an event that was closed as the result of a user action.  |
| 4      | Cleared      | State given to an event that was cleared by a corresponding clear event.   |
| 5      | Dropped      | State given to an event that was dropped via an event transform. These events are never persisted by the system.               |
| 6      | Aged         | State given to an event that was automatically closed by the system according to the severity and last seen time of the event. |

## severity Field

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

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

## summary and message Fields

The summary and message fields are free-form text fields. The summary field allows up to 255 characters. The message field allows up to 4096 characters. These fields usually contain similar data.

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

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

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

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

## Other Fields

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

The following table lists additional event fields.

| Field         | Description  |
|---------------|--|
| dedupid       | Dynamically generated fingerprint that allows the system to perform de-duplication on repeating events that share similar characteristics.   |
| component     | Free-form text field (maximum 255 characters) that allows additional context to be given to events (for example, the interface name for an interface threshold event).                                       |
| eventClass    | Name of the event class into which this event has been created or mapped.  |
| eventKey      | Free-form text field (maximum 128 characters) that allows another specificity key to be used to drive the de-duplication and auto-clearing correlation process.  |
| eventClassKey | Free-form text field (maximum 128 characters) that is used as the first step in mapping an unknown event into an event class.  |
| eventGroup    | Free-form text field (maximum 64 characters) that can be used to group similar types of events. This is primarily an extension point for customization. Currently not used in a standard system.             |
| stateChange   | Last time that any information about the event changed.  |
| firstTime     | First time that the event occurred.  |
| lastTime      | Most recent time that the event occurred.  |
| count         | Number of occurrences of the event between the firstTime and lastTime.   |
| prodState     | Production state of the device, updated when an event occurs. This value is not changed when a device's production state is changed; it always reflects the state when the event was received by the system. |
| agent         | Typically the name of the daemon that generated the event. For example, an SNMP threshold event will have zenperfsnmp as its agent.  |
| DeviceClass   | Device class of the device that the event is related to.   |
| Location      | Location of the device that the event is related to.   |
| Systems       | Pipe-delimited list of systems that the device is contained within.  |
| DeviceGroups  | Pipe-delimited list of systems that the device is contained within.  |
| facility      | Only present on events coming from syslog. The syslog facility.  |
| priority      | Only present on events coming from syslog. The syslog priority.  |
| nteventid     | Only present on events coming from Windows event log. The NT Event ID.   |
| ownerid       | Name of the user who acknowledged this event.  |

| Field             | Description  |
|-------------------|--|
| clearid           | Only present on events in the archive that were auto-cleared. The evid of the event that cleared this one.       |
| DevicePriority    | Priority of the device that the event is related to.   |
| eventClassMapping | If this event was matched by one of the configured event class mappings, contains the name of that mapping rule. |
| monitor           | In a distributed setup, contains the name of the collector from which the event originated.                      |

## Details

In addition to the standard fields, the system also allows events to add an arbitrary number of additional name/value pairs to events to give them more context.

## De-Duplication

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

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

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

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

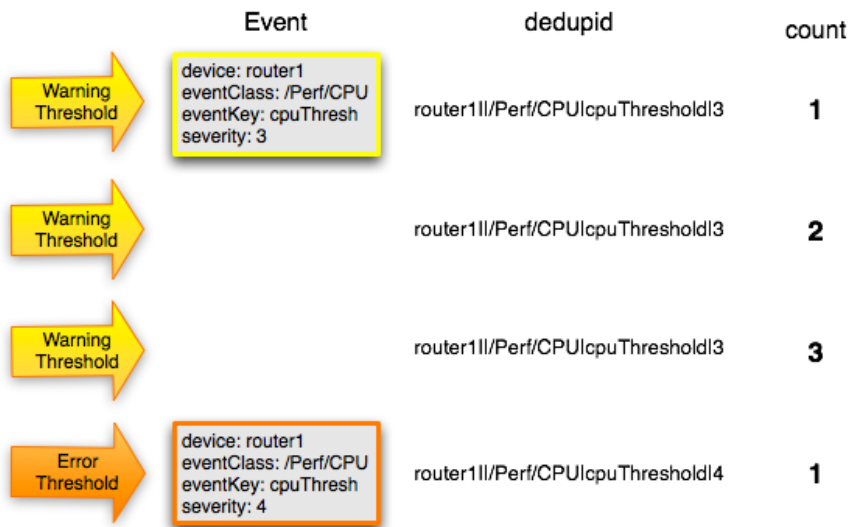
```
www.example.com | | /Status/Web | | 4 | WebTx check failed
```

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

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

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

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

**Figure 69: Event De-Duplication**

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

## Auto-Clear Correlation

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

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

If a component has been identified for the event, then the auto-clear fingerprint consists of these fields:

- If component UUID exists:
  - component UUID
  - eventClass (including zEventClearClasses from event class configuration properties)
  - eventKey (can be blank)
- If component UUID does not exist:
  - device
  - component (can be blank)
  - eventKey (can be blank)
  - eventClass (including zEventClearClasses from event class configuration properties)

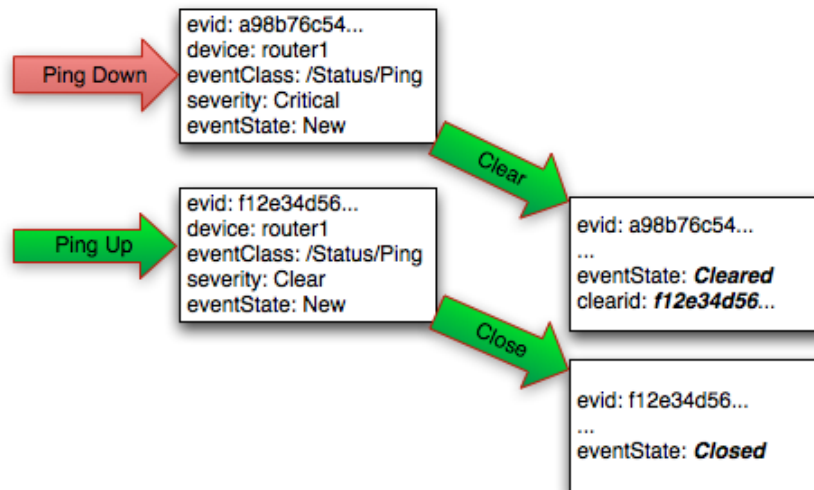
When a new event comes into the system with a special 0 (Clear) severity, Zenoss Core checks all active events to see if they match the auto-clear fingerprint of the new event. All active events that match the auto-clear fingerprint are updated with a Cleared state, and the clearid field is set to the UUID of the clear event. After a configurable period of time, all events in a closed state (Closed, Cleared, and Aged) are moved from the active events table to the event archive.



If an event is cleared by the clear event, it is also inserted into the active events table with a status of Closed; otherwise, it is dropped. This is done to prevent extraneous clear messages from filling your events database.

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

**Figure 70: Event Auto-Clear**



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

---

**Note** Avoid making clear events too generic; otherwise, you may inadvertently clear a wider range of events than you intend.

---

## Event Consoles

---

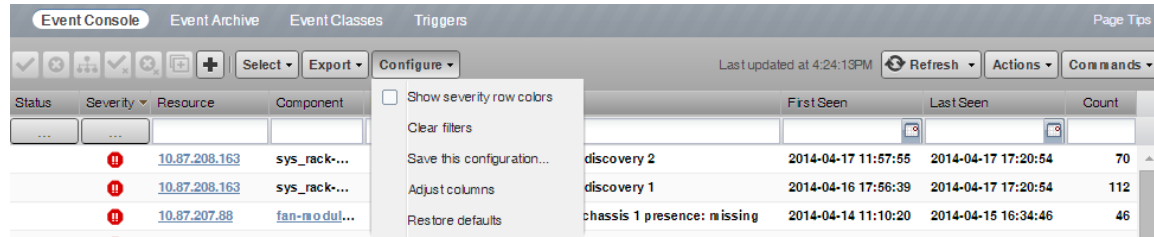
Zenoss Core features multiple event consoles that allow you to view and manage events. Each console shows different events subsets, depending on your current context.

Event consoles are:

- **Master-** To access this console, click Events on the Navigation menu. You can view all events from this console.
- **Contextual-** Contextual event consoles are found throughout the system. Each time you see an Events selection for a device, device organizer, component, or event class, you can view event information that has been automatically filtered to show events specific to the current context.

## Master Event Console

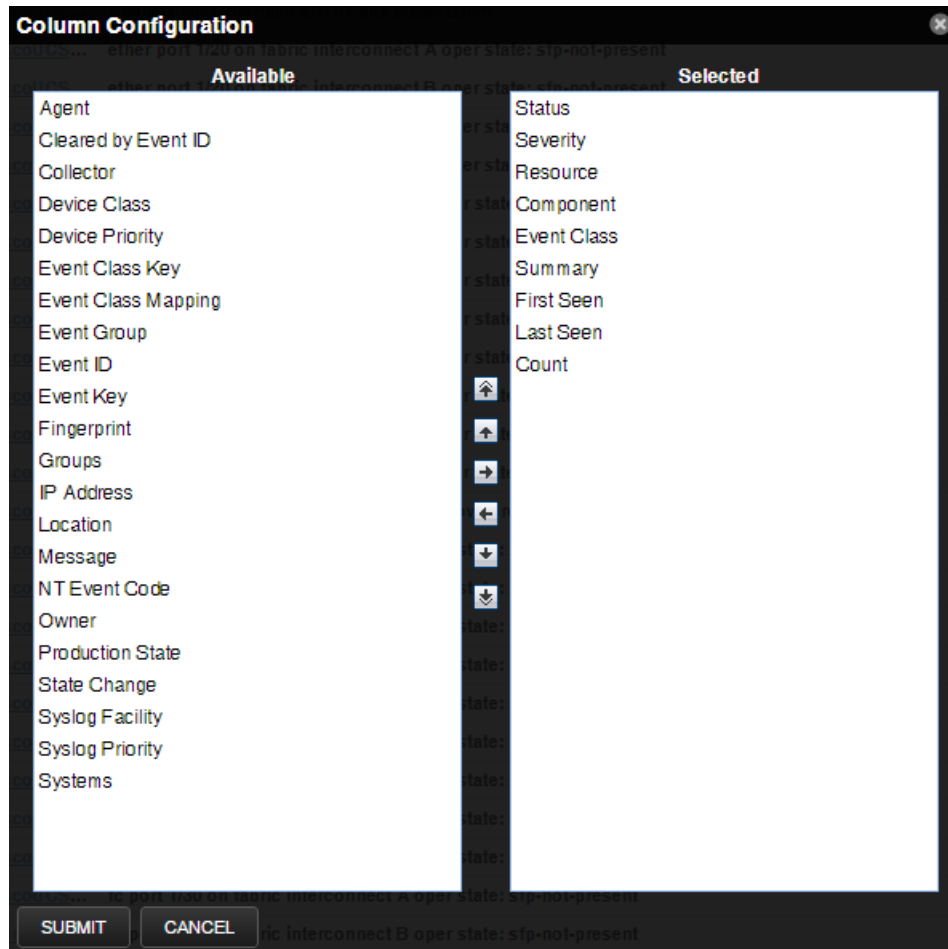
The master event console is the system's central nervous system, enabling you to view and manage events. It displays the repository of all events that have been collected by the system.

**Figure 71: Event Console**

## Customizing the Event Console

You can add or delete data columns to customize your event console view. To configure the columns to be displayed on the Event Console:

- 1 Navigate to the Event Console (**Events > Event Console**).
- 2 Click the **Configure** button and select **Adjust columns** from the drop-down list. The Column Configuration window appears.

**Figure 72: Event Console Column Configuration**

- 3 Move the names of the columns from the Available to the Selected column to display them. You can order the Selected column by using the arrow keys. The order of the selected column names determines the left-to-right display on the Event Console.
- 4 Click **Submit**.

## Selecting Events

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

- Click a row to select a single event
- Ctrl-Click rows to select multiple events, or Shift-Click to select a range of events

## Sorting and Filtering Events

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

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

Filter options appear below each column header. A match value can be any full string or a subset of a string, optionally with the wildcard (\*) contained in the values in that column. You can also use " | " (OR), or " ! " (NOT) expressions to further target your filters. For example, typing ! ! windows in the Event Class filter will return all the non-Windows device events.

**Figure 73: Event Console Filter Options**

| Status   | Severity | Resource     | Component    | Event Class | Summary                                       | First Seen          | Last Seen           | Count |
|--|----------|--------------|--------------|-------------|---|---------------------|---------------------|-------|
| <input checked="" type="checkbox"/> New          |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-7 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
| <input checked="" type="checkbox"/> Acknowledged |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-8 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
| <input type="checkbox"/> Suppressed              |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-1 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
| <input type="checkbox"/> Closed                  |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-5 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
| <input type="checkbox"/> Cleared                 |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-4 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
| <input type="checkbox"/> Aged                    |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-3 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |
|  |          | 10.87.207.88 | syschassi... | [Status]    | Fan module 1-2 in chassis 1 presence: missing | 2014-04-09 14:07:04 | 2014-04-09 14:07:04 | 1     |

You can filter the events that appear in the list in several ways, depending on the field type:

- **Resource** - Enter a match value to limit the list.
- **Component** - Enter a match value to limit the list.
- **Event Class** - Enter a match value to limit the list.
- **Summary** - Enter a match value to limit the list.
- **First Seen** - Enter a value or use a date selection tool to limit the list.
- **Last Seen** - Enter a value or use a date selection tool to limit the list.
- **Count** - Enter a value to filter the list, as follows:
  - *N* - Displays events with a count equal to *N*.
  - *:N* - Displays events with a count less than or equal to *N*.
  - *M:N* - Displays events with a count between *M* and *N* (inclusive).
  - *M:-* - Displays events with a count greater than or equal to *M*.

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

You also can re-arrange the display order of columns in the event console. Click-and-drag column headers to change their display.

## Working with Live Search

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

- **Device** (name) - Device name searches:

- Are case-insensitive.
- Are tokenized on whitespace (meaning that any searches that span whitespace and do not start with a complete token will return no results).
- If quoted, return only exact matches.
- **Component-** Component searches:
  - Are case-insensitive.
  - Are tokenized on whitespace (meaning that any searches that span whitespace and do not start with a complete token will return no results).
  - If quoted, return only exact matches.
- **Summary-** Summary searches:
  - Are case-insensitive.
  - Are tokenized on whitespace (meaning that any searches that span whitespace and do not start with a complete token will return no results).
- **Event class-** Event class searches:
  - Are case-insensitive.
  - Are tokenized on / (slash). If the search begins with a slash, and ends with a slash or asterisk, then event classes are searched by using a "starts with" approach. If a search starts with a slash and ends with any other character, then event classes are searched by using an exact match for the event class. If a search does not begin with a slash, then event classes are searched by using a sub-string match on each event class.
- **IP Address-** IP address searches (for IPv4 and IPv6 values):
  - Are tokenized by . (period) and : (colon). For example, the following searches would return a result of 129.168.1.100:
    - 168
    - 168.1
    - 129.16\*
    - \*29
- **First Seen, Last Seen, State Change-** This field is not tokenized; date searches are converted to numeric representations, and then ranges using these representations are created. Search values are inclusive. Searches on date fields will search from the value entered. Any results that match the value or any value in the future are returned.

The following searches would return the First Seen time of 2014-05-04 15:52:52:

- First Seen: 2014-05-01 00:00:00
- First Seen: 2014-05-04 15:52:52

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

## Saving an Event Console View

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

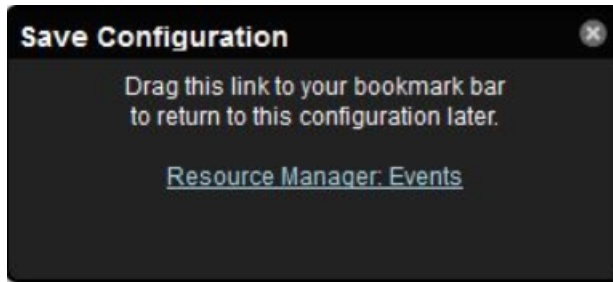
- 1 Select **Configure > Save this configuration**.

A dialog containing a link to the current view appears.

