LogMatrix®

NerveCenter™

Learning to Create Behavior Models

Version 6.2

LogMatrix, Inc.
230 N. Serenata Drive, Suite 711
Ponce Vedra Beach, FL 32082 USA

Toll Free +1 (800) 892-3646
Phone +1 (508) 597-5300
Fax +1 (774) 348-4953

info@logmatrix.com
http://www.logmatrix.com

Table of Contents

C

Introduction

1

Learning to Create Behavior Models is a tutorial that introduces how to design and create behavior models. It introduces concepts and procedures detailed fully in the book Designing and Managing Behavior Models.

NerveCenter Documentation

This section describes the available NerveCenter documentation, which explains important concepts in depth, describes how to use NerveCenter, and provides answers to specific questions.

The documentation set is provided in online (HTML) format, as well as PDF for printing or on-screen viewing.

Using the Online Help

You can view the documentation with Google Chrome, Mozilla Firefox, Apple Safari, or Microsoft Edge. Refer to the NerveCenter Release Notes for the browser versions supported with this release.

Note: For in-depth instructions on using the online documentation, click the Help button ? in the upper right of the Help window.

Printing the Documentation

The NerveCenter documentation is also available as Portable Document Format (PDF) files that you can open and print. All PDF files are located in your `*installpath*/doc` directory.

Note: You must have Adobe Acrobat Reader to open or print the PDF files. You can download the Reader free from Adobe's Web Site at www.adobe.com.

The NerveCenter Documentation Library

The following documents ship with NerveCenter.

Book Title	Description	Application	Audience	PDF for Print
NerveCenter Release Notes	Describes new NerveCenter features and includes late-breaking information, software support, corrections, and instructions.	All	All	relnotes.pdf
Installing NerveCenter	Helps you plan and carry out your NerveCenter upgrades and new installations. Use the *Release Notes* in conjunction with this book.	All	Installation team	install.pdf
Managing NerveCenter	Explains how to customize and tune NerveCenter after it has been installed.	NerveCenter Administrator	Administrator	managing_ nervecenter.pdf
NerveCenter Platform Integration Guide	Explains how to integrate NerveCenter with network management platforms.	NerveCenter Administrator	Administrator	integratingNC. pdf
Learning to Create Behavior Models	Provides step-by-step instructions and examples for creating behavior models.	NerveCenter Client	Users with administrative privileges	learningModel. pdf
Designing and Managing Behavior Models	Explains behavior models in depth, how to create or modify models, and how to manage your models.	NerveCenter Client	Users with administrative privileges	designingModels.pdf
Monitoring Your Network	Explains how NerveCenter works and how you can most effectively monitor your network.	NerveCenter Client	Users	monitoringNet. pdf

UNIX Systems

On UNIX systems, NerveCenter man pages provide command reference and usage information that you view from the UNIX shell as with other system man pages. When you specify documentation during NerveCenter installation, the script installs nroff-tagged man pages and updates your system’s MANPATH environment variable to point to the NerveCenter man page directory.

Document Conventions

This document uses the following typographical conventions:

Element	Convention	Example
Key names, button names, menu names, command names, and user entries	**Bold**	Press **Tab** Enter **ovpa -pc**
■ A variable you substitute with a specific entry ■ Emphasis ■ Heading or Publication Title	*Italic*	Enter **./installdb -f** *IDBfile*
Code samples, code to enter, or application output	`Code`	`iifInOctets > 0`
Messages in application dialog boxes	*Message*	*Are you sure you want to delete?*
An arrow (>) indicates a menu selection	**>**	Choose **Start > Programs > NerveCenter**

Caution: A caution warns you if a procedure or description could lead to unexpected results, even data loss, or damage to your system. If you see a caution, proceed carefully.

Note: A note provides additional information that might help you avoid problems, offers advice, and provides general information related to the current topic.

Documentation Feedback

LogMatrix, Inc. is committed to providing quality documentation and to helping you use our products to the best advantage. If you have any comments or suggestions, please send your documentation feedback to:

Documentation
LogMatrix, Inc.
230 N. Serenata Drive, Suite 711
Ponce Vedra Beach, FL 32082 USA

support@logmatrix.com

LogMatrix Technical Support

LogMatrix is committed to offering the industry's best technical support to our customers and partners. You can quickly and easily obtain support for NerveCenter, our proactive IT management software.

Professional Services

LogMatrix offers professional services when customization of our software is the best solution for a customer. These services enable us, in collaboration with our partners, to focus on technology, staffing, and business processes as we address a specific need.

Educational Services

LogMatrix is committed to providing ongoing education and training in the use of our products. Through a combined set of resources, we can offer quality classroom style or tailored on-site training.

Contacting the Customer Support Center

For Telephone Support

Phone: Toll Free +1 (800) 892-3646 or Phone +1 (508) 597-5300

For E-mail Support

E-mail: support@logmatrix.com.

How to Start Using NerveCenter Client

2

This book will give you hands-on experience designing and using NerveCenter behavior models. You should be completing the activities and exercises on an actual NerveCenter Client.

This chapter explains:

- What the NerveCenter Client is
- How to start and exit the NerveCenter Client
- How to connect to a NerveCenter Server
- How to disconnect from a NerveCenter Server

Before You Begin

Learning to Create Behavior Models is a tutorial for introducing you to some of the most commonly used features of NerveCenter.

The activities and exercises in this tutorial are self-paced and hands-on. They will give you the knowledge and skills necessary to design and manage NerveCenter behavior models.

Note: The activities and exercises depend on behavior models that you create and modify in previous chapters. Therefore it is important that you complete each section and each exercise in the order presented. Also, be sure to save your work at the end of each chapter.

Although you will be working in your network's real environment, you don't need to be worried about causing too much traffic or "messing things up." The activities in "How to Define Property Groups and Properties" on page 15 will help you to isolate a small subset of devices on your network. The rest of the activities will involve only those devices. The last chapter, "How to Reset Your Environment" on page 115 will step you through setting your network back to normal.

Note: To fully understand and appreciate the information provided in this tutorial you'll probably need a couple of days.

The answers to the review questions are listed in the appendix "Answers to Questions" on page 121.

What You Need to Know

Whether you are a network operator, administrator, or manager, you should be familiar with the following topics to get the most out of this tutorial:

- Computer networking concepts
- Simple Network Management Protocol (SNMP) basics
- Common MIB-II objects and related terminology
- Your network management platform (if applicable)

Note: Although you do not need to know Perl to use NerveCenter, several NerveCenter features can be enhanced and customized by Perl scripts. Knowledge of Perl is not necessary to complete this tutorial.

What You Need to Use

The activities and exercises in this tutorial make extensive use of the NerveCenter Client. Before you can begin, you must have

- A NerveCenter Client installed on your local Windows machine
- A NerveCenter Server installed on a machine accessible by your local machine.
- A NerveCenter administrator should configure the NerveCenter Server before you begin this tutorial. Where applicable the following items should be configured:
  - Inform recipients
  - Paging information
  - SMTP mail server
  - A NerveCenter administrator user name and password for you to use
- At least three devices running SNMP agents
- Access to your network management platform (if applicable)

And, optionally:

- A supported web browser such as Microsoft Edge, Google Chrome, or Mozilla Firefox on your local Windows machine
- An office suite such as Microsoft Office or LibreOffice installed on your local Windows machine

What is the NerveCenter Client?

NerveCenter is a distributed client/server application. It includes the following components:

- The NerveCenter Server
- The NerveCenter database
- NerveCenter user interfaces

The NerveCenter Client, which you will be using in this tutorial, is one of the NerveCenter user interfaces.

Architectural Components

The NerveCenter *Server* performs all the NerveCenter monitoring and processing. The NerveCenter Server also saves to the NerveCenter database all the behavior model components you modify or create as you use NerveCenter to manage your network.

The NerveCenter *database* is primarily a repository for all the objects found in a NerveCenter behavior model. On UNIX systems, the database is stored in a flat file.

The NerveCenter *user interfaces* provide the user the ability to view and control NerveCenter processes. The three primary user interfaces are:

- The NerveCenter Administrator
- A command line interface
- The NerveCenter Client

User Interfaces

The NerveCenter *Administrator* is used to configure NerveCenter once it is installed. The *command line interface* can be used to perform a limited number of operations on NerveCenter objects.

The NerveCenter *Client* is used to monitor a network for problems and to create new behavior models. The Client runs in either administrator or operator mode, depending on your user ID or group.

To complete the activities and exercises in this book, you must be able to connect with either the NerveCenter Administrator or Client applications to the NerveCenter Server. To do this, you must know the login and password credentials for an account on the NerveCenter Server host, where the referenced user account is a member of the NerveCenter administrators (ncadmins) group. Contact the administrator of your Linux host if you need assistance setting this up.

Table 1: User and Administrator Login Rights in NerveCenter Client

	User	Administrator
Monitor active alarms	✓	✓
View an alarm’s history	✓	✓
Reset alarms	✓	✓
Monitor the state of managed nodes	✓	✓
Generate reports	✓	✓
Create new behavior models		✓
Customize the predefined behavior models		✓
Modify, copy, or delete an object in the NerveCenter database		✓
Assign rights to other NerveCenter users		✓

How to Use the NerveCenter Client

In this scenario, you will want to open a client interface and connect to a server.

This scenario includes the following three activities:

1. Starting the NerveCenter Client below
2. Connecting to a NerveCenter Server on the facing page
3. Quitting the NerveCenter Client on page 11

Starting the NerveCenter Client

TO START THE NERVECENTER CLIENT

- Select the **Start** menu.Select **Programs > LogMatrix NerveCenter > Client**.

  NerveCenter displays the NerveCenter Client window.

Note: These steps depend on a typical NerveCenter installation. The directory path may be different.

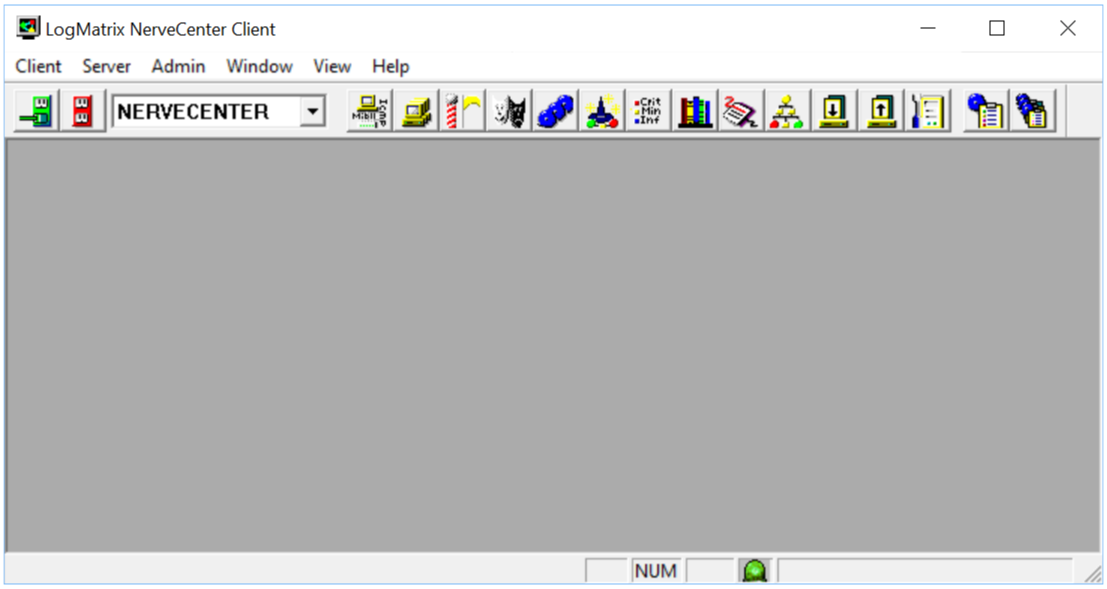

LogMatrix NerveCenter Client
Client Server Admin Window View Help
NERVECENTER
NUM

Figure 1: The NerveCenter Client Window

Most buttons and menu options are not enabled until you connect the client to a NerveCenter server.

Connecting to a NerveCenter Server

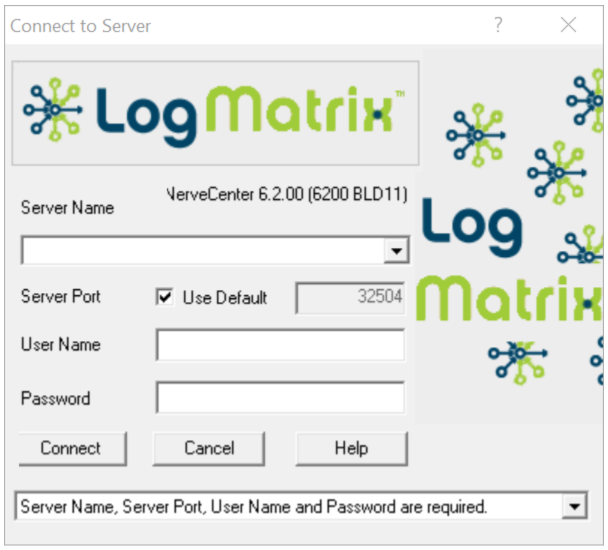
In the last activity, you opened the NerveCenter Client. However, there's not much you can do until you connect to a NerveCenter Server.

This next activity will step you through the process of connecting to a NerveCenter Server.

To connect to a NerveCenter Server

1. Make sure a NerveCenter Server is running on a host that your local machine can access. If you are having trouble, see your NerveCenter administrator.
2. In the NerveCenter Client, choose **Connect** from the **Server** menu.

   NerveCenter displays the Connect to Server window.



Server Port: ☑ Use Default 32504

Server Name, Server Port, User Name and Password are required.

3. In the **Server Name** field, type the host name or the IP address of the machine running a NerveCenter Server.

   The first time you connect to a server, the drop-down list box will be empty. After entering the name once, you can select the server's name from the drop-down list. The drop-down list box contains all the machines to which you've connected, or attempted to connect, in the past.

4. In the **User ID** field and **Password** fields, type a valid user name and a valid password.

   Remember, to perform the activities and exercises in this book, the account credentials you supply must be for a member of the NerveCenter Administrators group (ncadmins) on the host where the NerveCenter Server is running.

5. In the Connect to Server window, select **Connect**.

   The NerveCenter Client connects to the NerveCenter Server. The NerveCenter Client displays the active server as well as the Aggregate Alarm Summary window.

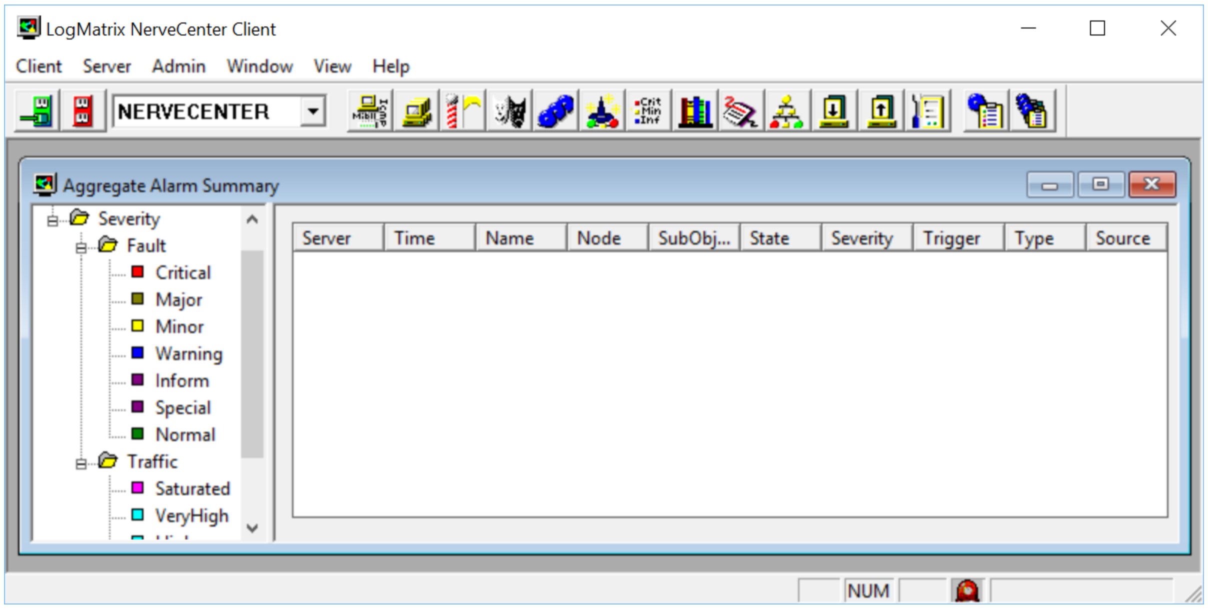


Figure 2: The NerveCenter Client Window after the User Has Connected to a Server

You have just connected to an active NerveCenter Server. There may be times when you want to disconnect from a server. The next activity will explain how.

Quitting the NerveCenter Client

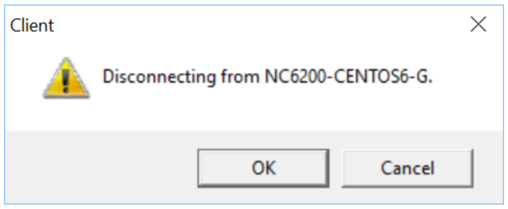
In the previous two activities you opened the NerveCenter Client and connected to a NerveCenter Server. At some point it may be necessary to disconnect from a server and/or quit the NerveCenter Client.