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

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

**Figure 74: Saving a Custom View (Bookmark)**




---

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

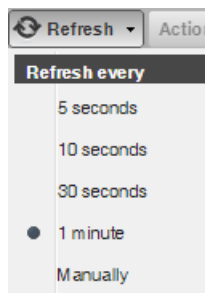
---

## Refreshing the View

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

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

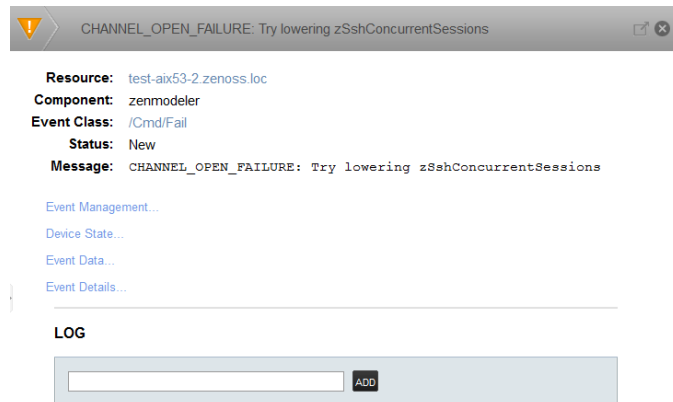
**Figure 75: Automatic Refresh Selections**



## Viewing Event Details

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

The Event Details area appears.

**Figure 76: Event Details**

To see more information about the event, click the Event Management, Device State, Event Data, or Event Details link. To display the event information in a new window, click the icon located at the top right.

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

### Acknowledging Events

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

- 1 Select one or more events in the event console view.
- 2 Click the **Acknowledge Events** icon.

A check mark appears for each acknowledged event.

### Returning Events to New Status

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

- 1 Select one or more events in the event console view.
- 2 Click the **Unacknowledge Events** icon.

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

### Classifying Events

Classifying events lets you associate events shown as **/Unknown** with a specific event class. To classify an unknown event, an event class key must be specified for the event.

To classify events:

- 1 Select one or more **/Unknown** events in the event console view.
- 2 Click the **Reclassify an Event** icon.

The Classify Events dialog appears.

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

---

**Note** You can also classify events from the event archive.

---

## Closing Events

When you no longer want to actively monitor an event (after you acknowledge it, for example), you can specify to close the event and move it to the event archive according to a configured event archive interval. To do this:

- 1 Select one or more events in the event console view.
- 2 Click the **Close Events** icon.

The selected events are closed and moved to the archive at the specified interval.

To view events in the event archive, select **Events > Event Archive**.

---

**Note** Users with no assigned role can view all events in the archive.

---

## Reopening Events

You can reopen events in the active event console that are in the Closed, Cleared, or Aged state.

To reopen events:

- 1 Select one or more Closed, Cleared, or Aged events.
- 2 Click the **Reopen Events** icon.

The selected events are returned to active status.

---

**Note** You cannot re-open a closed event if another active event with the same fingerprint exists. Before you can re-open the closed event, you must close the new event.

---

## Exporting Event Data

You can export data from the event console to a comma-separated value (.csv) or XML file. You can select individual events (to export only those events), or make no selections (to export all events that match the current filter criteria).

To export events:

- 1 Optionally select one or more events.
- 2 Select **Export > CSV** or **Export > XML**. By default, the exported file is named `events. Extension`.

## Creating Events

To create events from the event console, click the **Add an Event** icon.

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

## Event Sources

---

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

### Generated Events

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

- **zenping** - Ping up/down events

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

## Captured Events

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

These standard daemons are responsible for collecting captured events:

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

Any ZenPacks you install may optionally include their own daemons.

## Creating Events Manually

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

## Creating Events through the User Interface

To create events manually through the user interface:

- 1 Navigate to Events, then click the **Add an Event** icon.

The Create Event dialog appears.

**Figure 77: Create Event Dialog**

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

## Creating Events from the Command Line

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

```
zensendevent
  Options summary
```



Common options include:

- -d DEVICE, --device=DEVICE
- -i IPADDRESS, --ipAddress=IPADDRESS
- -y EVENTKEY, --eventkey=EVENTKEY
- -p COMPONENT, --component=COMPONENT
- -k EVENTCLASSKEY, --eventclasskey=EVENTCLASSKEY
- -o OPTIONALFIELD, --optional\_field x=OPTIONALFIELD
- -s SEVERITY, --severity=SEVERITY
- -c EVENTCLASS, --eventclass=EVENTCLASS

### Example

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

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

## Event Classes

---

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

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

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

## Event Class Configuration Properties

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

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

- **zEventAction**- How and where affected events are stored when they occur.
  - **status**- Active events table with original event state
  - **history**- Active events table with closed event state
  - **drop**- Events are not stored

- **zEventClearClasses**- Optional list of event class names whose active events will be cleared by clear events occurring in this class.
- **zEventSeverity**- The severity of affected events is changed to this value unless the Default value (-1) is used.
- **zFlappingIntervalSeconds**- Defines the time interval to check for event flapping (changing severity level repeatedly). Default value is 3600 seconds.
- **zFlappingSeverity**- Define the severity to check for event flapping. If the severity level on an event changes from this value a certain number of times (zFlappingThreshold) within a certain time range (zFlappingIntervalSeconds) then an event flapping event is generated. Possible values include: 5-Critical, 4-Error, 3-Warning, 2-Info, 1-Debug, and 0-Clear.
- **zFlappingThreshold**- Number of times an event severity must flap within an interval. One of the parameters to define in order to generate event flapping events.

A good example of how the system uses the event class configuration properties is found in the /Status event class. Within the /Status event class' configuration properties, zEventAction is set to "history" and zEventSeverity is set to "Default". This means that events sent with this event class are sent into the active events table with an initial state of closed, and the event severity unchanged.

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

## Mapping and Transformation

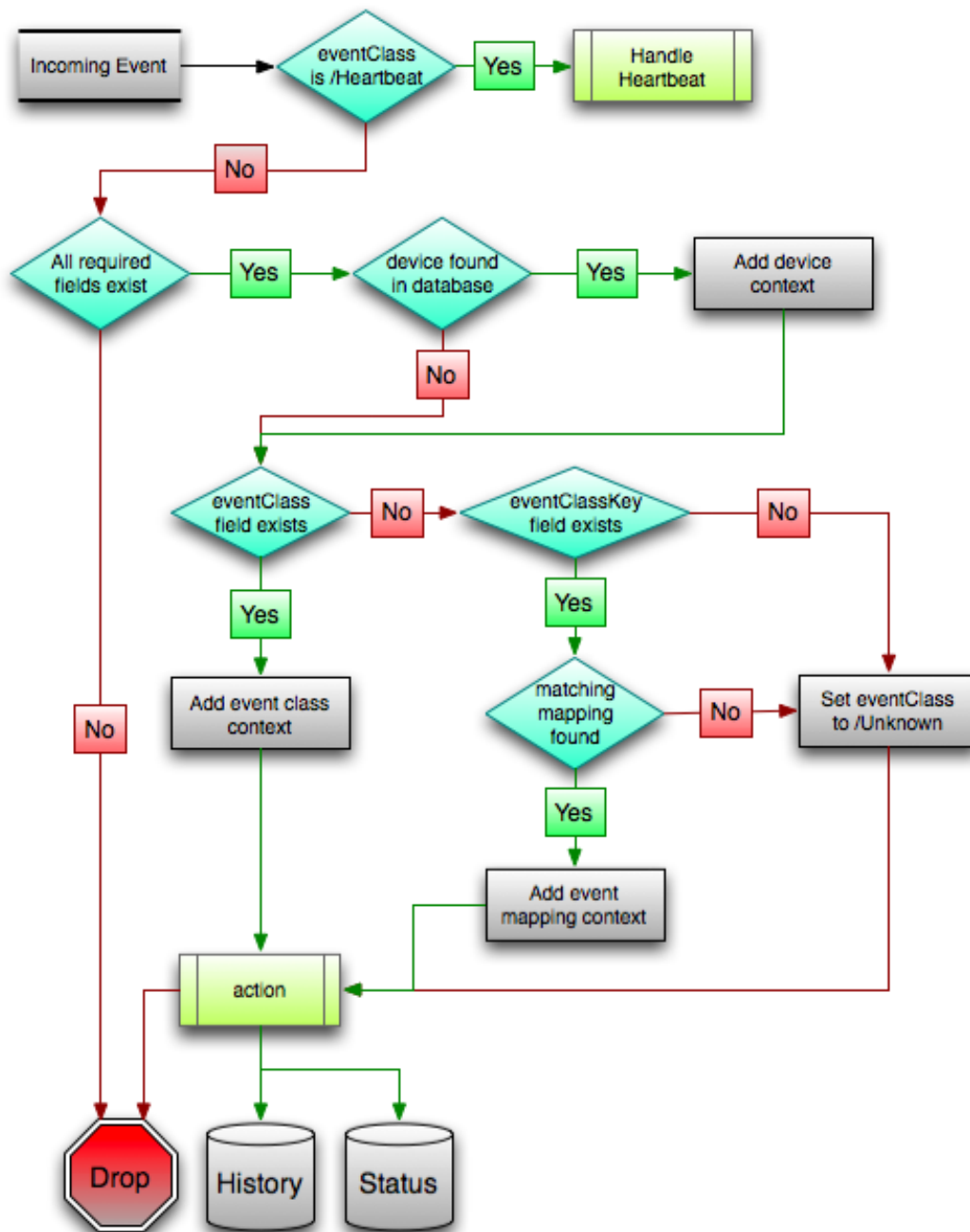
---

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

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

- evid
- firstTime
- lastTime
- count

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

**Figure 78: Event Processing**

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

## Event Class Mappings

To view event class mappings, select **Events > Event Classes**, and then select **Mapping Instances** in the drop-down list. This allows you to see all event class mappings in a single location. The ID column shows the mapping's event class.

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

- 1 Select an event that you want to match in the event console.
- 2 Click the **Reclassify an Event** icon.

The Classify Events dialog appears.

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

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

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

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

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

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

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

## Event Class Mapping Sequence

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

For example, suppose a router is sending in unclassified events that need to be mapped to two event classes:

- /Events/Router/fanDown
- /Events/Router/fanUnknown

The event class key for both has been sent to "router", but one has a message of "Fan Down" and the other has no message at all. The mapping on /Events/Router/fanDown has an event class key of "router" and a regex of "Fan Down." The mapping on /Events/Router/fanUnknown has only an event class key of "router" and (in this example) no regex. Because the fanUnknown mapping matches the fanDown events, the evaluation of fanDown needs to occur first.

You can modify the evaluation of mappings with the same event class key in the Sequence area of any of those event class mappings. In the previous example, you could go to either mapping, select Sequence, and both mappings would be displayed. You can set one to 0, and the other to 1. (You can enter other values, but they will be changed to the shortest list of integers, starting with 0.) Setting fanDown to 0 and fanUnknown to 1 will ensure that the events will be mapped properly.

## Event Class Transform

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

To access the transform for an event class:

- 1 Navigate to the event class from **Events > Event Classes**.
- 2 From the drop-down list, select **Transforms**.
- 3 Enter information into the dialog (as Python code), and then click the **Save** button in the upper-right corner. As you develop your transform, you can revert back to the last saved state by clicking the **Revert this Transform** button.

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

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

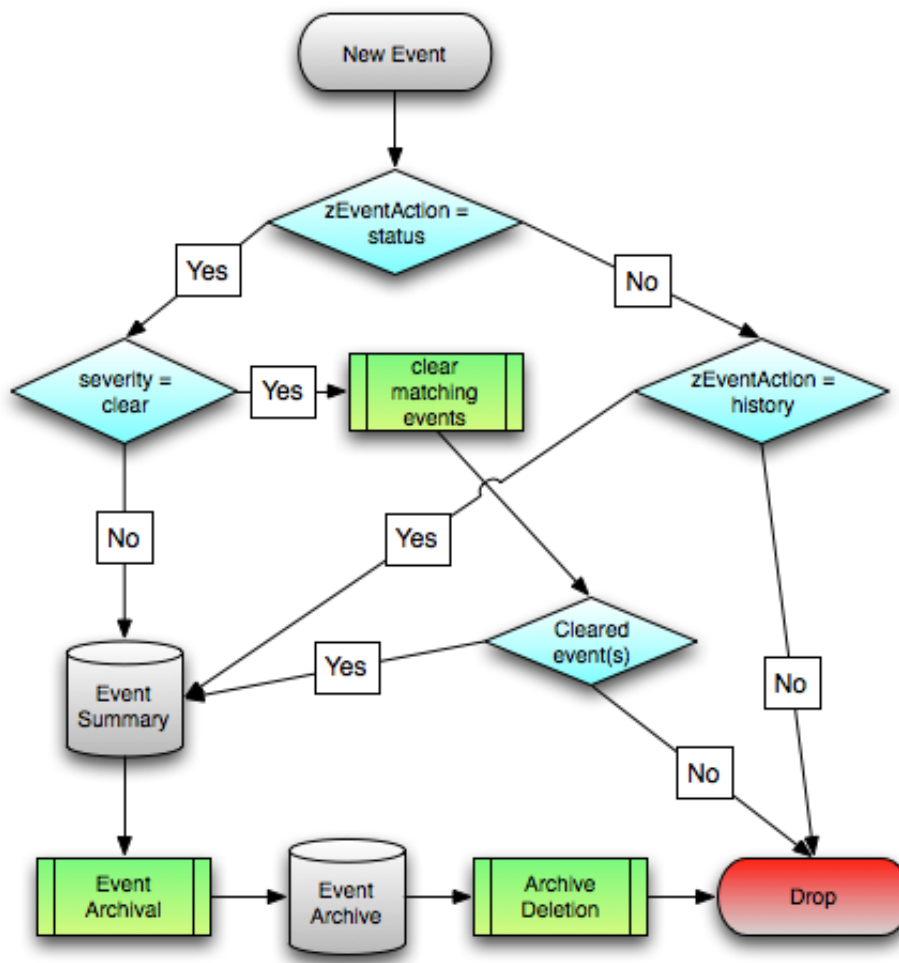
```
if device and "Hawaii" in device.getLocationName(): evt._action = "drop"
```

## Event Life Cycle

---

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

The following illustration depicts the event life cycle.

**Figure 79: Event Life Cycle**

## Automatic Event Aging

From the Event Configuration page (**Advanced > Settings > Events**), you can set up automatic aging of events. Aging of events will automatically update active events that match the severity and aging threshold to a status of Aged. After the configured event archive interval, all Closed, Aged, and Cleared events are moved to the event archive.

Properties that control this behavior are:

- **Don't Age This Severity and Above** - Options are Age All Events, Critical, Error, Warning, Info, Debug, and Clear. By default, this value is set to Error, meaning that all events with a status of Error or Critical are not aged.
- **Event Aging Threshold (minutes)** - Set the time value, in minutes, that an event must reach before it is aged. By default, this is 240 minutes.
- **Event Aging Interval (milliseconds)** - The interval when events are scanned to perform autoaging. By default, this is 60000 milliseconds (60 sec).
- **Event Aging Limit** - The maximum number of events to age in each interval. The limit should be kept relatively low to prevent large database transactions. By default, this is 1000 events.
- **Event Archive Threshold (minutes)** - Specify the number of minutes since a closed event was last seen before it is moved to the event archive. The minimum value is 1; the maximum value is 43200.
- **Event Archive Interval (milliseconds)** - The interval when events are scanned for moving to the archive. By default, this is 60000 milliseconds (60 sec).