This last activity will step you through the process of disconnecting from the server and exiting the NerveCenter Client.

To quit the NerveCenter Client

1. From the **Server** menu, choose **Disconnect**.

   NerveCenter displays a warning message that it is about to disconnect from the active server.



Disconnecting from NC6200-CENTOS6-G.

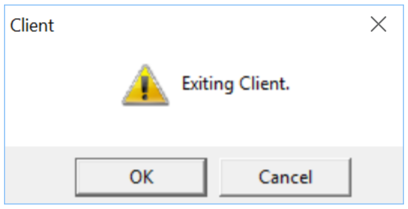
2. In the warning message, select **OK**.

   The NerveCenter Client disconnects from the active server.

3. From the **Client** menu, choose **Exit**.

   NerveCenter displays a warning message that you are about to exit the application.

Note: It is not necessary to first disconnect from the NerveCenter Server before exiting. Exiting the NerveCenter Client effectively disconnects from the Server as well as closes the Client.



Exiting Client.

4. In the warning message, select **OK**.

Note: When you exit the NerveCenter Client, all connections to NerveCenter Servers are broken. The NerveCenter Server continues to run.

You have just completed disconnecting from the active server and exiting the NerveCenter Client.

You now know how to start the NerveCenter Client and connect to a NerveCenter Server. You also can disconnect from a NerveCenter Server as well as quit the NerveCenter Client.

You are almost ready to begin creating behavior models. First, though, you will want to select some devices on your network so you can create a test group to work with. "How to Define Property Groups and Properties" on page 15 will explain how to do this.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter:

1. Open the NerveCenter Client.
2. Connect to a NerveCenter Server.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 2 Review Questions on page 121.

1. What are the three components of the NerveCenter client/server architecture?

2. If you have administrator privileges, what can you do with the NerveCenter Client?

Summary of What You Learned

In this chapter you learned how to start the NerveCenter Client as well as connect to an active NerveCenter Server. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to open the NerveCenter Client
- How to connect to a NerveCenter Server
- How to disconnect from a NerveCenter Server
- How to quit the NerveCenter Client
- You also were introduced to the following concepts:
- NerveCenter’s client/server architecture
- The NerveCenter Client
- The differences between user and administrator privileges in the NerveCenter Client

How to Define Property Groups and Properties

3

As you work through the exercises in this book, you will want to identify a set of devices to use. This can be a small set of devices such as printers, switches, and routers on your network, which your network administrator can identify for you. They each need to have configured and enabled SNMP Agents and you need to know the credentials needed to access each of these devices using SNMP.

With NerveCenter, you target the product's monitoring activities to a particular set of nodes. This is done by assigning them an appropriate property group.

This chapter explains:

- What are property groups and properties
- How to create a new property group
- How to create a new property
- How to assign a property group to a particular set of nodes

What is a Property Group?

Often you will create within NerveCenter a behavior model that monitors behavior on your entire network without a concern for the specific type of node causing the behavior. At times, however, you will want to limit a behavior model's activity to monitoring a specific node or type of nodes. You can achieve this limiting of a behavior model's focus by using property groups.

You assign each node in NerveCenter's node list to a prepared property group. A property group is simply a labeled set of properties, which can be either:

- The name of a Managed Information Base (MIB) base object
- A user-defined string

You can limit any object in a behavior model to focus on one specific property.

For example, if you wish to poll only a certain set of routers, you would assign to each router a property group with a unique property. When defining the poll, you would tell it to poll only those nodes which have the unique property.

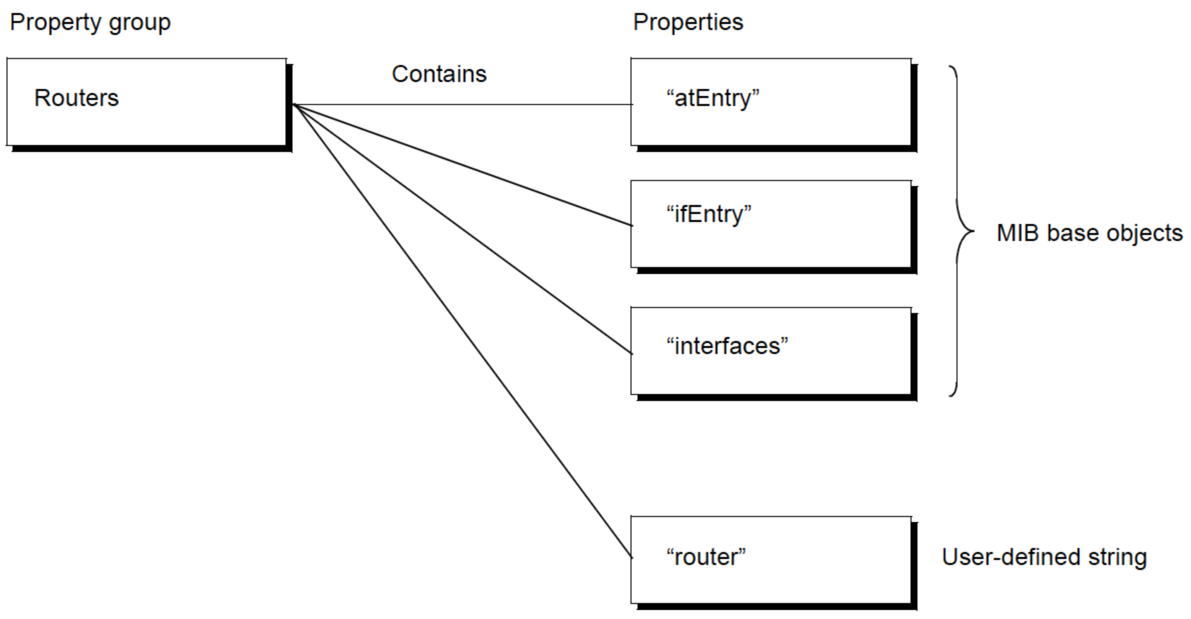


Figure 3: Property Groups and Properties

How to Create Property Groups and Properties

Throughout this tutorial, you will want to limit the activities of NerveCenter to a particular set of nodes. This will reduce the impact of the exercises on the network as well as make the information more manageable.

This scenario includes the following three activities:

1. Creating a New Property Group on the facing page
2. Creating a New Property on page 19
3. Assigning a Property Group to a Set of Nodes on page 20

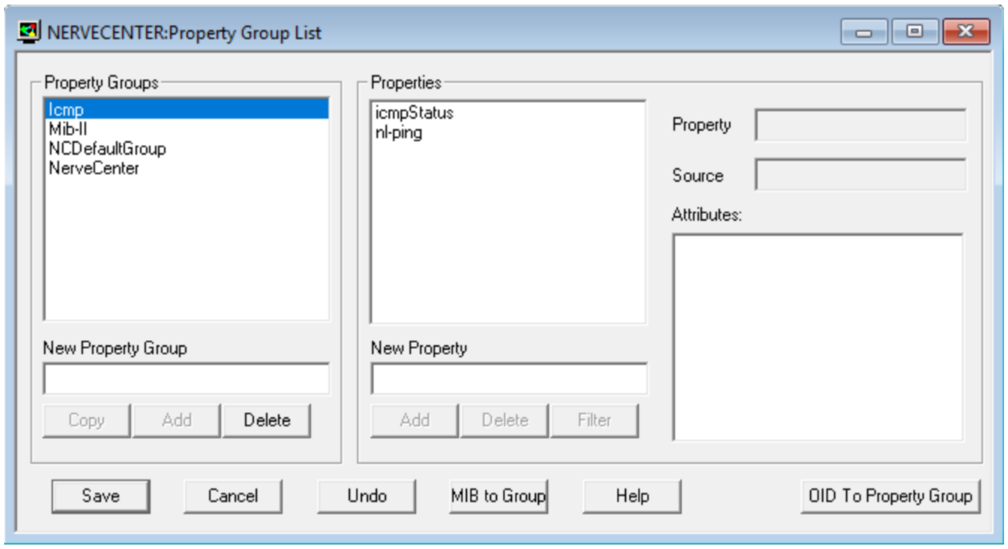
Creating a New Property Group

This first activity will step you through the process of creating a unique property group called **CriticalDevices**.

TO CREATE A PROPERTY GROUP

1. Open the NerveCenter Client and connect to the appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Property Group List**.

   NerveCenter displays the Property Group List window.



   The Property Group List window contains summary information about all property groups and properties in the NerveCenter database for the active server. From this window you can open existing property groups or create new ones.

3. Scroll through the **Property Group** list until you see the property group called **Mib-II**.

Note: Within the list box you can just begin typing the desired name until it displays. Most list boxes in NerveCenter contain this feature.

4. Select **Mib-II**.

   The properties belonging to the Mib-II property group display to the right in the Property list.

   Rather than create an entire new set of properties for our new property group, we will use the properties already present.

5. In the New Property Group field, type the name of your new property group **CriticalDevices**.

6. Under the **New Property Group** field, select **Copy**.

   CriticalDevices now appears in the list of property groups and is highlighted. Notice also the **Property** list to the right contains a list of properties.

   The **Copy** button creates a new property group that contains the same properties as the selected group. In this case, NerveCenter created a new property group called **CriticalDevices** containing the same properties as the group Mib-II.

7. Select **Save**.

Now that you have created a new property group, it is time to create a new property that will enable NerveCenter to target just those devices that interest you.

What is a property?

In this next activity you will be creating a property.

A *property* is a string. This string can be:

- The name of a MIB base object
- A user-defined string

This string determines (in part) if NerveCenter will interact with a particular node. A property group is a named container for properties.

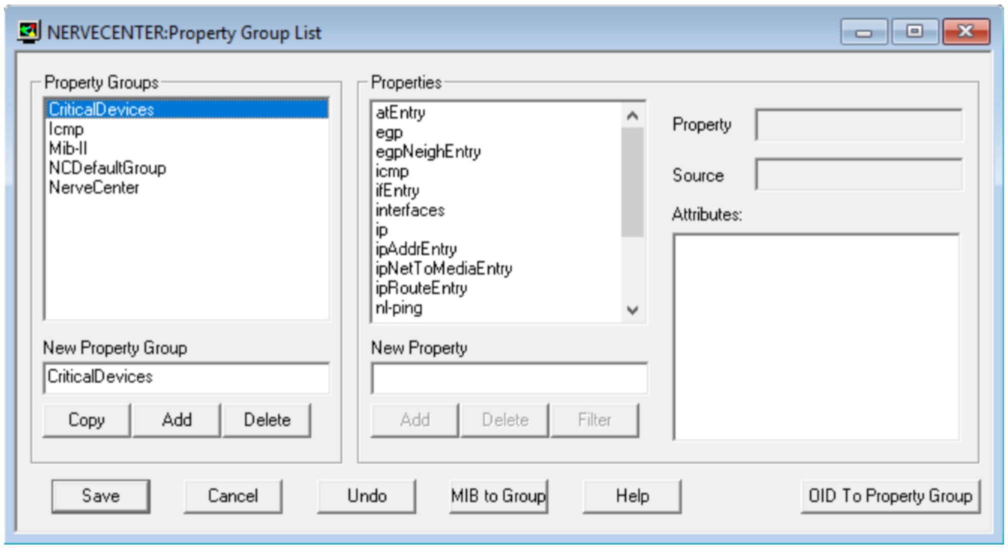
Creating a New Property

This next activity will step you through the process of creating within the property group **CriticalDevices** a new property called myNodes.

TO CREATE A NEW PROPERTY

1. In the **Property Group** list, select **CriticalDevices**.

   NerveCenter displays all the properties for this property group in the **Property** list.



New Property Group: CriticalDevices

2. In the **New Property** text box, type in the name of your new property **myNodes**.
3. Select **Add**.

   The **myNodes** property now appears in the list of properties for the **CriticalDevices** property group.

4. Select **Save**.
5. Select **Cancel** to close the Property Group List window.

Now that you have created the unique property **myNodes**, it is time to assign that property to the correct devices. The next activity will step you through that process.

Assigning a Property Group to a Set of Nodes

This next activity will step you through the process of centering NerveCenter's focus on the devices you will be monitoring throughout the rest of this book. You will achieve this by assigning the **CriticalDevices** property group to a particular set of nodes.

In our example network we would want to isolate the critical routers. For this tutorial, you will be isolating three of your department's managed workstations.

TO ASSIGN A PROPERTY GROUP TO A SET OF NODES

1. Identify at least three of your department's network devices (e.g., printers, workstations, switches) that you know are running SNMP agents and write their names on the following lines:

   ____

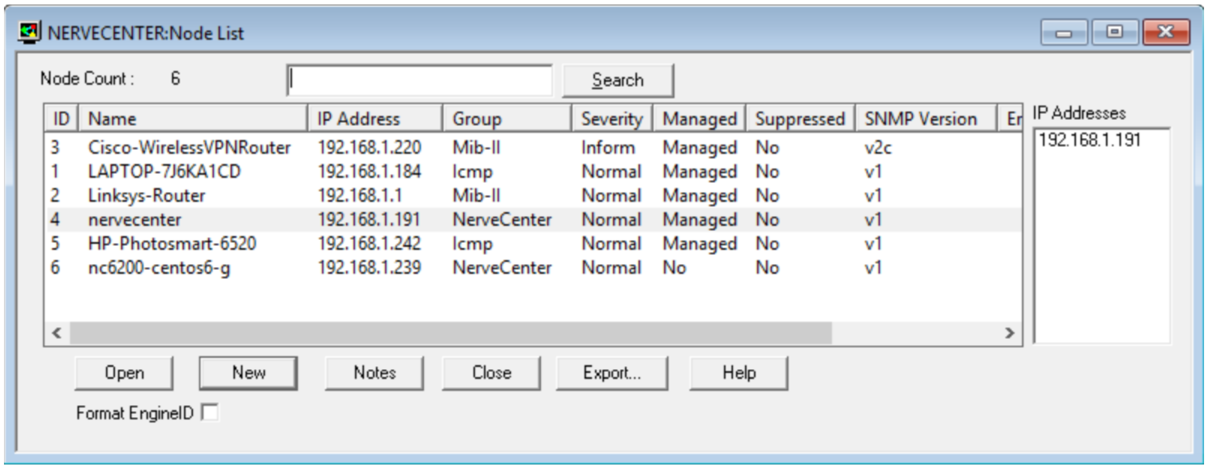
   ____

   ____

   If you have questions about which devices to use, see Before You Begin on page 5.

2. From the **Admin** menu, choose **Node List**.

   NerveCenter displays the Node List window.

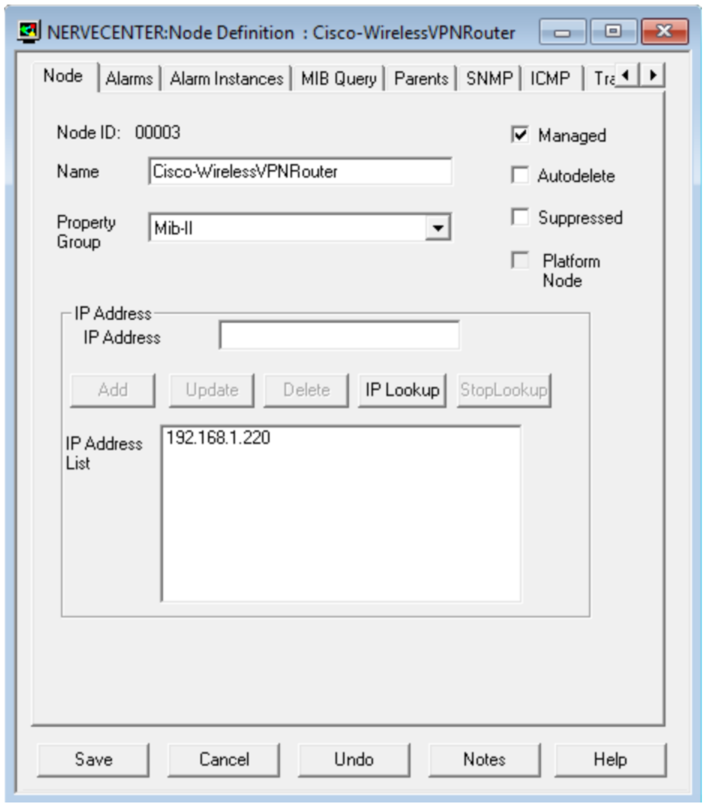


ID	Name	IP Address	Group	Severity	Managed	Suppressed	SNMP Version	Er
3	Cisco-WirelessVPNRouter	192.168.1.220	Mib-II	Inform	Managed	No	v2c	
1	LAPTOP-7J6KA1CD	192.168.1.184	Icmp	Normal	Managed	No	v1	
2	Linksys-Router	192.168.1.1	Mib-II	Normal	Managed	No	v1	
4	nervecenter	192.168.1.191	NerveCenter	Normal	Managed	No	v1	
5	HP-Photosmart-6520	192.168.1.242	Icmp	Normal	Managed	No	v1	
6	nc6200-centos6-g	192.168.1.239	NerveCenter	Normal	No	No	v1	

The Node List window lists all nodes in the NerveCenter database. This window provides the tools for identifying nodes of interest, changing their configuration, or opening a node's associated definition window.

3. Scroll through the node list until you see one of the devices you wish to monitor in these exercises. After highlighting that device, select **Open**.

   NerveCenter displays that device’s Node Definition window.



Node ID: 00003
Name: Cisco-WirelessVPNRouter
Property Group: Mib-II
[✓] Managed
[ ] Autodelete
[ ] Suppressed
[ ] Platform Node
IP Address
IP Address:
IP Address List: 192.168.1.220

The Node Definition window enables you to edit various elements of a managed device, such as its name, IP address, community string, deletion strategy, and property group.