- **Event Archive Limit** - The maximum number of events to archive in each interval. The limit should be kept relatively low to prevent large database transactions. By default, this is 1000 events.
- **Delete Archived Events Older Than (days)** - The number of days that events in the event archive are saved. By default, they are kept in the archive for 90 days. The minimum value is 1 and the maximum value is determined by the range of event archive partitions. With the default configuration, the maximum value is 1000 days.
- **Default Syslog Priority** -
- **Default Availability Report (days)** -
- **Max Event Size in Bytes** - The maximum size of an event that will be processed in bytes. Events that are too large will be logged and dropped. Events that will become too big will have their details overwritten with new details. By default, this is 32768 bytes.
- **Summary Index Interval (milliseconds)** - The default indexing interval of the event summary in milliseconds. By default, this is 1000 milliseconds (1 sec).
- **Archive Index Interval (milliseconds)** - The default indexing interval of the event archive in milliseconds. By default, this is 30000 milliseconds (30 sec).
- **Index Limit** - The number of events to index in each index interval. By default, this is 1000 events.
- **Event Time Purge Interval (days)** - The number of days that event occurrence time are kept. By default, they are kept for 7 days. The minimum value is 1 and the maximum value is determined by the range of event time partitions. With the default configuration, the maximum value is 7 days.
- **Enable Event Flapping Detection** - Select this check box if you wish to enable event flapping detection. If an event is created and then cleared *flapping\_threshold* times in *event\_flapping\_interval* time then an event of event flapping event class is created.
- **Event Flapping Event Class** - The event class under which generated flapping events belong.
- **Clear Event Heartbeats** - Click **Clear** to clear the event heartbeats.

## Automatic Archived Event Cleanup

You can set up automatic purging of events from the event archive from the Event Configuration page (**Advanced > Settings > Events**). When events are purged, they can be recovered only from backups.

The property that controls this behavior is Delete Archived Events Older Than (days). Acceptable values are between 1 and 1000 (days).

## Capturing Email Messages as Events

---

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

### 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 Core server explicitly by using the 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 Core 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).

---

**Note** In order to execute a command using \$ZENHOME (/opt/zenoss for the zenoss user), you must be attached to the container holding the Zenoss Core master host.

---

## 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.

## Translating Message Elements to the Event

Zenoss Core translates various message elements to the event, as follows:

- **FROM Field**- If the FROM field is an IP address, then the system associates the event with the device with the same IP address. If the FROM field is a fully qualified domain name, then the system resolves it to an IP address, and then performs the device association using the resolved IP address. The resolution of hostname uses "A" records rather than "MX" records.
- **TO Field**- The system ignores the TO field in the email message. ZenMail accepts email to any user and domain name combination. ZenPop also drops the TO field, and uses only the FROM field.
- **SUBJECT Field**- ZenMail and ZenPop use the SUBJECT as the event summary.
- **Message Body**- ZenMail and ZenPop use the first mime attachment as the event details. The system ignores secondary message bodies (typically HTML-encoded versions of the message). It also ignores attachments (such as files).

## SNMP Traps and Event Transforms

---

An SNMP trap is a message that is initiated by a network element and sent to the network management system. Often, traps indicate a failure of some sort, such as a router message indicating a power supply failure, or a printer message indicating an "out-of-ink" condition.

If an SNMP trap enters the system, and Zenoss Core cannot identify the event (the event is classified as "/Unknown"), then you can classify the event so that the system handles it consistently.

## Classifying SNMP Traps

To classify an SNMP trap event:

- 1 From the Event Console, select the unknown event or events.
- 2 Click the **Reclassify an event** icon.

The Classify Events dialog appears.

- 3 Select /App, and then click **Submit**.



To edit this classification:

- 1 From the Navigation area, select **Events > Event Classes**.
- 2 Ensure Mapping Instances is displayed.
- 3 Select the event map you created.
- 4 In the left panel, select **Edit** from the **Action** icon.

The edit page appears. This page contains rules used to map the event to the /App category. This rule, since it matches the trap by a specific OID, is all that is needed.

In the Transform area, you can enter code to modify the summary. For example, if you want to set the summary string to "Spam Filter Detects Virus," then you can enter:

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

A trap has a header with some standard information, followed by a sequence of attribute/values.

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

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

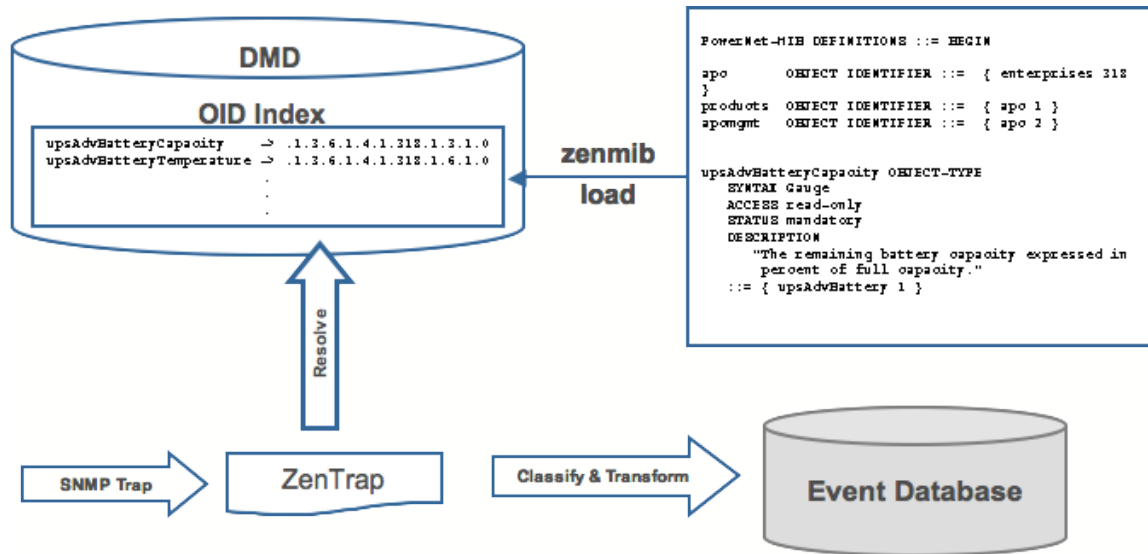
However, the OID and the data is still in there. Instead, use the slightly more cryptic:

```
evt.summary = getattr(evt, ".1.3.6.1.4.1.9789.1500.2.5", "Unexpected  
missing OID")
```

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

```
evt.summary = getattr(evt, ".1.3.6.1.4.1.9789.1500.2.5", "Unexpected  
missing OID") \ + " from device " + device.getId()
```

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

**Figure 80: 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
```

## 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 Austin"
```

- 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 Austin
```

- 7 Send this event to the event archive.

Now, load some MIBs into the system so that this OID is translated into a better format:

- 1 Copy the demonstration MIB into `$ZENHOME/share/mibs/site`.

---

**Note** In order to execute a command using `$ZENHOME` (`/opt/zenoss` for the `zenoss` user), you must be attached to the container holding the Zenoss Core master host.

---

- 2 Run `zenmib` to load it:

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

- 3 The MIB loaded, but is missing some other definitions. Copy them:

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

- 4 Run `zenmib` again and load the definitions into the system:

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

- 5 Restart the `zentrap` daemon to retrieve the new MIB information:

```
$ zentrap restart
```

- 6 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 Austin"
```

- 7 Check the event. Make sure the count is 1. If the count is 2, send the event to the event archive and send the trap again. Look at the Details tab. Now you should see something like this:

```
sysLocation Device in Austin
```