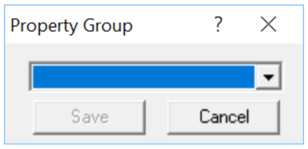
4. In the **Property Group** list, select the **CriticalDevices** property group.
5. Select **Save**.
6. To close the window, select **Cancel**.

7. In the Node List window, hold the **Ctrl** key and select the other devices you wish to monitor.

   Each node that is selected is highlighted. If you inadvertently select an unwanted node, select it again so that it is no longer highlighted.

8. Right-click one of the highlighted devices and select **Property Group**.

   The Property Group window displays.



9. In the Property Group window, select the **CriticalDevices** property group and click **Save**.
10. In the Node List window, select **Close**.

You have just assigned the unique property group **CriticalDevices** to a particular set of nodes. This will allow you to isolate only relevant devices as you work through the rest of this book. Now you are ready to begin using NerveCenter. "How to Use Polls" on page 25 will illustrate how to use NerveCenter to proactively monitor your **CriticalDevices** nodes.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter:

1. Create a new property group to identify devices that are giving the network difficulty. Some of the actions you will take include:
   a. Naming the property group **Troublemakers**
   b. Copying the MIB-II properties
   c. Creating a new property called **trouble**
2. Assign the property group Troublemakers to the **CriticalDevices** nodes on your network. Then return those devices to the **CriticalDevices** property group.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 3 Review Questions on page 121.

1. What is the purpose of property groups?

2. What are the two types of properties?

Summary of What You Learned

In this chapter you learned how to create and assign property groups and properties. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to create a new property group
- How to create a new property
- How to assign a property group to a particular set of nodes
- You also were introduced to the following concepts:
- Property groups
- Properties

How to Use Polls

4

In the last chapter you learned how to use property groups to tell NerveCenter to monitor a particular set of nodes. Polling is one of the ways NerveCenter can monitor these nodes.

This chapter explains:

- What a poll is and how it fits into a behavior model
- How to create a new poll
- How to write a poll condition
- How to enable a poll
- How to modify a poll condition

What is a Poll?

NerveCenter allows you to proactively monitor conditions on your network through the use of polls. NerveCenter polls retrieve information from SNMP agents on devices to determine the status of nodes.

Whenever NerveCenter polls a particular node, it should receive a response. If the poll detects a particular set of conditions from the node, it generates a trigger that will then cause an alarm to transition to another state.

review - no illustration?

Figure 4: The Role of a Poll within a Behavior Model

How to Create a Poll

In this scenario, you want to know which of the CriticalDevices nodes are experiencing high traffic. To accomplish this, you must create and enable a poll to check for this condition.

This scenario includes the following four activities:

1. Creating a New Poll on the next page
2. Writing a Poll Condition on page 28
3. Enabling a Poll on page 32
4. Modifying a Poll Condition on page 33

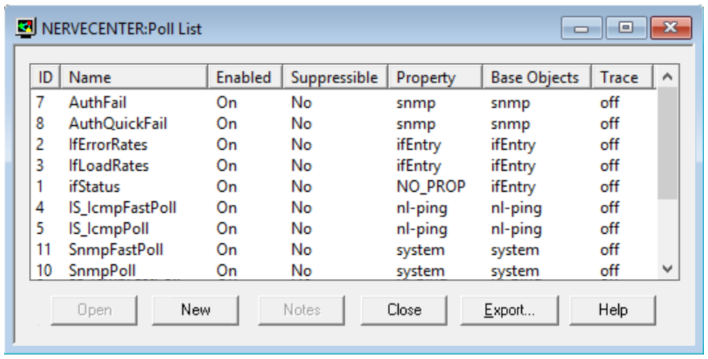
Creating a New Poll

This first activity will step you through the first stage of the process of creating a poll that checks for high traffic.

TO CREATE A NEW POLL

1. Open the NerveCenter Client and connect to the appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Poll List**.

   NerveCenter displays the Poll List window.

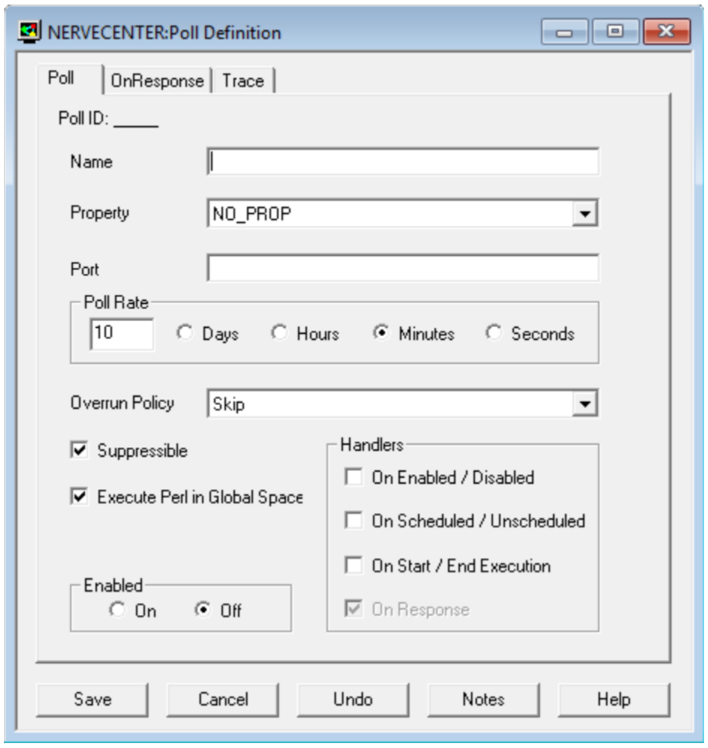


ID	Name	Enabled	Suppressible	Property	Base Objects	Trace
7	AuthFail	On	No	snmp	snmp	off
8	AuthQuickFail	On	No	snmp	snmp	off
2	IfErrorRates	On	No	ifEntry	ifEntry	off
3	IfLoadRates	On	No	ifEntry	ifEntry	off
1	ifStatus	On	No	NO_PROP	ifEntry	off
4	IS_IcmpFastPoll	On	No	nl-ping	nl-ping	off
5	IS_IcmpPoll	On	No	nl-ping	nl-ping	off
11	SnmpFastPoll	On	No	system	system	off
10	SnmpPoll	On	No	system	system	off

The Poll List window contains summary information about all poll definitions in the NerveCenter database for the active server. From this window you can open existing poll definitions or create new ones.

3. Select **New**.

   NerveCenter displays the Poll Definition window.



NERVECENTER:Poll Definition

Poll ID: ____

Name:

Property: NO_PROP

Port:

Poll Rate: 10 ( ) Days ( ) Hours [✓ Minutes] ( ) Seconds

Overrun Policy: Skip

☑ Suppressible

☑ Execute Perl in Global Space

Enabled: ( ) On [✓ Off]

Handlers:
☐ On Enabled / Disabled
☐ On Scheduled / Unscheduled
☐ On Start / End Execution
☑ On Response

4. In the **Name** field, type the name for your new poll definition, **1CheckTraffic**.
5. From the **Property** list select **myNodes**.

   In the last chapter you assigned a particular set of nodes to the CriticalDevices property group. Since **myNodes** is unique to the CriticalDevices property group, when you select it you are telling NerveCenter to monitor only that particular set of nodes.

6. Leave **Port** blank.

   By leaving the **Port** text field blank you are telling this poll to communicate with each node on the port specified in the node's definition. (This would have been done by your network administrator.)

7. In the **Poll Rate** area, type **30** in the field and select the **Seconds** button.

   This sets the poll rate at 30 seconds.

You are now ready to define the core of your poll, the poll condition. The next activity will step you through that process.

Writing a Poll Condition

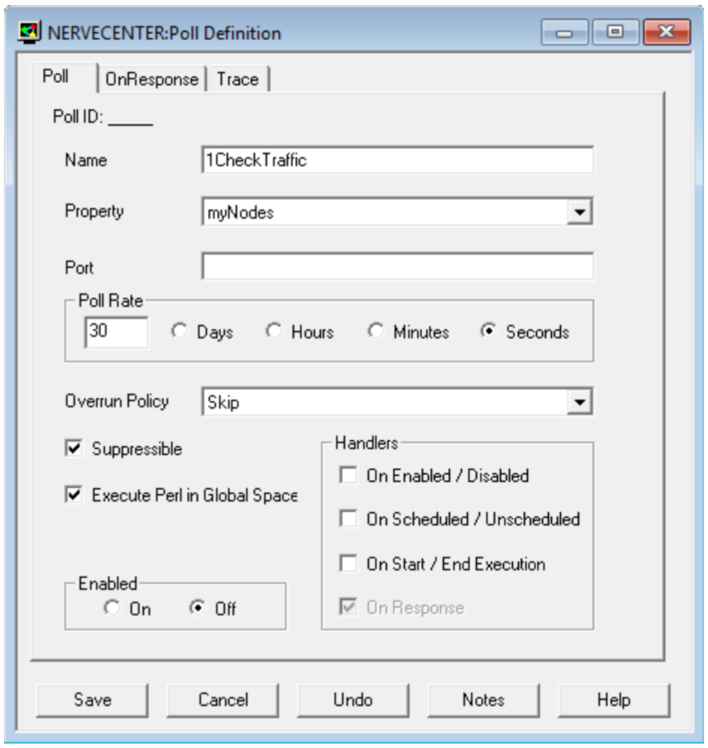
The core of each poll is its poll condition. A poll condition is a Perl script describing the conditions the poll should monitor. The poll condition also defines what trigger to fire when the poll detects those conditions.

This next activity will step you through the process of writing a poll condition for the 1CheckTraffic poll.

To write a poll condition

1. In the Poll Definition window, select the **Poll Condition** tab.

   NerveCenter displays the Poll Condition page.



NERVECENTER:Poll Definition

Poll | OnResponse | Trace

Poll ID: ____

Name: 1CheckTraffic

Property: myNodes

Port:

Poll Rate: 30 ( ) Days ( ) Hours ( ) Minutes (●) Seconds

Overrun Policy: Skip

[✓] Suppressible

[✓] Execute Perl in Global Space

Enabled: ( ) On (●) Off

Handlers:
[ ] On Enabled / Disabled
[ ] On Scheduled / Unscheduled
[ ] On Start / End Execution
[✓] On Response

2. From the **Base Object** drop-down list, select **ifEntry**.

   Selecting this Management Information Base (MIB) base object tells the poll condition to reference only the ifEntry object.

3. In the **Attribute** list, double-click the attribute **ifInOctets**.

   ifEntry.ifInOctets appears in the **Poll Condition** text field.

4. In the **Poll Condition** text field, position the cursor before **ifEntry.ifInOctets**. Then type if(so that the expression states this:

   ```
   if( ifEntry.ifInOctets
   ```

5. With the cursor still at the position between **if(** and **ifEntry.ifInOctets**, right-click. Select **Other functions**, then **delta**.

   The word delta and a left parenthesis appears in the expression.

   The delta in this expression will return the difference between the values of ifEntry.ifInOctets taken over two consecutive polling instances.

6. Type in the necessary characters to complete the expression, so that it looks like the following:

   ```
   if( delta( ifEntry.ifInOctets) >= 5)
   {

   }
   ```

   To ensure that this poll will find a high-traffic condition, a low number for ifInOctets (five) is being used. In a real poll, you will want to define poll conditions based on the performance statistics for your network.

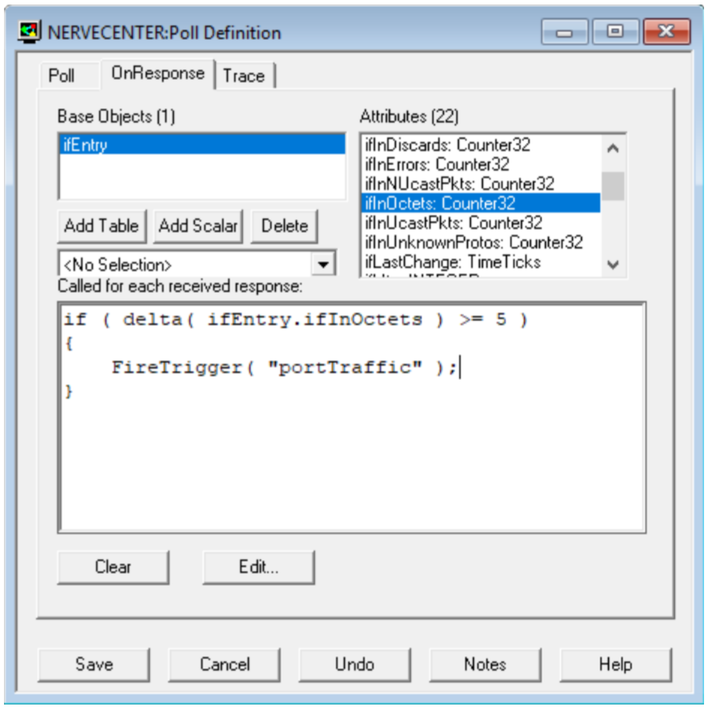
7. Position the cursor on the line between the two brackets and right-click. Select **Other functions**, then **FireTrigger**.

   The word **FireTrigger** and a left parenthesis appears in the expression.

8. Type **"portTraffic");**

   This tells the poll to fire the trigger portTraffic if it detects a difference between the values of ifEntry.ifInOctets of two successive polls to be at least five.

   The entire expression should looks like this:



```
if ( delta( ifEntry.ifInOctets ) >= 5 )
{
    FireTrigger( "portTraffic" );
}
```

Note: You can always type the entire expression manually. Right-click selections allow you to see what elements are available to build a poll condition. Just be sure to use the correct syntax.

9. Select **Save**.

You have now completed writing your poll condition. Before 1CheckTraffic can begin polling, you must complete a few more steps as explained in "Enabling a Poll" on the next page.

What is a trigger?

In the last activity you had the poll fire a trigger if it detects a certain condition. While writing the poll condition, you created the trigger portTraffic.

A *trigger* is a flag that can be generated by one of the following:

- A poll
- A trap mask
- An alarm
- A Perl subroutine

Triggers cause alarms to transition from one state to another. Such a transition may cause NerveCenter to take certain actions.

Enabling a Poll

You have just completed creating the 1CheckTraffic poll, which will poll all the nodes in the CriticalDevices property group for the number of ifEntry.ifInOctets. When the poll detects high-traffic conditions (what we've defined as a difference of five), it will fire the trigger portTraffic.

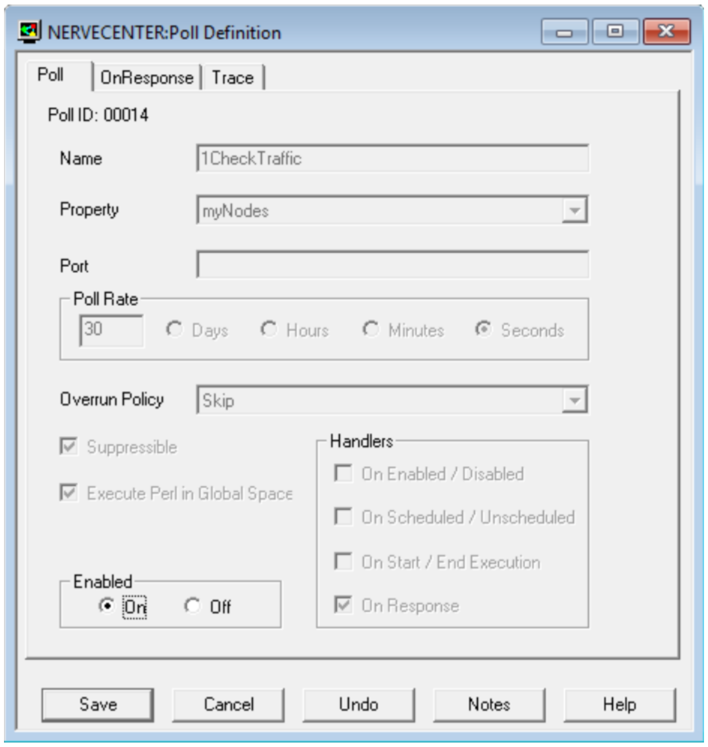
For this poll to begin actively polling, it must be enabled.

TO ENABLE A POLL

1. In the Poll Definition window, select the **Poll** tab.

   NerveCenter displays the Poll page.

   The poll condition now appears in the poll condition window on the Poll page.



Poll ID: 00014

Name: 1CheckTraffic

Property: myNodes

Port:

Poll Rate: 30 ( ) Days ( ) Hours ( ) Minutes (●) Seconds

Overrun Policy: Skip

[✓] Suppressible

[✓] Execute Perl in Global Space

Handlers: [ ] On Enabled / Disabled [ ] On Scheduled / Unscheduled [ ] On Start / End Execution [✓] On Response

Enabled: (●) On ( ) Off

2. In the **Enabled** frame, select **On**.

   Notice that once the poll is enabled, all active fields and buttons are grayed out. A poll must be disabled to be modified.

3. Select **Save**.

You have just enabled the poll 1CheckTraffic. However, because of a NerveCenter feature called *smart polling*, this poll will not work until you create an alarm that is instantiated by its trigger. You will learn how to create this alarm in the next chapter, How to Use Alarms on page 37.

What is smart polling?

In the previous activity, you enabled the 1CheckTraffic poll. However, because of smart polling, NerveCenter will not poll the CriticalDevices nodes until an alarm has been created that includes the portTraffic trigger.