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
```

## Transforming Events with Event Mappings

To modify events as they arrive, create an event map through the user interface:

- 1 Create an event class.
- 2 Go to the event console and create an event mapping in this class from the existing event.
- 3 Edit the map.
- 4 In the Transform area, update the event with detail data. The entry field allows you to insert Python scripts. The event is provided as `"evt"` and the device as `"dev."`

In this case, extract the `sysLocation` event detail and make it the summary with:

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

5 Save the event mapping.

If you move the event to the event archive and resend the trap, the summary for the trap should now read the device name in the location you assigned.

If you encounter problems with the transform, check the `zentrap.log` file for errors that occurred.

## Event Transforms Based on Event Class

When an event arrives in the system, you can change values (such as severity). For example, you can make the summary more informative, or change severity according to text within the summary.

Each event class allows for a short Python script to be executed when an event arrives.

### Example

A user may want full file system threshold events on `/data` to be critical. Add the following Python script in the Threshold Transform of `/Events/Perf/Filesystem`:

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

Like event mappings for event class keys, "evt" and "dev" objects are available in the script of the transform.

# Production States and Maintenance Windows

# 9

*Production state* determines the level of monitoring and alerting applied to an individual device. Typically, alerting rules specify that the system will monitor and create events for devices that are in the "Production" production state. *Maintenance windows* are planned time periods used to temporarily modify alerting rules so that event-generated alerts are temporarily halted during the window.

## Production States

Production state determines whether a device is monitored, and can be used to control several elements of the event system, such as whether an event will produce a remote alert (email or page).

Choose a production state for a device based on whether you want:

- The device to be monitored
- The device to appear on the dashboard
- Alerting to occur

The following table lists production states and their characteristics.

| Production State | Devices Monitored? | Appear on Dashboard? |
|------------------|--------------------|----------------------|
| Production       | yes                | yes                  |
| Pre-Production   | yes                | no                   |
| Test             | yes                | no                   |
| Maintenance      | yes                | may appear           |
| Decommissioned   | no                 | no                   |

When you add a device to the system, its default state is Production. You may want to add triggers and notifications to alert you to various conditions that occur in the system, such as production state changes or a severity level being reached. For example, you can set up a trigger when a device is in either a production or a maintenance state and has a severity of Error or higher. You can then notify users when this trigger condition is met. For more information, see [Working with Triggers and Notifications](#) on page 19.

## Setting the Production State for Devices

To set the production state for a device:

- 1 Click a device name in the list of devices. The Device Overview page appears.

- 2 Select a production state from the list of options, and then click **Save**.

To set the production state for a group of devices:

- 1 Select a category of devices (by class, group, system, or location) from the hierarchy.
- 2 From the list of options in the Production State column, select a production state.

The newly selected state is applied to all devices that appear in the list.

**Figure 81: Select Production State (Multiple Devices)**

| Device               | IP Address    | Device Class              | Production State | Events |
|----------------------|---------------|---------------------------|------------------|--------|
| 10.87.208.11         | 10.87.208.11  | /CiscoUCS                 | ...              |        |
| ACC2                 | 10.181.100.91 | /Network/Cisco/Nexus/5000 |                  |        |
| aus-ucs2             | 10.87.207.89  | /CiscoUCS                 |                  |        |
| aus-ucs7.zenoss.lbc  | 10.87.207.94  | /CiscoUCS                 |                  |        |
| Cisco_10.181.100.14  | 10.181.100.14 | /Network/Cisco/6500       |                  |        |
| CiscoUCS_BladeServer |               | /vSphere                  |                  |        |
| FABa                 | 10.181.100.86 | /Network/Cisco/MDS/9000   | Production       | 38     |

## Maintenance Windows

Maintenance windows allow scheduled production state changes of a device or all devices in a system, group, or location. You might want to set up a maintenance window, for example, to prevent alerts and warnings while you perform configuration changes or reboot a device.

**Note** In lieu of setting up a maintenance window, you can change the production state for a device manually at the time you want to make changes.

When the maintenance window starts, the production state of the device is set to the value of Start Production State (Maintenance). When the maintenance window closes, the production state of the device reverts to the value of Stop Production State (the state the device was in prior to maintenance).

## Maintenance Window Events

When a maintenance window starts, an event is created with the following information:

- depuid - zenactions | *Monitor* | *MaintenanceWindowName* | *TargetOrganizerOrDevice*
- prodState - *StartProductionState*
- severity - Info
- summary/message - Maintenance window starting *MaintenanceWindow* for *Target*
- eventClass - /Status/Update
- eventClassKey - mw\_change
- maintenance\_devices - *Target*
- maintenance\_window - *MaintenanceWindow*

When a maintenance window stops, an event is created with the following information:

- severity - Clear
- summary/message - Maintenance window stopping *MaintenanceWindow* for *Target*

- prodState - -99 (meaning "unknown.")

Maintenance window events auto-clear, meaning that stop events clear start events.

## Creating and Using Maintenance Windows

You can create a maintenance window for an individual device or group of devices in the devices hierarchy.

### Create a Maintenance Window for a Single Device

To create a maintenance window for a device:

- 1 Click a device name in the devices list.

The device Overview page appears.

- 2 In the left panel, select **Device Administration**.
- 3 In the Maintenance Windows area, select the **Set up a new Maintenance Window** icon.

The Add New Maintenance Window appears.

- 4 Define the attributes for the maintenance window:
  - **Name** - Name of the maintenance window.
  - **Enabled** - Click the check box to make the maintenance window active, or clear it to de-activate it.
  - **Start Date and Time** - Specify a date and time for the window to become active. The time should be specified in 24-hour format and in UTC (Universal Coordinated Time), not local time.
  - **Duration in Days: Hrs: Mins:** - Specify the length of time for the window to be in effect.
  - **Repeat** - Specify how often to repeat the maintenance window. Default value is Never.
  - **Window Production State** - Define the production state for the window when the maintenance period is in effect. Default value is Maintenance. When the maintenance window is over, the device will return to the state it was in just before the maintenance window started.
- 5 Click **Submit**.

### Create a Maintenance Window for a Group of Devices

To create a maintenance window for a group of devices:

- 1 Select a group of devices from the devices hierarchy or devices list and click **Details** in the left column.
- 2 In the left panel, select **Device Administration**.
- 3 In the Maintenance Windows area, select the **Set up a new Maintenance Window** icon.

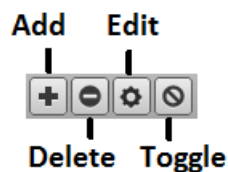
The Add New Maintenance Window appears.

- 4 Define the attributes for the maintenance window:
  - **Name** - Name of the maintenance window.
  - **Enabled** - Click the check box to make the maintenance window active, or clear it to de-activate it.
  - **Start Date and Time** - Specify a date and time for the window to become active. The time should be specified in 24-hour format and in UTC (Universal Coordinated Time), not local time.
  - **Duration in Days: Hrs: Mins:** - Specify the length of time for the window to be in effect.
  - **Repeat** - Specify how often to repeat the maintenance window. Default value is Never.
  - **Window Production State** - Define the production state for the window when the maintenance period is in effect. Default value is Maintenance. When the maintenance window is over, the device will return to the state it was in just before the maintenance window started.
- 5 Click **Submit**.

## Managing Maintenance Windows

Once you have created maintenance windows for your devices or groups of devices, you can quickly manage these instances on the Maintenance Windows screen.

- 1 Navigate to the Maintenance Window screen. This is the same place where you initially created the maintenance window (Device Administration link on device overview page). On this screen you can perform any of the following by clicking the appropriate icon:



- Add a new maintenance window
  - Delete the selected maintenance window
  - Edit the selected maintenance window (can also double-click a maintenance window)
  - Toggle the selected maintenance window from enabled to disabled and vice-versa. The Enabled column will switch values.
- 2 Ensure that your changes are reflected in the Maintenance Window screen.



## 10

# Organizers and Path Navigation

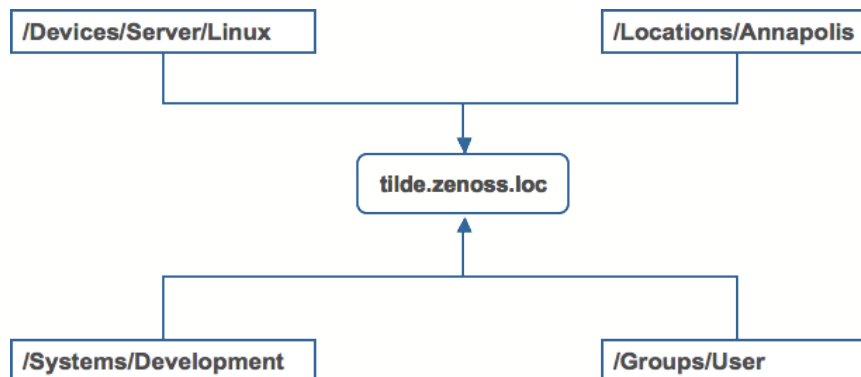
---

You can group system objects, including devices, sub-systems, configuration properties, and templates. A device, for example, can belong to multiple classifications, including:

- Device classes
- Groups
- Systems
- Locations

In the following illustration, the device `tilde.zenoss.loc` belongs to five different classifications. Configuration properties and monitoring settings for each of these groups are applied to this device.

**Figure 82: Device Groupings**



## Classes

---

The most important organizers are *classes*, which comprise:

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

Templates and configuration properties can be inherited based on class. These attributes can be overwritten further down the class hierarchy, all the way down to the individual component level. The class hierarchy includes all defined and standard classes and sub-classes.

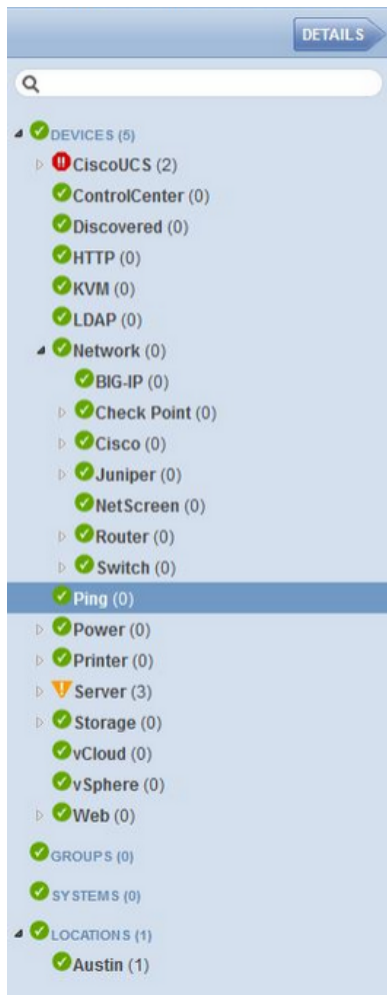
The following procedures are illustrated using device classes and sub-classes, but the same concepts apply to event classes, service classes, and product classes. When you add a device to the system, you should (after providing the network name or IP address) specify its device class. Templates and configuration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

## Viewing Device Classes

To view device classes and the devices they contain, select Infrastructure from the Navigation menu.

The device list appears. The top of the devices hierarchy lists device classes. Click a class name to view devices in that class, or expand it to show sub-classes.

**Figure 83:** Devices Hierarchy



An indicator appears next to each listed class to show whether there are events associated with any devices in that class.

## Adding Classes

To create a device class:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select the parent location in the device class hierarchy where you want to add a new child class.
- 3 Click the **Add** icon.

The Add Device Class dialog appears.

- 4 Enter a name and description for the new device class, and then click Submit.

The new device class appears in the hierarchy under the parent device class. You can now move devices into this class. Select one or more devices in the device list, and then drag them to the new class.

## Moving a Class

To move a class in the hierarchy:

- 1 Select the class in the hierarchy.
- 2 Drag the class to its new location.

The Move Organizer confirmation dialog appears.

- 3 Click **OK** to confirm the action.

The class appears at its new location in the hierarchy.

## Setting Configuration Properties at the Class Level

To set configuration properties at the device class level:

- 1 Select a device class in the devices hierarchy.
- 2 Click Details, then click Configuration Properties.

The Configuration Properties page for the selected device class appears.

**Figure 84: Device Class Configuration Properties**

| Discovered <span>SEE ALL</span> |                  | Delete Local Copy           |                    |      |
|---------------------------------|------------------|-----------------------------|--------------------|------|
| Is Local                        | Category         | Name                        | Value              | Path |
|                                 | Cisco            | zCiscoACEUseSSL             | true               | /    |
|                                 | Cisco            | zCiscoRemodeEventClassKeys  |                    | /    |
|                                 | Cisco UCS        | zCiscoUCSCIMCEventsInterval | 60                 | /    |
|                                 | Cisco UCS        | zCiscoUCSCIMCPerInterval    | 300                | /    |
|                                 | Cisco UCS        | zCiscoUCSManagerPassword    | *****              | /    |
|                                 | Cisco UCS        | zCiscoUCSManagerPort        | 443                | /    |
|                                 | Cisco UCS        | zCiscoUCSManagerUseSSL      | true               | /    |
|                                 | Cisco UCS        | zCiscoUCSManagerUser        | admin              | /    |
|                                 | Modeler Controls | zCollectorClientTimeout     | 180                | /    |
|                                 | Modeler Controls | zCollectorDecoding          | utf-8              | /    |
|                                 | Misc             | zCollectorLogChanges        | false              | /    |
|                                 | zencommand       | zCommandCommandTimeout      | 15                 | /    |
|                                 | zencommand       | zCommandExistenceTest       | test -f %s         | /    |
|                                 | zencommand       | zCommandLoginTimeout        | 10                 | /    |
|                                 | zencommand       | zCommandLoginTries          | 1                  | /    |
|                                 | zencommand       | zCommandPassword            |                    | /    |
|                                 | zencommand       | zCommandPath                | \$ZENHOME/libexec  | /    |
|                                 | zencommand       | zCommandPort                | 22                 | /    |
|                                 | zencommand       | zCommandProtocol            | ssh                | /    |
|                                 | zencommand       | zCommandSearchPath          |                    | /    |
|                                 | zencommand       | zCommandUsername            |                    | /    |
|                                 | Control Center   | zControlCenterHost          | \$(here/managerip) | /    |

- 3 Edit configuration properties definitions as desired, and then click **Save**.

Definitions are applied to all devices currently in, and added to, this class (unless overridden at a lower level in the hierarchy).

## Groups

---

Groups are functional divisions that allow you to assign attributes to multiple objects with similar functions. Groups can be used, for example, to arrange objects along departmental lines. Groups do not appear on the Dashboard.

### Adding Groups

To add a group:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select the parent location in the groups hierarchy where you want to create a child group.
- 3 Click the **Add** icon.

The Add Group dialog appears.

- 4 Enter a name and description for the group, and then click **Submit**.

The group appears in the hierarchy. You can now move devices to this group. Select one or more devices in the device list, and then drag them to the new group.

### Moving a Group

To move a group:

- 1 Select the group in the hierarchy.
- 2 Drag the group to its new location.

The Move Organizer confirmation dialog appears.

- 3 Click **OK** to confirm the action.

The group appears at its new location in the hierarchy.

## Systems

---

Systems are intended to follow virtual setups, such as those in a network setup or systems grouped by functionality.

### Adding Systems

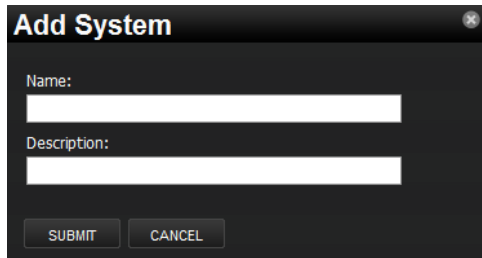
To add a system or sub-system:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select the parent location in the systems hierarchy where you want to create a new child system.
- 3 Click the **Add** icon.

The Add System dialog appears.

**Figure 85: Add System**

 A dark-themed dialog box titled "Add System" with a close button in the top right corner. It contains two text input fields: "Name:" and "Description:". Below the fields are two buttons: "SUBMIT" and "CANCEL".

- 4 Enter a name and description for the system, and then click **Submit**.

The system appears in the hierarchy. You can now move devices to this system. Select one or more devices in the device list, and then drag them to the new system.

### Moving a System

To move a system:

- 1 Select the system in the hierarchy.
- 2 Drag the system to its new location.

The Move Organizer confirmation dialog appears.

- 3 Click **OK** to confirm the action.

The system appears at its new location in the hierarchy.

## Locations

---

Locations are logical groupings for physical systems. They can be as general as city and state, or as specific as rack or closet. Locations do not appear on the Dashboard.

### Adding Locations

To add a location:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select the parent level in the Locations hierarchy where you want to create a child location.
- 3 Click the **Add** icon.

The Add Location dialog appears.

- 4 Enter a name and description for the location, and then click **Submit**.

The location appears in the hierarchy. You can now move devices to this location. Select one or more devices in the device list, and then drag them to the new location.

### Moving a Location

To move a location:

- 1 Select the location in the hierarchy.
- 2 Drag the location to its new place.

The Move Organizer confirmation dialog appears.

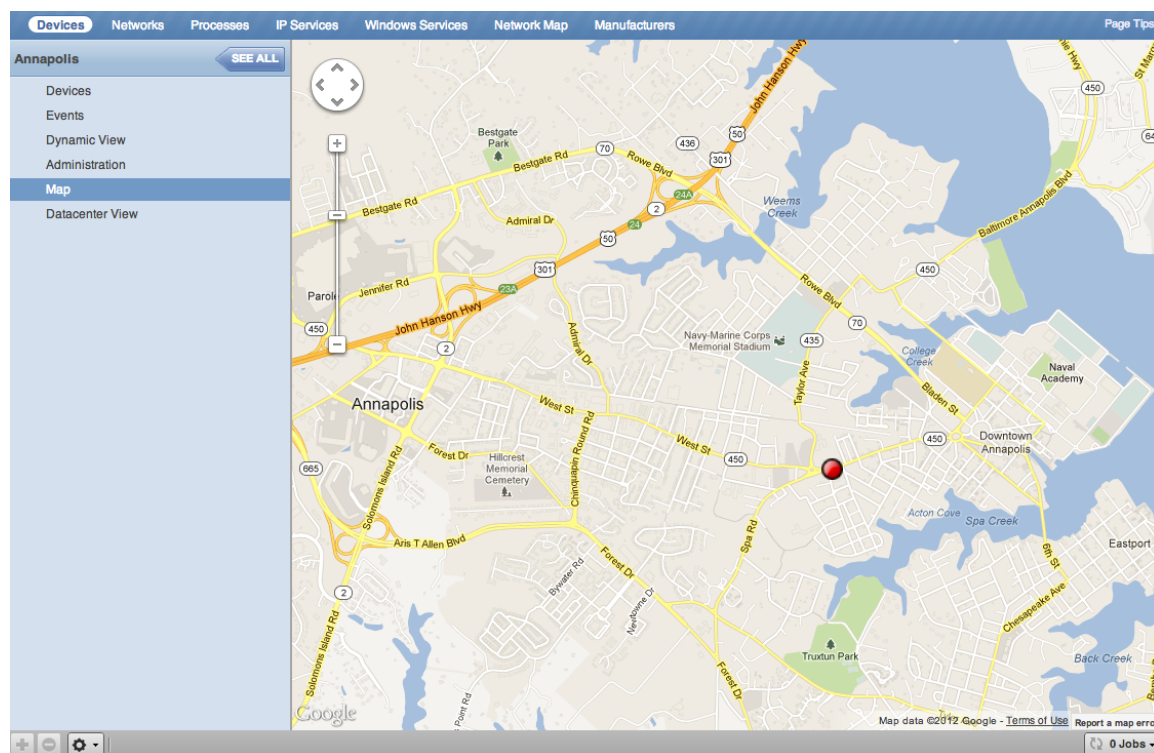
- 3 Click **OK** to confirm the action.

## Integration with Google Maps

The system can map locations by using a Google Maps mashup feature that sets the location's Address property to a valid Google Maps address. The selected location appears on the map as a dot. The color of the dot represents the highest severity of any event on any device in that location.

Network connections that span locations are represented on the map by lines. Each line color matches the status of the connection it represents.

**Figure 86:** Network Location - Google Map



To view the Google Map for a network location:

- 1 From the Navigation menu, select Infrastructure.

The device list appears.

- 2 In the hierarchy, select a location.
- 3 Click **Details**.
- 4 Select **Map**.

The network map appears.

---

**Note** You also can view the network location map from the Google Maps portlet on the Dashboard.

---

## Enabling Google Maps on Your System

Before you can use the Google Maps feature, you must register your Zenoss Core server key (UUID) with Zenoss. Typically, this is done when you license the product.

You can verify whether your server key is registered. From the Zenoss Core interface, select Advanced > Licensing. If the Licensing Report indicates that no license is associated with your server key, then open a support case through the customer support portal, at:

<https://support.zenoss.com>

Be sure to provide your Zenoss Core server key; this is required to generate the Google Maps API key.

## Setting an Address for a Location

Follow these steps to set a location address:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select a location in the hierarchy.
- 3 Select Edit from the **Action** icon.

The Edit Organizer dialog appears.

- 4 In the Description field, enter a description for the location.
- 5 In the Address field, enter a complete address that can be resolved by Google Maps. This address must include a valid zip code. (To test your address in advance, enter it at <http://maps.google.com>.)
- 6 Click **Submit**.

The selected address for the location is created. You must add at least one device to the location for the location "dot" to appear on the map.

## Clearing the Google Maps Cache

Sometimes there are issues with drawing the maps and seeing the network status of locations or connections. Clearing the Geocode cache will solve these problems. To clear the Geocode cache:

- 1 Select Infrastructure from the Navigation menu.

The device list appears.

- 2 Select a location in the hierarchy.
- 3 Select Clear Geocode Cache from the **Action** icon.

A confirmation dialog appears.

- 4 Click **OK**.

## Network Links

If two devices in the same network are in different map-able locations, a line is 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 is drawn. The color of the line represents 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.

## Drawing Map Links (zDrawMapLinks Configuration Property)

Calculating network links on the fly is an time-intensive procedure. If you have a large number of devices that have been assigned locations, drawing those links on the map may take a long time.

To save time, you can tell the system not to attempt to draw links for specific networks. You might want to do this, for example, for a local network comprising many devices that you know does not span multiple locations.

To edit the value for this property:

- 1 Select Infrastructure > Networks from the Navigation menu.

The Networks page appears.

- 2 Select a network or sub-network for which you want to disable map links.
- 3 Display configuration properties for the network.
- 4 Double-click the zDrawMapLinks configuration property in the list.

The Edit Config Property dialog appears.

- 5 De-select the value (uncheck the box), and then click **Submit**.

---

**Note** This setting will be inherited by networks or sub-networks below this selection in the hierarchy. If you have few networks for which links would be drawn, you might want to disable map links on /Networks, enabling it only on a network where you know a location-spanning WAN connection exists.

---

## Google Maps Example

This example will show you how to:

- Create and display Google map links of devices
  - Send a test event to see how map links are affected by system changes
- 1 Disable map links. (Refer to the procedure in Drawing Map Links for instructions.)
  - 2 Create two locations: "New York" and "Los Angeles." (Refer to the procedure in Adding Locations for instructions.)
  - 3 Enter the following values in the Address field of the Add Location dialog: "New York, NY" and "Los Angeles, CA" respectively.
  - 4 Set the location of a device to New York. Locate another device on the same network and set its location to Los Angeles.
  - 5 Select Locations in the hierarchy, click **Details**, and then select Map.

New York and Los Angeles are represented by dots on the map; however, no link is drawn between these locations.

- 6 Select Networks and re-enable map linking.
- 7 Select Infrastructure, then select Locations in the hierarchy.
- 8 Click **Details**, and then select Map.

A green line is now drawn between New York and Los Angeles.

- 9 Send an event with a severity of Critical to the device in New York. (For information about creating events, refer to the chapter titled Event Management.) Do not specify a component.
- 10 Return to the Locations map.

The dot representing New York is now red, but the link between New York and Los Angeles remains green.

- 11 Navigate to the New York device and determine the ID of the component that is connected to the network shared with the Los Angeles device.
- 12 Send another test event, this time specifying that component.



### 13 Return to the Locations map.

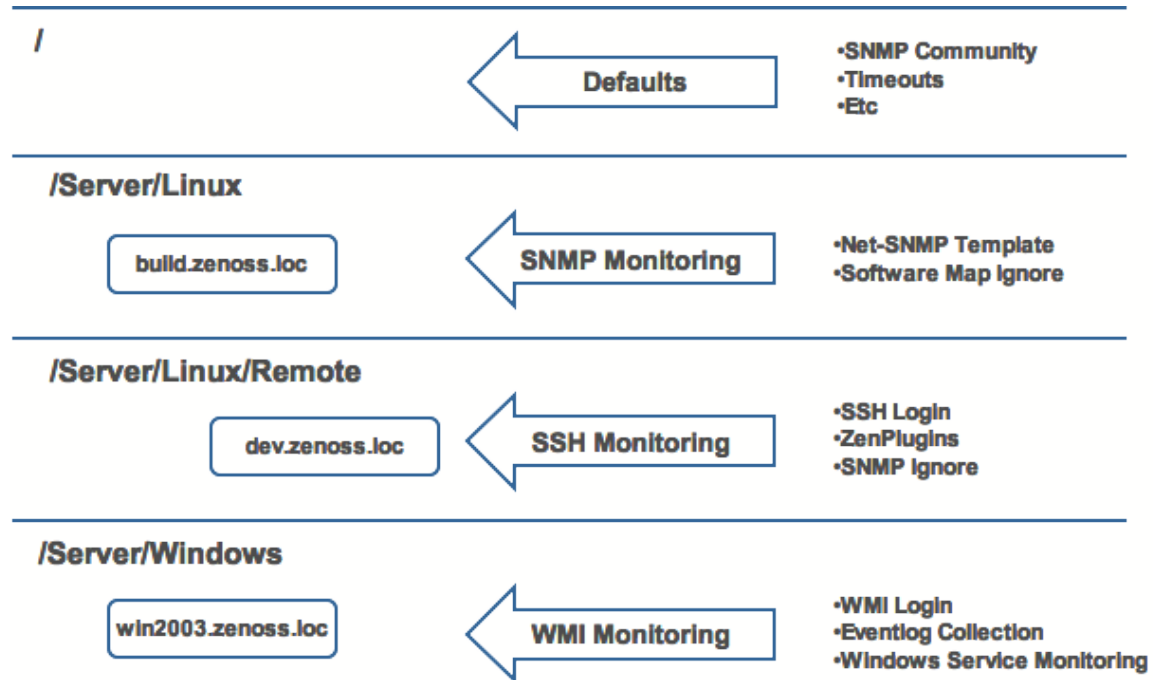
The link between New York and Los Angeles is now red.

## Inheritance

Inheritance is defined by how many attributes are applied to a device at different points in the device hierarchy.

The following diagram shows an example of how and where configuration properties can be set throughout the device class tree.

**Figure 87: Device Class Tree and Inheritance**



In this example, you can see that the default properties can be set at the highest level (/). However, as you travel further down the hierarchy, you see that you can override any of the configuration properties set at the root level.

The next two lines show how the device tree further defines properties for Linux servers. If you wanted, for example, to set up and use SNMP monitoring for all Linux servers (inclusive of) build.zenoss.loc, you could change these properties at the /Server/Linux level.

Further, if you wanted to change how you collect information for remote Linux servers, you could create a sub-group in /Server/Linux called /Server/Linux/Remote, setting these servers to use SSH monitoring and changing the associated properties for that sub-group.

Also within the /Server group you could create another sub-group for Windows servers that changes the configuration properties specifically for WMI monitoring.

All of these configuration properties and groupings co-exist, with any changes made lower in the hierarchy taking priority.

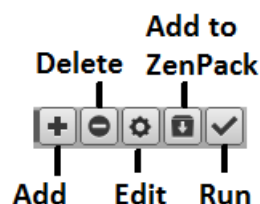
## User Commands

---

User commands allow you to execute arbitrary shell commands from the Zenoss Core master server. A user command is executed on the server rather than the remote device unless the command explicitly uses SSH to connect to the remote device.

You can define and run user commands on a device or organizer (device class, system, group, or location). You also can define commands globally. The User Commands menu bar shows the various functions that can be used in the User Commands screen. See the following sections for detailed instructions on adding and running user commands on specific devices or groups of devices.