Smart polling is a feature of NerveCenter designed to minimize the amount of traffic polls generate. NerveCenter sends a poll to a node only if the poll:

- Is part of a behavior model designed to manage that node
- Can cause an immediate state transition in an alarm
- The poll's base object exists as a property in the node's property group

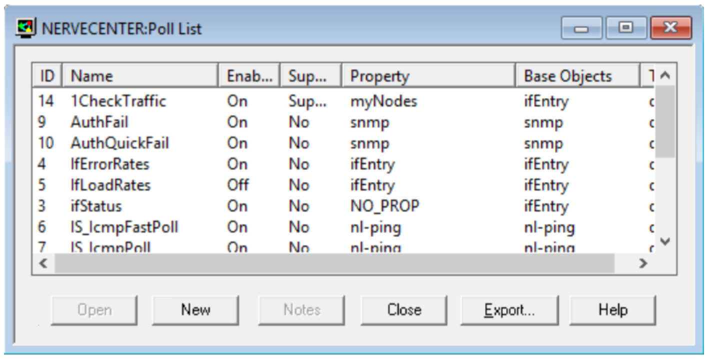
Modifying a Poll Condition

Before creating an alarm associated with the 1CheckTraffic poll, you should modify the poll condition to improve the poll, which gives you a chance to see an additional feature of the poll list.

TO MODIFY A POLL CONDITION

1. From the **Admin** menu, choose **Poll List**.

   NerveCenter displays the Poll List window.



ID	Name	Enab...	Sup...	Property	Base Objects	T
14	1CheckTraffic	On	Sup...	myNodes	ifEntry	c
9	AuthFail	On	No	snmp	snmp	c
10	AuthQuickFail	On	No	snmp	snmp	c
4	IfErrorRates	On	No	ifEntry	ifEntry	c
5	IfLoadRates	Off	No	ifEntry	ifEntry	c
3	ifStatus	On	No	NO_PROP	ifEntry	c
6	IS_IcmpFastPoll	On	No	nl-ping	nl-ping	c
7	IS_IcmpPoll	On	No	nl-ping	nl-ping	c

Notice that the 1CheckTraffic poll is listed as being enabled. You will need to turn it off before we can modify it.

2. Right-click on the **1CheckTraffic** poll.

   Notice that the pop-up menu gives you several ways to alter the poll without entering the Poll Definition window.

3. Select **Off**.

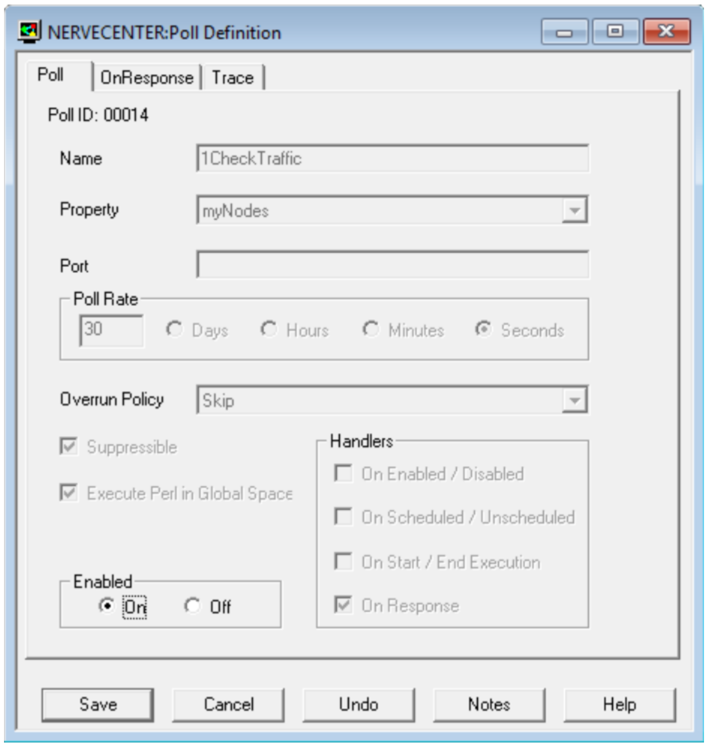
   You could also have turned off the poll from within the Poll Definition window.

4. Select **Open**.
5. In the Poll Definition window, select the **Poll Condition** tab.
6. Modify the poll expression in the **Poll Condition** field so that it looks like this:

   ```
   if((delta( ifEntry.ifInOctets )>=5) and
   (delta( ifEntry.ifOutOctets )>=5))
   {
   FireTrigger( “portTraffic” );
   }
   ```

   The modified poll will now check for an increase of incoming traffic, as well as outgoing.

7. In the Poll Definition window, select the **Poll** tab.
8. In the **Enabled** frame, select **On**.



NERVECENTER:Poll Definition

Poll | OnResponse | Trace

Poll ID: 00014

Name: 1CheckTraffic

Property: myNodes

Port:

Poll Rate: 30 ( ) Days ( ) Hours ( ) Minutes [✓ Seconds]

Overrun Policy: Skip

☑ Suppressible

☑ Execute Perl in Global Space

Enabled: [✓ On] ( ) Off

Handlers:
☐ On Enabled / Disabled
☐ On Scheduled / Unscheduled
☐ On Start / End Execution
☑ On Response

9. Select **Save**.

Now you have created, modified, and enabled the 1CheckTraffic poll. In "How to Use Alarms" on page 37 you will create an alarm that will transition states when your new poll generates the portTraffic trigger.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter.

1. Create a new poll to check if a host on your network can function as a gateway. Some of the actions you will take include:
   a. Naming the poll **1FindGateways**
   b. Selecting the MIB base object ip
   c. Using the attribute **ipForwarding**
   d. Generating a trigger called **gotGateway**
2. Modify the 1CheckTraffic poll to generate the trigger **notBusy** if the true conditions are not met.

Note: Include an else statement in the poll condition.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 4 Review Questions on page 122.

1. How can you configure a poll to monitor only the file servers on your network?

2. What is a trigger?

3. What conditions must be met for a poll to be active?

Summary of What You Learned

In this chapter you learned how to use a poll in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to create a new poll
- How to write a poll condition
- How to enable a poll
- How to modify poll
- You also were introduced to the following concepts:
- Polls
- Triggers
- Smart polling

How to Use Alarms

5

In the last chapter, you learned how to create and enable polls to monitor certain conditions on your network. However, a poll will only work if it can cause an alarm to transition. Alarms are at the heart of NerveCenter's behavior models.

This chapter explains:

- What an alarm is and how it fits in a behavior model
- How to create a new alarm
- How to design a state diagram
- How to enable an alarm
- How to modify an alarm

What is an Alarm?

At the heart of a NerveCenter behavior model is one or more *alarms*. An alarm is an object that correlates one or more detected network events and performs certain actions in response to those events.

Alarms stay in a normal state, usually called Ground. When the alarm manager sees a trigger whose key attributes match those of a pending alarm transition, the manager causes the alarm to move from one state to another. NerveCenter then performs any of the actions associated with that transition.

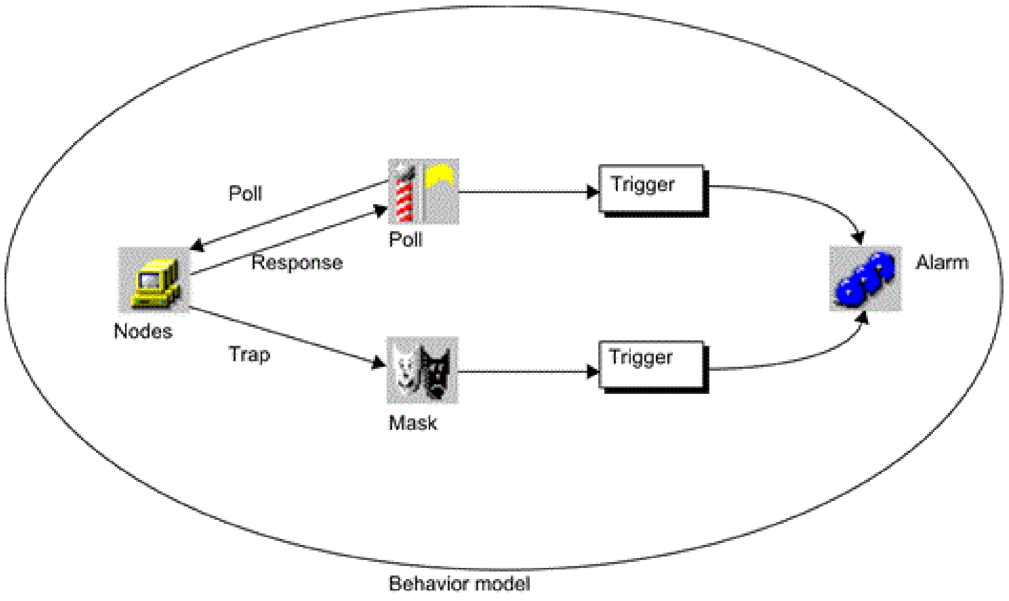


Figure 5: The Role of an Alarm within a Behavior Model

NerveCenter ignores alarms until they are instantiated. Instantiated alarms have transitioned from Ground into another state.