**Figure 88:** User Commands Menu Bar



## Defining Global User Commands

---

Global commands appear in the Commands list of options located at the top of the Devices page.

To define global user commands:

- 1 Select **Advanced > Settings** from the Navigation menu.
- 2 In the left panel, select **Commands**.
- 3 In the Define Commands area, select **Add User Command** from the Action menu.

The Add User Command dialog appears.

- 4 Enter a name for the command, and then click **OK**. Only letters, numbers, and underscores are allowed in command names. Spaces are not allowed.

The Define Commands page appears.

- 5 In the Description field, enter a description of what the command will do.
- 6 In the Command section, enter the TALES expression-based command you want to run on the device.
- 7 Enter your system account password for confirmation, and then click **Save**.

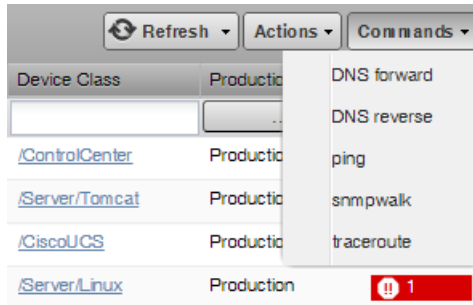
The command is saved and added to the commands menu.

**Note** Global commands also can be edited from a specific device. Changes to a global command from a device are not limited to that device.

## Running Global User Commands

To run a global user command, select one or more devices in the devices list, and then select a command from the Commands list of options.

**Figure 89: Run Commands**



## Defining User Commands for a Single Device

To define a user command for a device:

- 1 Select **Infrastructure** from the Navigation menu.
- The Devices page appears.
- 2 Click a device name from the list to open the Device Overview page.
- 3 In the left panel, select **Device Administration**.
- 4 In the User Commands area, click the **Add a User Command** icon.

The Add New User Command dialog appears.

**Figure 90: Add New User Command**

- 5 Enter the following information about the user command:

- **Name** - Name of the user command.
  - **Description** - Description of what the command will do.
  - **Command** - TALEs expression-based command you want to run.
  - **Confirm Password** - Enter your system account password for confirmation.
- 6 Click **Submit**.

The command is saved and added to the user commands menu.

- 7 Optionally, test the command by selecting the command from the list and clicking the **Run** icon.

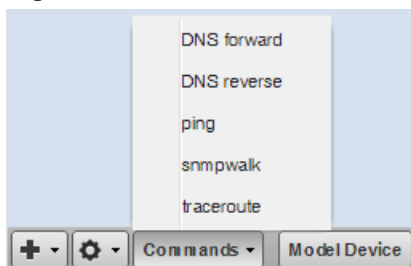
## Running User Commands for a Single Device

---

To run a command defined for a single device:

- 1 Navigate to the **Infrastructure > Devices** page.
- 2 Click the device name in the device list to open the Device Overview page.
- 3 Select the command from the Commands list of options located at the bottom of the page.

**Figure 91: Commands Menu**



## Defining User Commands for All Devices in an Organizer

---

To define a user command for all devices in an organizer:

- 1 On the Infrastructure page, select a device organizer in the devices hierarchy (e.g., Server/Linux)
- 2 Click **Details**.
- 3 In the left panel, select **Device Administration**.
- 4 In the User Commands area, click the **Add a User Command** icon.

The Add New User Command dialog appears.

**Figure 92: Add New User Command**

- 5 Enter the following information about the user command:
  - **Name** - Name of the user command.
  - **Description** - Description of what the command will do.
  - **Command** - TALEs expression-based command you want to run.
  - **Confirm Password** - Enter your system account password for confirmation.
- 6 Click **Submit**.

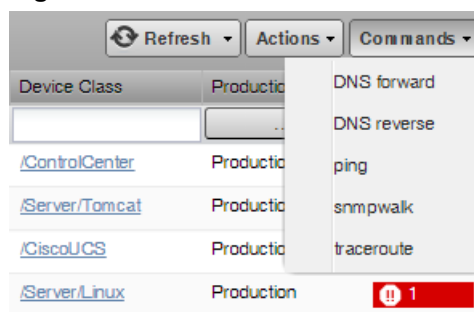
The command is saved and added to the user commands menu.

- 7 Optionally, test the command by selecting the command from the list and clicking the **Run** icon.

## Running User Commands for Devices in an Organizer

To run a command defined for devices in an organizer:

- 1 On the Infrastructure page, select a device organizer.
- 2 Select one or more devices in the filtered view.
- 3 Select the command from the Commands list located at the top of the page.

**Figure 93: Run Commands**

## User Command Example: Echo Command

This example shows how to create an echo user command. You can see the use of TALEs expressions in the definition of this command.

- 1 Add a command called “echoDevice”
- 2 In the command definition, echo the name and IP address of the device:

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

In a TALEs expression, ‘here’ is the object against which the expression is executed. Some TALEs expressions in the system have other variables (such as evt for event, and dev or device for the device). See the Appendix titled TALEs Expressions for more information about TALEs expressions syntax.

- 3 Select a device and then run the command.
- 4 Edit the command to add more information:

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

- 5 Run the command against a group of devices and view the command outputs.

## 12

## Managing Users

---

Each user has a unique user ID, which allows you to assign group permissions and alerting rules that are unique to each user. Unique IDs also help ensure secure access to the system.

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

### Creating User Accounts

---

To create a user account:

- 1 From the Navigation menu, select **Advanced**.

The Settings page appears.

- 2 In the left panel, select **Users**.

The users and groups administration page appears.

- 3 From the Action icon, select Add New User.

The Add User dialog appears.

- 4 In the Username field, enter a unique name for the account.
- 5 In the Email field, enter the user account email address. Any alerts that you set up for this user will be send 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.

### Editing User Accounts

---

To access and edit user account information:

- 1 In the Users list, click the name of the user you want to edit.

The edit user page appears. The following example shows the admin user.

**Figure 94: Edit User**

**ZenUsers > admin**

State at time: 2015-01-14 19:11:13

Automatically generate a new password and send it to the email listed below. **Reset Password**

**USER PREFERENCES**

Reset all preferences such as grid columns and filters to their default values. **Reset Preferences**

**USER SETTINGS**

Roles: Manager (selected), ZenManager, ZenOperator, ZenUser

Groups: (empty)

Email: admin@zenoss.com

Pager: (empty)

Default Page Size: 40

Default Admin Role: ZenUser

Network Map Start Object: (empty)

Time Zone: America/Chicago

Set New Password: (empty)

Confirm New Password: (empty)

Current Password for admin: (empty) **Save Settings**

2 Make changes to one or more settings:

- **Reset Password** - Facilitates user self-service by allowing a user to reset his or her own password. Click to reset and email the new password to the email address associated with the user's account.
- **User Preferences** - Resets all preferences such as grid columns and filters to their default values.
- **Roles** - Assign one or more roles (user privileges) to the user. To edit or assign roles, you must be a system Admin or be assigned the Manager role.

For more information about user roles, and for a list of available roles and the privileges they provide, see [Roles](#) on page 147.

- **Groups** - Specify one or more groups to which this user belongs.
- **Email** - Enter the user's email address. To verify that the address is valid, click the test link.
- **Pager** - Enter the user's pager number.
- **Default Page Size** - Controls how many entries (by default) appear in tables. Enter a value for the default page size. The default value is 40.
- **Default Admin Role** - Select the default role that this user will have for administered objects associated with him or her.
- **Network Map Start Object** - Specify the default view for this user in the network map.
- **Time Zone** - Specify the time zone to be displayed on all charts and graphs within the product.
- **Set New Password / Confirm New Password** - Enter a new password for the user and confirm the entry.

Enter your password, and then click **Save** to confirm and save the changes for the user.



## Associating Objects with Specific Users

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

For more information about object-specific roles, see [Roles](#) on page 147.

To create an object association:

- 1 From **Advanced > Settings**, select **Users** in the left panel.
- 2 Select a user.
- 3 From the Edit page, select **Administered Objects** in the left panel.

The list of administered objects appears.


- 4 Select an object type from the Administered Objects Action menu. You can add:
  - Device
  - Device class
  - System
  - Group
  - Location

The Add Administered Device dialog appears.

- 5 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 95: Administered Objects - Objects Added**

| Name   | Role      | Level |
|--|-----------|-------|
|  test | ZenUser ▼ | 1     |

Save

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

---

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

---

- 7 Click **Save** to save changes.

### Adding Administrators

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

- 1 Navigate to the object you want to add to the user's list of administered objects.
- 2 Select **Device Administration**.

**Figure 96: Administered Objects - Add Administrator**

The screenshot shows the Zenoss Core Administered Objects interface. On the left is a navigation pane with 'Device Administration' selected. The main area is divided into three sections: 'Maintenance Windows', 'User Commands', and 'Administrators'.

**Maintenance Windows Table:**

| Enabled | Name                 | Start               | Duration     | Repeat                | State       |
|---------|----------------------|---------------------|--------------|-----------------------|-------------|
| No      | 1st of Month         | 2014/12/01 02:00:00 | 01:00:00 hrs | Monthly: day of month | Maintenance |
| Yes     | Every Thursday Night | 2014/12/11 22:00:00 | 30:00 mins   | Weekly                | Maintenance |
| Yes     | One Time Testing     | 2014/12/20 08:00:00 | 04:00:00 hrs | Never                 | Test        |

**User Commands Table:**

| Name        | Command  |
|-------------|--|
| DNS forward | host \$(device/id)                             |
| DNS reverse | host \$(device/manageip)                       |
| ping        | \$(device/pingCommand) -c2 \$(device/mana...   |
| snmpwalk    | snmpwalk -\$(device/zSnmpVer) -c\$(device/z... |
| traceroute  | \$(device/tracerouteCommand) -q 1 -w 2 \$(d... |

**Administrators Table:**

| Name       | Role    | Email   | Pager |
|------------|---------|---------|-------|
| cgilchrist | Manager | cgil... |       |

- Click the Add Administrator icon in the Administrators area.

The Add Administrator dialog appears.

- 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.

## User Groups

Zenoss Core 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, select **Advanced > Settings**, and then select **Users** from the left panel.

The groups area shows each user group and the users assigned to that group.

### Creating User Groups

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

To create a user group:

- Navigate to **Advanced > Settings**.
- In the left panel, select **Users**.

The Users page appears.

- From the Groups area Action menu, select **Add New Group**.

The Add Group dialog appears.

- In the Group field, enter a name for this user group, and then click **OK**.

The group name appears in the Groups list.

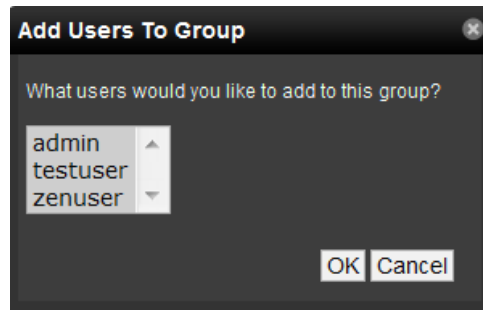
- 5 Click the name of the group you created.

The Users in Group page appears.

- 6 From the Action menu, select Add User.

The Add User to Group dialog appears.

**Figure 97: Add User to Group**



- 7 From the User list of selections, select one or more users you want to add to the group, and then click **OK**.

The user or users you select appear in the list of users for this group.

You also can choose administered objects and alerting rules for this user group. These alerting rules will apply to all users in the group. The user's original alerting rules and objects will also apply.

## Roles

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

The following table lists available roles.

| Role        | Permissions  |
|-------------|--|
| ZenUser     | Provides global read-only access to system objects.  |
| ZenManager  | Provides global read-write access to system objects.   |
| Manager     | Provides global read-write access to system objects. Additionally provides read-write access to the Zope object database.  |
| ZenOperator | Provides event management. Combine the ZenOperator role with the ZenUser role to allow users read-only access to the system, but also allow them to acknowledge and close events, move events to history, and add log messages to events. You can associate the ZenOperator role with an individual device, a device class, or a group of devices. |

## Device Access Control Lists

### About Device Access Control Lists (ACL)

Zenoss Core supports fine-grained security controls. For example, this control can be used to give limited access to certain departments within a large organization or limit a customer to see only his own data. A user with limited access to objects also has a more limited view of features within the system. As an example, most global views, such as the network map, event console, and all types of class management, are not available. The device list is available, as are the device organizers: systems, groups, and locations. A limited set of reports can also be accessed.

## Permissions and Roles

Actions in the system are assigned permissions. For instance to access the device edit screen you must have the “Change Device” permission. Permissions are not assigned directly to a user; instead, permissions are granted to roles, which are then assigned to a user. A common example is the ZenUser role. 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. When you assign a role to a user using the Roles field (on the Edit page), it is global.

## Administered Objects

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

## 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.

## Assigning Administered Object Access

For each user or group there is an Administered Objects selection, which lets you add items for each type of administered object. After adding an object you can assign it a role. Roles can be different for each object, so a user or group might have ZenUser on a particular device but ZenManager on a location organizer. If multiple roles are granted to a device through 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.

## Portlet Access Control

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

### Example: 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 Make sure that no role is assigned to the user.
- 3 Edit the user's administered objects.
- 4 Add an existing device to the user.

The device's role will default to ZenUser.

- 5 Log out of your browser, or open a second browser and then log in as acltest.
- 6 Select Infrastructure.

You should see only the device you assigned to acltest.

- 7 Navigate to the device and notice that the edit capabilities are not available. This is because you are in read-only mode for this device.

## Example: Restricted User with ZenOperator Role

The ZenUser role from the previous section allows read-only access to devices. By adding the ZenOperator role to specific devices, device classes, or groups of devices, a user will be able to acknowledge and close events, move events to history, and add log messages to events.

To add the ZenOperator role to specific devices, device classes, or groups of devices:

- 1 Select the user name whose role must be changed on certain devices.
- 2 In the left-hand pane, click Administered Objects.
- 3 Click the Action icon and choose the device, device class, or other device organizer to which you want to grant the ZenOperator role.
- 4 Select the ZenOperator role from the drop-down menu for the newly selected device, device class, or device organizer.

The user now has the ZenUser role for all devices in this instance, with the exception of the device(s) selected above which function under the ZenOperator role.

## Example: Restricted User with ZenManager Role

Following the example above:

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

## Example: Adding Device Organizers

- 1 Go to Groups and create a group called "RestrictGroup."
- 2 Go to the acltest user's administered objects and add the group to the user.
- 3 Logged in as acltest, notice that groups can be added to a user.
- 4 Place a device within this group and as acltest you should not only see the device within the group but also in the device list.

## 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.

## 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.

## Detailed Restricted Screen Functionality

### Dashboard

By default, the dashboard is configured with three portlets:

- Object Watch List
- Device Issues
- Production State

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

### **Device List**

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

### **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.

### **Reporting**

Reports are limited to device reports and performance reports.

## 13

# Reporting

Zenoss Core provides many useful summaries of monitored resources at the **Reports** tab of the Zenoss Core web interface.

**Note** If you experience any stairstepping in your graphs, you may want to change the reporting collection interval in Zenoss Core. For example, setting the reporting collection interval to 60 minutes tells Zenoss Core to update the API-driven reporting data at that interval which is different than the native collection interval of 15 minutes.

To view a report, select the report's name in the left column.

## Device Reports

### All Devices

A summary of each device Zenoss Core is monitoring.

| Column         | Content   |
|----------------|---|
| <b>Name</b>    | The name of the device.   |
| <b>Class</b>   | The Zenoss Core device class associated with the device.  |
| <b>Product</b> | The hardware model information associated with the device, which is provided by the device's SNMP MIB, or entered manually. If the value in this column is an SNMP OID, the Zenoss Core database does not include a definition of the object. |
| <b>State</b>   | The device's production state. Valid states include <i>Production</i> , <i>Pre-Production</i> , <i>Test</i> , and <i>Maintenance</i> .  |
| <b>Ping</b>    | The result of the most recent ping of the device.   |
| <b>SNMP</b>    | The result of the most recent attempt to gather data through the device's SNMP agent.   |

### All Monitored Components

A summary of each component Zenoss Core is monitoring.

| Column           | Content  |
|------------------|--|
| <b>Device</b>    | The name of the device which contains the component, with a link to its overview page. |
| <b>Component</b> | The name of the component, with a link to its overview page.                           |

| Column             | Content   |
|--------------------|---|
| <b>Type</b>        | The class associated with the component.  |
| <b>Description</b> | A description of the component; typically, the component's name.                    |
| <b>Status</b>      | The state of the component as of the most recent attempt to gather monitoring data. |

## Device Changes

A summary of the devices in which changes were detected during the most recent collection of model data.

| Column            | Content   |
|-------------------|---|
| <b>Name</b>       | The name of the changed device, with a link to its overview page.                                 |
| <b>Class</b>      | The Zenoss Core device class associated with the device.  |
| <b>First Seen</b> | The timestamp of the initial collection of modeling data for the device.                          |
| <b>Collection</b> | The timestamp of the collection in which a change was detected before the most recent collection. |
| <b>Change</b>     | The timestamp of the most recent collection in which a change was detected.                       |

## MAC Addresses (MAC Address Inventory)

A list of the unique device name, interface ID, and MAC address combinations in the Zenoss Core database.

| Column              | Content  |
|---------------------|--|
| <b>Device</b>       | The name of a device, with a link to its overview page.          |
| <b>Interface ID</b> | The ID of a network interface, with a link to its overview page. |
| <b>MAC address</b>  | A MAC address.   |

## Model Collection Age

A summary of devices that were not available for modeling data collection during the most recent 48 hour period.

| Column            | Content  |
|-------------------|--|
| <b>Name</b>       | The name of the changed device, with a link to its overview page.        |
| <b>Class</b>      | The Zenoss Core device class associated with the device.                 |
| <b>First Seen</b> | The timestamp of the initial collection of modeling data for the device. |
| <b>Collection</b> | TBD  |
| <b>Change</b>     | The timestamp of the most recent collection attempt.                     |

## New Devices

The list of devices that were discovered and added to Zenoss Core recently.

| Column            | Content  |
|-------------------|--|
| <b>Name</b>       | The name of the changed device, with a link to its overview page.        |
| <b>Class</b>      | The Zenoss Core device class associated with the device.                 |
| <b>First Seen</b> | The timestamp of the initial collection of modeling data for the device. |
| <b>Collection</b> | TBD  |



| Column | Content |
|--------|---------|
| Change | TBD     |

## Ping Status Issues

A list of the devices which were down during the most recent collection of monitoring data.

## SNMP Status Issues

A list of the devices for which no SNMP agent responded during the most recent collection of monitoring data.

## Software Inventory

A list of the software installed in the devices and components which Zenoss Core monitors.

| Column       | Content   |
|--------------|---|
| Manufacturer | The name of the company that makes the software product.                      |
| Product      | The name of the software product.   |
| Count        | The total number of devices or components on which the software is installed. |

## Event Reports

### All EventClasses (All Event Classes)

A list of each item in the event hierarchy in Zenoss Core. Each item (class) includes the total number of subclasses, instances, and events associated with the class.

| Column     | Content   |
|------------|---|
| Name       | The event class name.   |
| Subclasses | The total number of subclasses associated with the event class. |
| Instances  | The total number of instances of the class and its subclasses.  |
| Events     | The total number of events associated with the class.           |

### All EventMappings (All Event Mappings)

A list of each item in the event mapping hierarchy in Zenoss Core. Each item (event mapping) includes its key and example text, and a count of the events associated with the event mapping.

| Column        | Content   |
|---------------|---|
| Name          | The name of the event mapping, which includes its location in the event class hierarchy, and its key. |
| EventClassKey | The unique identifier of the event mapping.   |
| Evaluation    | A portion of the example associated with the event mapping.   |
| Events        | The total number of events associated with the event mapping.   |

## All Heartbeats

A list of all Zenoss Core daemons, showing the number of seconds elapsed since each daemon sent a heartbeat event.

| Column    | Content  |
|-----------|--|
| Device    | The device on which the daemon is running.                             |
| Component | The name of the daemon.  |
| Seconds   | The number of seconds elapsed since the daemon sent a heartbeat event. |

## Performance Reports

---

### Availability Report

### CPU Utilization

### Filesystem Util Report

### Interface Utilization

### Memory Utilization

### Threshold Summary

# ZenPacks

---

ZenPacks extend and modify the system to add new functionality. This can be as simple as adding new device classes or monitoring templates, or as complex as extending the data model and providing new collection daemons.

You can use ZenPacks to add:

- Monitoring templates
- Data sources
- Graphs
- Event classes
- User commands
- Reports
- Model extensions
- Product definitions

Simple ZenPacks can be created completely within the user interface. More complex ZenPacks require development of scripts or daemons, using Python or another programming language.

ZenPacks can be distributed for installation on other Zenoss Core systems.

## Viewing ZenPacks

---

To see which ZenPacks are loaded on your system:

- 1 From the Navigation menu, select Advanced.

The Settings page appears.

- 2 Select ZenPacks in the left panel.

The list of loaded ZenPacks appears.

**Figure 98:** Loaded ZenPacks

| Settings Control Center Monitoring Templates Jobs MIBs Licensing Page Tips   |   |         |        |         |     |
|--|---|---------|--------|---------|-----|
| <a href="#">Settings</a><br><a href="#">Commands</a><br><a href="#">Users</a><br><b><a href="#">ZenPacks</a></b><br><a href="#">Portlets</a><br><a href="#">Versions</a><br><a href="#">Events</a><br><a href="#">User Interface</a><br><a href="#">LDAP</a> | Loaded ZenPacks   |         |        |         |     |
|  | Pack  | Package | Author | Version | Egg |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.AdvancedSearch</a>    | zenoss  | Zenoss | 1.1.4   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.AixMonitor</a>        | zenoss  | Zenoss | 1.3.0   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.ApacheMonitor</a>     | zenoss  | Zenoss | 2.1.4   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.AuditLog</a>          | zenoss  | Zenoss | 1.3.0   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.BigIpMonitor</a>      | zenoss  | Zenoss | 2.6.3   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.BrocadeMonitor</a>    | zenoss  | Zenoss | 2.1.1   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.CatalogService</a>    | zenoss  | Zenoss | 3.0.9   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.CheckPointMonitor</a> | zenoss  | Zenoss | 2.0.0   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.CiscoMonitor</a>      | zenoss  | Zenoss | 5.3.1   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.CiscoUCS</a>          | zenoss  | Zenoss | 1.9.1   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.ControlCenter</a>     | zenoss  | Zenoss | 1.0.0   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.Dashboard</a>         | zenoss  | Zenoss | 1.0.3   | Yes |
|  | <input type="radio"/> <a href="#">ZenPacks.zenoss.DellMonitor</a>       | zenoss  | Zenoss | 2.2.0   | Yes |

**Note** Alternatively, you can view loaded ZenPacks from the command line:

```
zenpack --list
```

**Note** You can no longer create a ZenPack from this action menu. You can only create a ZenPack from the command-line interface (CLI). For example:

```
serviced service run zope zenpack create ZenPacks.awesome.Extension
```

## ZenPack Information Resources

Zenoss provides numerous ZenPacks that add and extend system functionality. The Zenoss Core web interface includes a link (the question mark icon) to the documentation of the ZenPacks included in your installation of Zenoss Core, and [ZenPack Catalog](#) provides detailed descriptions of all of the ZenPacks developed by Zenoss.

You may also create your own ZenPacks, or download and install ZenPacks developed by others. The following list identifies ZenPack resources.

- [ZenPack Discussion Forum](#)
- [ZenPack Development Forum](#)
- [Public Zenoss repositories on GitHub](#)

## Installing and Upgrading ZenPacks

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

ZenPacks can introduce fundamental changes to the ways in which Zenoss Core operates, so the safest approach to installing a ZenPack is to suspend operations temporarily.

When upgrading a ZenPack, the old version of the ZenPack is removed before installing the new version. As such, the ZenPack framework removes all services tagged with the name of the ZenPack resulting in a loss of any edits

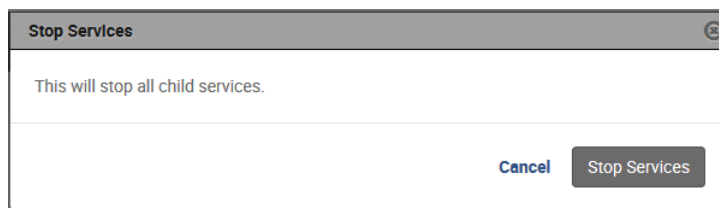
that were made to the service definition (e.g., RAMCommitment, Number of Instances, etc.). Take note of any edits you have made to the service definition so that you can re-create those settings after the upgrade.

Zenoss recommends making a backup of Zenoss Core before installing a ZenPack, so that you can restore to the previous state if you encounter problems with the installation or upgrade.

- 1 Log in to the Zenoss Control Center browser interface.

The screenshot displays the Zenoss Control Center interface. At the top, there's a navigation bar with tabs for Applications, Resource Pools, Hosts, Logs, and Backup / Restore. The 'Applications' tab is active. Below the navigation bar, there's a section titled 'Applications' with a 'Services Map' and 'Application' button. A table lists applications: 'Internal Services' (status: Running) and 'Zenoss.core (v5.0)' (status: Stopped). Below this is the 'Application Templates' section, showing a table with 'Zenoss.core (v5.0)' as the only template.

- 2 In the **Applications** table, identify the name of the Zenoss Core instance to modify.
- 3 Stop the instance, and verify its subservices are stopped.
  - a In the **Actions** column of the **Applications** table, click **Stop**.



- b In the **Stop Services** dialog, click **Stop Services**.
  - c In the **Applications** column of the **Applications** table, click the name of the stopped instance, and then scroll down to the **Services** table.
- 4 When all services are stopped, restart the following individual services.
    - MariaDB
    - RabbitMQ
    - Zope
    - redis
    - zeneventserver

**Note** In the **Services** table, the Failing icon (a red circle with an exclamation point) in the **Status** column represents the cumulative result of one or more customized health checks. To view the status of individual health checks, move the pointer over the icon, which displays a pop-up.

When the Failing icon is present, a service is unable to support the normal operations of Zenoss Core. For this maintenance task, the Zope health checks includes failing checks of `zproxy_answering` and `zenhub_answering`, which do not affect this procedure.

- 5 Log in to a Zenoss Control Center host as a user with `sudo` and `docker` privileges.
- 6 Download the egg file of the ZenPack to install or upgrade from the [ZenPack Catalog site](#).  
The ZenPack egg file must be located on the local filesystem, and readable by the current user.

- 7 Change directory to the directory in which the ZenPack egg file is located, and then install the ZenPack.

```
serviced service run zope zenpack install ZenPack-File.egg
```

If a ZenPack introduces a new daemon, the daemon becomes a new service in the current instance of Zenoss Core.

- 8 In the Zenoss Control Center browser interface, scroll up to the top of the page, and then start Zenoss Core.

## Removing a ZenPack

The ZenPack removal procedure requires stopping and starting Zenoss Core.

**Note** Removing a ZenPack can have unexpected consequences.

- Removing a ZenPack removes all objects provided by the ZenPack, as well as all objects that depend on code provided by the ZenPack.
- Removing a newer version of a ZenPack to install an older version fails if the newer version includes migration code.
- Removing a ZenPack that installs a device class removes the device class, any contained device classes, and all devices in that class.

Often, the safest choice is not to remove a ZenPack.

To avoid negative consequences, Zenoss recommends the following practices.

- Review the documentation of the ZenPack to remove for information about classes and daemons (services) associated with it.
- Delete data sources provided by the ZenPack to remove.
- Back up Zenoss Core before removing a ZenPack.

- 1 Log in to the Zenoss Control Center browser interface.

The screenshot shows the Zenoss Control Center web interface. The top navigation bar includes the Control Center logo and links for Applications, Resource Pools, Hosts, Logs, and Backup / Restore. The user is logged in as 'ecuser'. The main content area displays the 'Applications' section with a table of installed applications. Below this, there is a section for 'Application Templates'.

| Application        | Description       | Status  | Deployment ID | Resource Pool | Virtual Host Names   | Actions           |
|--------------------|-------------------|---------|---------------|---------------|----------------------|-------------------|
| Internal Services  | Internal Services | Running | Internal      | N/A           | N/A                  | N/A               |
| Zenoss.core (v5.0) | Zenoss Core       | Stopped | Test          | default       | https://zenoss5x.c70 | Start Stop Delete |

| Application Template | ID                               | Description | Actions |
|----------------------|----------------------------------|-------------|---------|
| Zenoss.core (v5.0)   | 44441b24f9761b063d9f5ce6ec3d4eb7 | Zenoss Core | Delete  |

- 2 In the **Applications** table, identify the name of the Zenoss Core instance to modify.
- 3 Stop the instance, and verify its subservices are stopped.
  - a In the **Actions** column of the **Applications** table, click **Stop**.



- b In the **Stop Services** dialog, click **Stop Services**.
  - c In the **Applications** column of the **Applications** table, click the name of the stopped instance, and then scroll down to the **Services** table.
- 4 When all services are stopped, restart the following individual services.
  - MariaDB
  - RabbitMQ
  - Zope
  - redis
  - zeneventserver

---

**Note** In the **Services** table, the Failing icon (a red circle with an exclamation point) in the **Status** column represents the cumulative result of one or more customized health checks. To view the status of individual health checks, move the pointer over the icon, which displays a pop-up.

---

When the Failing icon is present, a service is unable to support the normal operations of Zenoss Core. For this maintenance task, the Zope health checks includes failing checks of `zproxy_answering` and `zenhub_answering`, which do not affect this procedure.

- 5 Log in to a Zenoss Control Center host as a user with `sudo` and `docker` privileges.
- 6 From the list of installed ZenPacks, identify and note the name of the ZenPack to remove.
  - a Execute the following command:

```
serviced service run zope zenpack list
```

The first item of each line of output is the full name of an installed ZenPack.

- 7 Remove the ZenPack.
 

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

```
serviced service run zope zenpack uninstall ZenPack-Name
```

- 8 Remove services associated with the removed ZenPack, if necessary.

The daemons a ZenPack provides become services in this release of Zenoss Core.

---

**Note** Some ZenPacks provide services upon which other ZenPacks rely. Make sure the service you remove is not needed by another ZenPack.

---

The following example removes the `zenwebtx` service, which is provided by the `ZenPacks.zenoss.ZenWebTx` ZenPack.

```
serviced service rm zenwebtx
```

- 9 In the Zenoss Control Center browser interface, scroll up to the top of the page, and then start Zenoss Core.

## General Administration and Settings

---

### Recovering from a POSKey error during login

---

To perform this procedure, you need an account on the Zenoss Control Center master host that has permission to use the Zenoss Control Center command-line interface.

Zenoss Core automatically recovers from most causes of POSKey errors that prevent logging into the Zenoss Core interface. If a POSKey error does occur, perform this procedure.

- 1 Log in to an account on the Zenoss Control Center master host that has permission to use the Zenoss Control Center command-line interface.
- 2 Stop the Zope and authentication services.

```
serviced service stop zope && serviced service stop zauth
```

- 3 Monitor the services until they are stopped.

```
watch serviced service status zope  
watch serviced service status zauth
```

- 4 Rebuild the session database.

```
serviced service run mariadb rebuild_zodb_session
```

- 5 Restart the authentication and Zope services.

```
serviced service start zauth && serviced service start zope
```

### Events Settings

---

You can adjust events settings for:

- Events database connection
- Event maintenance

### Changing Events Database Connection Information

To edit events database connection settings, make changes in the `zeneventserver.conf` file. You can edit the file directly, or run a configuration script.



Configurable database connection settings are:

- **JDBC Hostname (zep.jdbc.hostname)**- Specify the IP address of the host.
- **JDBC Port (zep.jdbc.port)**- Specify the port to use when accessing the events database.
- **JDBC Database Name (zep.jdbc.dbname)**- Specify the database name.
- **JDBC Username (zep.jdbc.username)**- Specify the user name for the database.
- **JDBC Password (zep.jdbc.password)**- Specify the password for the database.

To edit these values, run the `zeneventserver` configuration script, as follows:

```
zeneventserver-config -u zep.jdbc.  
    Name=  
    Value
```

Where *Name* is the partial setting name and *Value* is the value you want to specify for the setting.

## Changing Events Maintenance Settings

To edit maintenance settings, make changes to one or more fields on the Event Configuration page (Advanced > Settings > Events):

- **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 be aged by the system.
- **Event Aging Threshold(minutes)** - Specify how long the system should wait before aging an event.
- **Event Archive Interval(minutes)** - Specify how long a closed event remains in the Event Console before moving to the event archive.
- **Delete Archived Events Older Than(days)** - Enter a value in days. Zenoss Core will automatically delete events from the event archive that are older than this value.
- **Default Syslog Priority**- Specify the default severity level assigned to an event coming from zensyslog if no priority can be determined from the event.
- **Default Availability Report(days)** - Enter the number of days to include in the automatically generated Availability Report. This report shows a graphical summary of availability and status.

## Rebuilding the Events Index

If you encounter inconsistent search results, you can rebuild the events index. As the `zenoss` user, follow these steps:

- 1 Stop `zeneventserver`:

```
zeneventserver stop
```

- 2 Delete the index:

```
rm -rf $ZENHOME/var/zeneventserver/index
```

---

**Note** In order to execute a command using `$ZENHOME` (`/opt/zenoss` for the `zenoss` user), you must be attached to the container holding the Zenoss Core master host.

---

- 3 Restart `zeneventserver`:

```
zeneventserver start
```

Depending on the number of events in the database, it may take a significant amount of time for indexing to complete. Until every event is indexed, the number of events shown in the event console may be inconsistent.

## Audit Logging

The audit log tracks user actions in syslog or log files. The system maintains logged information in a format optimized for searching and reporting.

Logged information can appear in several locations:

- Log file
- Rotating log files (limited by time or size)
- syslog

By default, the `$ZENHOME/log/audit.log` file stores the latest 10MB of data, with three rolling backups.

---

**Note** In order to execute a command using `$ZENHOME` (`/opt/zenoss` for the `zenoss` user), you must be attached to the container holding the Zenoss Core master host.

---

## Configuring the Audit Logs

Settings in the `$ZENHOME/etc/audit_log.conf` configuration file determine the location and content of logged information output.

The `audit_log.conf` and `audit_log.conf.example` files are created at installation (if they do not exist).

An entry in the audit log indicates that a user attempted an action, but does not always indicate whether that action was successful. For example, a log entry stating that a user added a device simply indicates that the user created a job to add the device; however, the job could still fail when it runs at a later time.

As shown in the following sample, the configuration file contains examples and instructions for each of the output methods.

```
## Audit Log configuration file
## ## Initially this outputs up to 10 megs to ZENHOME/log/audit.log with
## 3 backups.
##
## To output to the syslog or somewhere else:
## - Uncomment the desired handlers and formatters, or create your own.
## - Update the "keys" lists under [handlers] and [formatters].
## - Update the "handlers" list under [logger_audit].
## - Restart Zope with "zenwebserver restart".
##
## To change the log severity level:
## - Update "level" under [logger_audit]
##
## This file has all the features of the Python logging file format:
## http://docs.python.org/library/logging.config.html#configuration-file-
## format

[loggers]
## DO NOT CHANGE
keys=audit

##
##
## List all output handlers here. (part 1 of 3)
```

```

## This should match part 3 below.
##
## Example: keys=syslog,file,rotatingfile,timedrotatingfile,console
##
[handlers]
keys=rotatingfile

##
##
## List all string formatters here. (part 2 of 3)
##
## Example: keys=syslog,file,console
##
##

[formatters]
keys=file

[logger_audit]
## DO NOT CHANGE
qualname=zen.audit
propagate=0

##
##
## This is the severity level of all audit messages.
## (DEBUG, INFO, WARNING, ERROR, CRITICAL)
##
## You can override the level of individual handlers below,
## or keep them as NOTSET to use this default level.
##
##
level=INFO

##
##
## List all output handlers here. (part 3 of 3)
## This should match part 1 above, except "handlers=" not "keys=".
##
## Example: handlers=syslog,file,rotatingfile,timedrotatingfile,console
##
##
handlers=rotatingfile
##### Output Handlers

## SysLog
##
## See http://docs.python.org/library/logging.handlers.html#sysloghandler
##
## Here are typical configurations:
##
## Linux: args=('/dev/log', handlers.SysLogHandler.LOG_USER)
## OS/X : args=('/var/run/syslog', handlers.SysLogHandler.LOG_USER)
## UDP : args=('localhost', handlers.SYSLOG_UDP_PORT),
##          handlers.SysLogHandler.LOG_USER)
##
##
##[handler_syslog]
##class=handlers.SysLogHandler
##level=NOTSET

```

```

##formatter=syslog
##args=()

## File
##
## See http://docs.python.org/library/logging.handlers.html#filehandler
##
## To store in ZENHOME/log:
class=Products.ZenUtils.configlog.ZenFileHandler
## To store elsewhere: class=FileHandler
##
## Format and example:
## args=(filename, mode, encoding, delay)
## args=('audit.log', 'a', None, True)
##
##
##[handler_file]
##class=Products.ZenUtils.configlog.ZenFileHandler
##level=NOTSET
##formatter=file
##args=('audit.log', 'a', None, True)

## RotatingFile
##
## See http://docs.python.org/library/logging.handlers.html#rotatingfilehandler
##
## To store in ZENHOME/log:
class=Products.ZenUtils.configlog.ZenRotatingFileHandler
## To store elsewhere: class=handlers.RotatingFileHandler
##
## Format:
## args=(filename, mode, maxBytes, backupCount, encoding, delay)
##
## Example of one 10-meg file in ZENHOME/log/
## args=('audit.log', 'a', 10000000, 0, None, True)
##
## Example of ten 1-meg files in ZENHOME/log/audit/. The path must
## already exist.
## args=('audit/audit.log', 'a', 1000000, 10, None, True)
##
##
[handler_rotatingfile]
class=Products.ZenUtils.configlog.ZenRotatingFileHandler
level=NOTSET
formatter=file
args=('audit.log', 'a', 10485760, 3, None, True)

## TimedRotatingFile
##
## See http://docs.python.org/library/logging.handlers.html#timedrotatingfilehandler
##
## To store in ZENHOME/log:
class=Products.ZenUtils.configlog.ZenTimedRotatingFileHandler
## To store elsewhere: class=handlers.TimedRotatingFileHandler
##
## Format and example:
## args=(filename, when, interval, backupCount, encoding, delay, utc)
##

```

```

## Example of weekly log files for the past year in ZENHOME/log/audit/
## args=('audit/weekly.log', 'midnight', 7, 52, None, True, False)
##
##
##[handler_rotatingfile]
##class=Products.ZenUtils.configlog.ZenTimedRotatingFileHandler
##level=NOTSET
##formatter=file
##args=('audit/weekly.log', 'midnight', 7, 52, None, True, False)

## Console
##
## See http://docs.python.org/library/
logging.handlers.html#streamhandler
##
##
##[handler_console]
##class=StreamHandler
##level=NOTSET
##formatter=console
##args=(sys.stdout,)

##### String Formatters
##
## These must be uncommented if used by a handler above.
##
## See the very bottom of http://docs.python.org/library/
logging.config.html
##
##

##[formatter_syslog]
##format=zenoss[% (process)d]: %(message)s

[formatter_file]
format=%(asctime)s %(message)s
datefmt=%Y-%m-%d %H:%M:%S

##[formatter_console]
##format=Audit: %(asctime)s %(message)s
##datefmt=%H:%M:%S

```

After editing the `audit_log.conf` file, restart Zope with the command:

```
zenwebserver restart
```

## Examples

While enabled, user actions are tracked as specified by the configuration file.

### Example 1

```

Sep 12 12:55:10 zenoss[8432] user=hsolo action=SetDeviceClass
kind=Device device=/Devices/Server/Linux/devices/emailsrv05
deviceclass=/Devices/Server/SSH/Linux

```

In this example, user hsolo moved device "emailsrv05" from device class /Server/Linux to /Server/SSH/Linux.

## Example 2

```
Sep 12 12:57:19 zenoss[8432] user=lskywalker action=Edit
  kind=ThresholdClass maxval=100 minval=0 old_minval=-100
  thresholdclass="/Devices/Server/Linux/rrdTemplates/Device/thresholds/
CPU pct"
```

In this example, user lskeywalker edited values of threshold "CPU pct" by adding a max value of 100 and changing the min from -100 to 0.

## Searching File Content

You can use central logging tools or `grep` to parse the configuration file content. Using Splunk, for example, searching for "device=\*/localhost" finds any action on machines named localhost in any device class. Searching for "action=Add kind=User | table user username" creates a table that lists new users and which user added them.

## Utility

The `zensendaudit` utility allows you to log custom messages. For example:

```
zensendaudit Hello world.
```

generates the output:

```
Message sent: Hello world.
```

Further, it logs this message to the configured syslog or files:

```
zenoss[9350]: user=admin type=ManualEntry comment="Hello world."
```

## Stopping Audit Logging

Audit logging has minimal impact on system performance; however, you can disable audit logging by removing the `ZenPacks.zenoss.AuditLog` ZenPack. If re-installed, the system does not overwrite the existing configuration and example files, so that you can refer to the `audit_log.conf.example` file if needed.

## Setting Portlet Permissions

---

By setting permissions, you determine which users can view and interact with portlets. Permissions settings restrict which Zope Access Control List (ACL) can access each portlet.

Before you can successfully set portlet permissions, you must assign the user a specific Zenoss Core role. (You assign roles from the user edit page, from Advanced > Settings.) Each user role is mapped to one or more Zope ACL permissions, which allow you to restrict the portlets a permission level can see.

A user's specific portlet permissions are defined in part by Zope ACL permissions, and in part by the role to which he is assigned.

## User Role to ACL Mapping

The following table shows how user roles map to ACLs.

| User Roles                 | ACLPermission               |
|----------------------------|-----------------------------|
| ZenUser, ZenOperator       | ZenCommon, View             |
| ZenManager, Manager        | ZenCommon, View, Manage DMD |
| No Role, Administered Objs | ZenCommon                   |

## Setting Permissions

To set portlet permissions:

- 1 Select Advanced from the Navigation menu.

The Settings page appears.

- 2 In the left panel, select Portlets.

The Portlets page appears.

**Figure 99: Portlet Permissions**

| Available Portlets    | Permissions                        |
|-----------------------|------------------------------------|
| Device Issues         | Users with ZenCommon permission ▼  |
| Google Maps           | Users with View permission ▼       |
| Daemon Processes Down | Users with Manage DMD permission ▼ |
| Production States     | Users with ZenCommon permission ▼  |
| Site Window           | Users with View permission ▼       |
| Top Level Organizers  | Users with View permission ▼       |
| Messages              | Users with ZenCommon permission ▼  |
| Watch List            | Users with ZenCommon permission ▼  |

- 3 For one or more portlets in the Available Portlets list, select the permissions you want to apply.
- 4 Click **Save**.

## Troubleshooting: Users Cannot See All Portlets

You may mistakenly block users from being able to access some portlets. Often, this happens when a user has been set to see only particular devices. By default, this user will see only portlets set to the ZenCommon permission level. In effect, this blocks three of six portlets.

To remedy this problem, you can:

- Change the permission levels (on the Portlets page) to ZenCommon, or
- Change the user role to a role higher than "No Role."

## Working with the Job Manager

The Job Manager runs background tasks, such as discovering a network or adding a device. When you ask the system to perform one of these tasks, it adds a job to the queue. Jobs are run by the `zenjobs` daemon.

Not all actions are performed in the Job Manager. Some jobs are run automatically in the foreground. Others, such as moving devices, depend on user interface configuration settings.

When running jobs in the foreground, do not navigate away from the current page until the action completes.

## Viewing the Job Manager

To access the Job Manager:

- 1 From the Navigation menu, select **Advanced**.

The Settings page appears.

- 2 From the menu, select **Jobs**.

**Figure 100:** Job Manager

| Status  | Description                                  | Scheduled      | Started       | Finished      |
|---------|--|----------------|---------------|---------------|
| Success | Delete device test-win7-1.zenoss.lvc         | 10 minutes ago | 8 minutes ago | 8 minutes ago |
| Success | Move 6 devices to /zport/land/Devices/KVM    | 8 minutes ago  | 7 minutes ago | 7 minutes ago |
| Success | Delete 3 devices                             | 6 minutes ago  | 6 minutes ago | 6 minutes ago |
| Success | Add VMware infrastructure asxwin0.zenoss.lvc | 4 minutes ago  | 4 minutes ago | 4 minutes ago |
| Failure | Discover devices in network 10.175.208.0/22  | 4 minutes ago  | 4 minutes ago | 4 minutes ago |

**Job Log**

Log file: /Users/zenoss/dev/zenhome/log/jobs/506cbb0b-0722-4e08-a9fa-74e5c2c032fa.log

```

2012-05-10 13:51:10,782 INFO zen.Job: Beginning job
2012-05-10 13:51:16,782 INFO zen.Job: Spawning subprocess: /Users/zenoss/dev/zenhome/bin/zenclsc run --net 10.175.208.0/22
Traceback (most recent call last):
  File "", line 13, in
  File "", line 7, in create_raw_socket
  File "/Users/zenoss/dev/zenhome/lib/python2.7/socket.py", line 187, in __init__
    _sock = _realsocket(family, type, proto)
socket.error: [Errno 1] operation not permitted

```

No background jobs

The jobs list appears and shows information about all jobs currently in the system.

- **Status**- Shows the current job status. Status options are Pending (waiting for zenjobs to begin running), Running, Succeeded, and Failed.
- **Description**- Provides a description of the job.
- **Scheduled**- Shows when the job was scheduled to begin.
- **Started / Finished**- Provide information about the time period in which the job ran.
- **Created By**- User that created the job.

The lower section of the page displays the job log for the job selected in the list. You can view job info here, or by viewing the log file.

## Stopping and Deleting Jobs

To stop a job, select it in the list, and then click **Abort**. The zenjobs daemon will not run the job.

To remove a job from the system, select it and then click **Delete**.

## Configuring Jobs

You can determine, when moving devices, whether the action is performed immediately or as a job. By default, if you select five or more devices, the move action is performed as a job. To adjust this setting:

- 1 Select **Advanced > Settings**.
- 2 Select **User Interface** in the left panel.



- 3 Enter a value for Device Move Job Threshold, and then click **Save**.

## Running the zenjobs Daemon

You can stop and start the zenjobs daemon from the command line, and from **Advanced > Settings (Daemons selection)**.

## Host Name Changes

---

If you change the host name of your Zenoss Core server, then you must clear and rebuild queues before the zenhub and zenjobs daemons will restart.

To work around this issue, you can issue the following commands (although any data queued at restart time will be lost):

```
export VHOST="/zenoss"
export USER="zenoss"
export PASS="zenoss"
rabbitmqctl stop_app
rabbitmqctl reset
rabbitmqctl start_app
rabbitmqctl add_vhost "$VHOST"
rabbitmqctl add_user "$USER" "$PASS"
rabbitmqctl set_permissions -p "$VHOST" "$USER" '.*' '.*' '.*'
```

## Versions and Update Checks

---

You can check to see if there are newer versions of Zenoss Core available. By default, the system is configured to check for this each day; however, you can verify this at any time from **Advanced > Settings > Versions**. Click **Check Zenoss Version Now** to check for updates.

When the system performs its daily update check (enabled by the Check for Updates Daily option), by default it sends a utilization report containing non-identifying system metrics to Zenoss. Types of information sent about your installation include:

- Device count
- Number of systems, locations, and groups
- Installed ZenPacks
- Event count
- System statistics (such as operating system and version)
- Number of users
- Performance data (such as memory and file system space used)

To view a report of the collected data, click **Show Report**. To download an encrypted version of the utilization report, click **Save Encrypted**.



## SNMP Device Preparation

---

This section provides information about SNMP support and lists Net-SNMP configuration settings that are required by the system.

### 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 -v 2c -cpublic <your device name> system
```

- 2 Retrieve the IP table for the device with `snmpwalk`:

```
> snmpwalk -v 2c -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
```

### SNMP V3 Support

---

Zenoss Core provides support for SNMPv3 data collection.

The following configuration properties control the authentication and privacy of these requests:

- **zSnmpAuthType**- Use "MD5" or "SHA" signatures to authenticate SNMP requests.

If only `zSnmpAuthType` and `zSnmpAuthPassword` are set, then the message is sent with authentication but no privacy.

- **zSnmpAuthPassword**- Shared private key used for authentication. Must be at least 8 characters long.
- **zSnmpPrivType**- "DES" or "AES" cryptographic algorithms.

If `zSnmpPrivType` and `zSnmpPrivPassword` are set, then the message is sent with privacy and authentication.

You cannot set a PrivType and PrivPassword without also setting an AuthType and AuthPassword. If neither Priv nor Auth values are set, then the message is sent with no authentication or privacy.

- **zSnmprPrivKey**- Shared private key used for encrypting SNMP requests. Must be at least 8 characters long.
- **zSnmprSecurityName**- Security Name (user) to use when making SNMPv3 requests.

If monitoring SNMPv3 devices, make sure that msgAuthoritativeEngineID (also known as snmpEngineID or Engine ID) is not shared by two devices. It must be unique for each device.

## Advanced Encryption Standard

SNMPv3 encryption using the Advanced Encryption Standard (AES) algorithm is supported only if the host platform net-snmp library supports it.

You can determine if your platform supports AES by using the following test:

```
$ snmpwalk -x AES 2>&1 | head -1
```

If the response is:

```
"Invalid privacy protocol specified after -x flag: AES"
```

then your platform does not support AES encryption for SNMPv3.

If the response is:

```
"No hostname specified."
```

Then your platform supports AES.

## Community Information

---

Add these lines to your `snmp.conf` file.

This line will map the community name "public" into a "security name":

```
# sec.name source community
```

```
com2sec notConfigUser default public
```

This line will map the security name into a group name:

```
# groupName securityModel securityName
```

```
group notConfigGroup 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
```

## 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
```

## Extra Information

---

For more information, see the `snmpd.conf` manual page, and the output of the `snmpd -H` command.

```
trapcommunity public
```

```
trapsink default
```

# B

## Syslog Device Preparation

---

### Forwarding Syslog Messages from UNIX/Linux Devices

---

Zenoss Core has its own syslog server (zensyslog). Managed devices should point their syslog daemons to the system.

To do this, edit the `/etc/syslog.conf` file and add an entry, where 1.2.3.4 is the zensyslog IP:

- 1 Log on to the target device (as a super user).
- 2 Open the `/etc/syslog.conf` file with a text editor (such as vi).
- 3 Enter `*.debug`, and then press the Tab key.
- 4 Enter the host name or IP address of the server. For example:

```
*.debug @192.168.X.X
```

- 5 Save the file and exit the file editor program.
- 6 Restart the Syslog service using the command below:

```
/etc/init.d/syslog restart
```

### Forwarding Syslog Messages from a Cisco IOS Router

---

Here are some links to Cisco commands to turn on syslog. Typically, it is easier to use syslog than SNMP traps from network devices. The most basic IOS command to send syslog messages is:

```
logging 1.2.3.4
```

### 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 `conf` at the prompt.
- 5 Type the command `terminal` at the configuration prompt.

- At the prompt, Set the Syslog forwarding mechanism. See example below:

```
logging <IP address of the server>
```

- 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
```

## Forwarding Syslog Messages from a Cisco CatOS Switch

---

- Log on to the target switch.
- Type the command enable at the prompt.
- Enter the password when prompted.
- Set the Syslog forwarding mechanism; for example:

```
set logging server <IP address of the server>
```

- 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
```

## Forwarding Syslog Messages using Syslog-ng

---

Here is an example for FreeBSD and Linux platforms.

- Log on to the target device (as a super user)
- Open /etc/syslog-ng/syslog-ng.conf file with a text editor (e.g VI).
- Add source information to file. See example below:

FreeBSD:

```
source src { unix-dgram("/var/run/log"); internal ();};
```

Linux: (will gather both system and kernel logs)

```
source src { internal(); unix-stream("/dev/log" keep-alive(yes) max-
connections(100)); pipe("/proc/kmsg"); udp();};
```

- 4 Add destination information (in this case, the server). For example:

```
log { source(src); destination(zenoss); };
```

## C

## TALES Expressions

---

Use TALES syntax to retrieve values and call methods on Zenoss Core objects. Several areas accept TALES syntax; these include:

- Command templates
- User commands
- Notifications
- zLinks

Commands (those associated with devices and those associated with events) can use TALES expressions to incorporate data from the related devices or events. TALES is a syntax for specifying expressions that let you access the attributes of certain objects, such as a device or an event.

For additional documentation on TALES syntax, see the TALES section of the Zope Page Templates Reference:

<http://docs.zope.org/zope2/zope2book/AppendixC.html>

Depending on context, you may have access to a device, an event, or both. Following is a list of the attributes and methods you may want to use on device and event objects. The syntax for accessing device attributes and methods is `${dev/attributename}`. For example, to get the `manageIp` of a device you would use `${dev/manageIp}`. For events, the syntax is `${evt/attributename}`.

A command to ping a device might look like this. (The `${ . . }` is a TALES expression to get the `manageIp` value for the device.)

```
ping -c 10 ${device/manageIp}
```

## Examples

---

- DNS Forward Lookup (assumes device/id is a resolvable name)

```
host ${device/id}
```

- DNS Reverse Lookup

```
host ${device/manageIp}
```



- SNMP Walk

```
snmpwalk -v 2c -c${device/zSnmpCommunity} ${device/manageIp} system
```

To use these expressions effectively, you must know which objects, attributes, and methods are available, and in which contexts. Usually there is a device that allows you to access the device in a particular context. Contexts related to a particular event usually have event defined.

## TALES Device Attributes

The following table lists available device attributes.

| Attribute                   | Description   |
|-----------------------------|---|
| getId                       | The primary means of identifying a device within the system   |
| getManageIp                 | The IP address used to contact the device in most situations  |
| productionState             | The production status of the device: Production, Pre-Production, Test, Maintenance or Decommissioned. This attribute is a numeric value, use getProductionStateString for a textual representation. |
| getProductionStateString    | Returns a textual representation of the productionState   |
| snmpAgent                   | The agent returned from SNMP collection   |
| snmpDescr                   | The description returned by the SNMP agent  |
| snmpOid                     | The oid returned by the SNMP agent  |
| snmpContact                 | The contact returned by the SNMP agent  |
| snmpSysName                 | The system name returned by the SNMP agent  |
| snmpLocation                | The location returned by the SNMP agent   |
| snmpLastCollection          | When SNMP collection was last performed on the device. This is a DateTime object.   |
| getSnmpLastCollectionString | Textual representation of snmpLastCollection  |
| rackSlot                    | The slot name/number in the rack where this physical device is installed  |
| comments                    | User entered comments regarding the device  |
| priority                    | A numeric value: 0 (Trivial), 1 (Lowest), 2 (Low), 3 (Normal), 4 (High), 5 (Highest)  |
| getPriorityString           | A textual representation of the priority  |
| getHWManufacturerName       | Name of the manufacturer of this hardware   |
| getHWProductName            | Name of this physical product   |
| getHWProductKey             | Used to associate this device with a hardware product class   |
| getOSManufacturerName       | Name of the manufacturer of this device's operating system.   |
| getOSProductName            | Name of the operating system running on this device.  |
| getOSProductKey             | Used to associate the operating system with a software product class  |
| getHWSerialNumber           | Serial number for this physical device  |
| getLocationName             | Name of the Location assigned to this device  |
| getLocationLink             | Link to the system page for the assigned Location   |

| Attribute             | Description   |
|-----------------------|---|
| 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       |

## Tales Event Attributes

The following table lists available event attributes.

| Attribute     | Description  |
|---------------|--|
| agent         | Collector name from which the event came (such as zensyslog or zentrap).   |
| component     | Component of the associated device, if applicable. (Examples: eth0, httpd.)  |
| count         | Number of times this event has been seen.  |
| dedupid       | Key used to correlate duplicate events. By default, this is: device, component, eventClass, eventKey, severity.                    |
| device        | ID of the associated device, if applicable.  |
| DeviceClass   | Device class from device context.  |
| DeviceGroups  | Device systems from device context, separated by  .  |
| eventClass    | Event class associated with this device. If not specified, may be added by the rule process. If this fails, then will be /Unknown. |
| eventClassKey | Key by which rules processing begins. Often equal to component.  |
| eventGroup    | Logical group of event source (such as syslog, ping, or nteventlog).   |
| eventKey      | Primary criteria for mapping events into event classes. Use if a component needs further de-duplication specification.             |
| eventState    | State of event. 0 = new, 1 = acknowledged, 2 = suppressed.   |
| evid          | Unique ID for the event.   |
| facility      | syslog facility, if this is a syslog event.  |
| firstTime     | UNIX timestamp when event is received.   |
| ipAddress     | IP Address of the associated device, if applicable.  |
| lastTime      | Last time this event was seen and its count incremented.   |
| Location      | Device location from device context.   |
| manager       | Fully qualified domain name of the collector from which this event came.   |
| message       | Full message text.   |

| Attribute   | Description  |
|-------------|--|
| nteventid   | nt event ID, if this is an nt eventlog event.                                  |
| priority    | syslog priority, if this is a syslog event.                                    |
| prodState   | prodState of the device context.   |
| severity    | One of 0 (Clear), 1 (Debug), 2 (Info), 3 (Warning), 4 (Error) or 5 (Critical). |
| stateChange | Time the MySQL record for this event was last modified.                        |
| summary     | Text description of the event. Limited to 150 characters.                      |
| suppid      | ID of the event that suppressed this event.                                    |
| Systems     | Device systems from device context, separated by  .                            |

### Configuration Properties and Custom Properties

Configuration properties and custom properties also are available for devices, and use the same syntax as shown in the previous sections.

# Glossary of terms

## aggregation pools

---

A logical bundling of multiple physical network interfaces, commonly known as a port channel. For example, the Per Chassis Ethernet Pools includes all links from all chassis to all fabric interconnects which is used for chassis bandwidth balance comparison. For more examples, see the Aggregation Pools component section of CiscoUCS devices.

## bandwidth utilization

---

The total amount of bandwidth being used by an aggregation pool, a port, a fabric interconnect, a FEX, etc.

## component

---

Object contained by a device. Components include interfaces, OS processes, file systems, CPUs, and hard drives.

## data point

---

Data returned from a data source. In many cases, there is only one data point for a data source (such as in SNMP); but there may also be many data points for a data source (such as when a command results in the output of several variables).

## data source

---

Method used to collect monitoring information. Example data sources include SNMP OIDs, SSH commands, and perfmon paths.

## device

---

Primary monitoring object in the system. Generally, a device is the combination of hardware and an operating system.

## device class

---

Special type of organizer used to manage how the system models and monitors devices (through configuration properties and monitoring templates).

## discovery

---

Process by which Zenoss Core gathers detailed information about devices in the infrastructure. Results of discovery are used to populate the model.

## event

---

Manifestation of important occurrence within the system. Events are generated internally (such as when a threshold is exceeded) or externally (such as through a syslog message or SNMP trap).

## event class

---

Categorization system used to organize event rules.

## event rules

---

Controls how events are manipulated as they enter the system (for example, changing the severity of an event). Configuration properties configure event rules.

## graph

---

Displays one or more data points, thresholds, or both.

## headroom

---

The unused bandwidth in an aggregation pool, a port, a fabric interconnect (FI), a FEX, etc. For example, if an aggregation pool including 4 ports between a chassis and the FIs has 40 GB of capacity and if the bandwidth use of that pool is 25 GB, then the headroom is 15 GB.

## integrated infrastructure

---

A bundle of compute, storage, networking, and virtualization components. Most integrated infrastructures are bought as one from a vendor:

- NetApp FlexPod
- VCE Vblock
- EMC VSPEX

All of these have UCS as the common compute element, Nexus as networking components, and VMware as virtualization.

## managed resource

---

Servers, networks, virtual machines, and other devices in the IT environment.

## model

---

Representation of the IT infrastructure. The model tells the system "what is out there" and how to monitor it.

## monitoring template

---

Description of what to monitor on a device or device component. Monitoring templates comprise four main elements: data sources, data points, thresholds, and graphs.

## notification

---

Sends email or pages to system users or groups when certain events occur.

## organizer

---

Hierarchical system used to describe locations and groups. Zenoss Core also includes special organizers, which are classes that control system configuration.

## **out of balance**

---

Indicates that the bandwidth use is quite different among the ports in an aggregation pool. This can often be corrected by reconfiguration, for example, by moving a service profile from one chassis to another.

## **resource component**

---

Interfaces, services and processes, and installed software in the IT environment.

## **service profile**

---

A service profile is a software definition of a server and its LAN and SAN network connectivity, in other words, a service profile defines a single server and its storage and networking characteristics.

## **threshold**

---

Defines a value beyond which a data point should not go. When a threshold is reached, the system generates an event. Typically, threshold events use the `/Capacity` event class.

## **trigger**

---

Determines how and when notifications are sent. Specifies a rule comprising a series of one or more conditions.