How to Create an Alarm

As in the last scenario, you want to know which of the CriticalDevices nodes are experiencing high traffic. Now that you have created the poll 1CheckTraffic, you must create an alarm that will notify you when NerveCenter has detected high traffic.

This scenario includes five activities:

1. Creating a New Alarm on the facing page
2. Designing a State Diagram on page 42
3. Enabling an Alarm on page 45
4. Modifying Alarm State Diagrams on page 48
5. Adding Another State to an Alarm on page 50

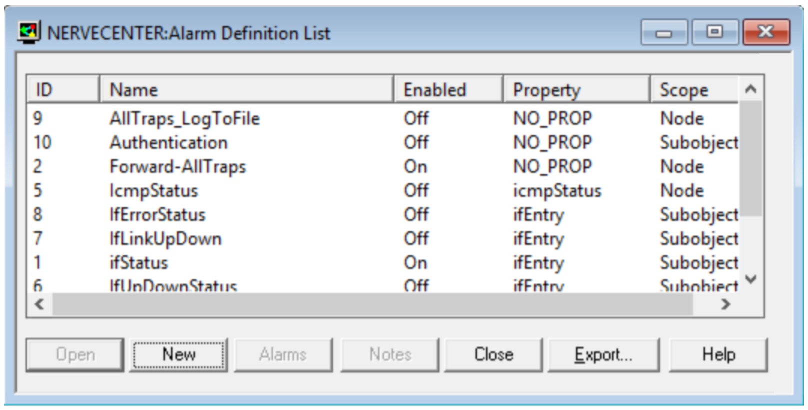
Creating a New Alarm

This first activity steps you through the first stage of creating a new alarm to monitor high traffic.

To Create a New Alarm

1. Open the NerveCenter Client and connect to the appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Alarm Definition List**.

   NerveCenter displays the Alarm Definition List window.



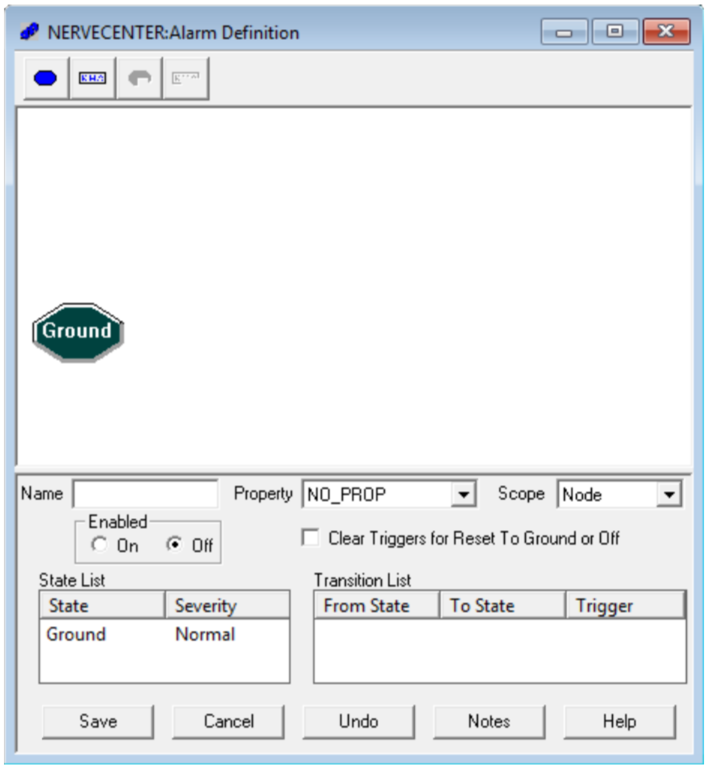
ID	Name	Enabled	Property	Scope
9	AllTraps_LogToFile	Off	NO_PROP	Node
10	Authentication	Off	NO_PROP	Subobject
2	Forward-AllTraps	On	NO_PROP	Node
5	IcmpStatus	Off	icmpStatus	Node
8	IfErrorStatus	Off	ifEntry	Subobject
7	IfLinkUpDown	Off	ifEntry	Subobject
1	ifStatus	On	ifEntry	Subobject
6	IfUpDownStatus	Off	ifEntry	Subobject

This window lists all the alarms in the NerveCenter database.

3. In the Alarm Definition List window, select **New**.

   NerveCenter displays the Alarm Definition window.

   The Alarm Definition window allows you to examine, create, or change an alarm definition that is implemented through the state diagram. The next activity steps you through the process of designing the state diagram. You must first name the alarm and limit the nodes it will monitor.



Name:
Property: NO_PROP
Scope: Node
Enabled: On [✓ Off]
Clear Triggers for Reset To Ground or Off: ☐

State List

State	Severity
Ground	Normal

Transition List

From State	To State	Trigger

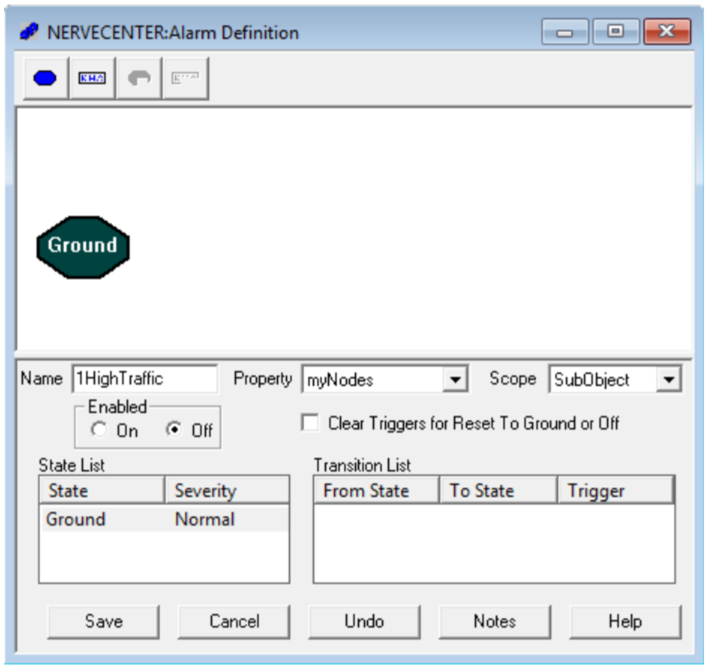
4. In the **Name** field, type **1HighTraffic**.
5. From the **Property** list, select the **myNodes** property.

   By selecting the **myNodes** property, you are limiting the alarm to monitor only nodes in the CriticalDevices property group.

   In this case, limiting the nodes is redundant, because you have already limited the related 1CheckTraffic poll to this property group. If you had wanted the alarm to monitor all the nodes in your network, you would have selected the property **NO_PROP**.

6. From the **Scope** list, select **SubObject**.

   By selecting the SubObject scope, you are telling the alarm to monitor each port on each CriticalDevices node separately. Using scope in alarms is an advanced topic that is discussed in How to Use Alarm Scope in Behavior Models on page 89.



Name: 1HighTraffic
Property: myNodes
Scope: SubObject
Enabled: On [✓ Off]
☐ Clear Triggers for Reset To Ground or Off

State List

State	Severity
Ground	Normal

Transition List

From State	To State	Trigger

7. Select **Save**.

You have just set the parameters of your new alarm by naming it, limiting it to the CriticalDevices property group, and setting the scope to the SubObject level. In the next activity you will design the core of the alarm, the State Diagram.

What is a state diagram?

In the next activity you will be designing the state diagram for the 1HighTraffic alarm.

An alarm's *state diagram* is the area where you define the potential states of the alarm and what would trigger an instance of that alarm.

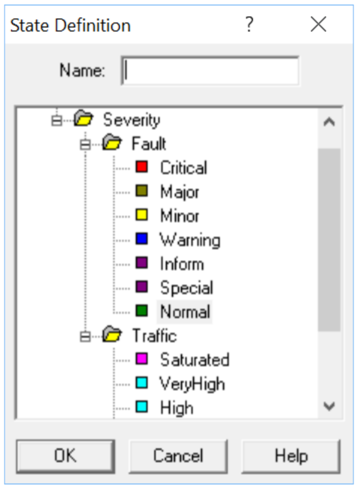
Every alarm must have at least a normal state. This is usually called Ground and is designated by a dark green hexagon in the state diagram.

Designing a State Diagram

This next activity steps you through the process of designing a state diagram for the 1HighTraffic alarm.

TO DESIGN A STATE DIAGRAM

1. Right-click on the background of the alarm state diagram and select **Add State**. NerveCenter displays the State Definition window.

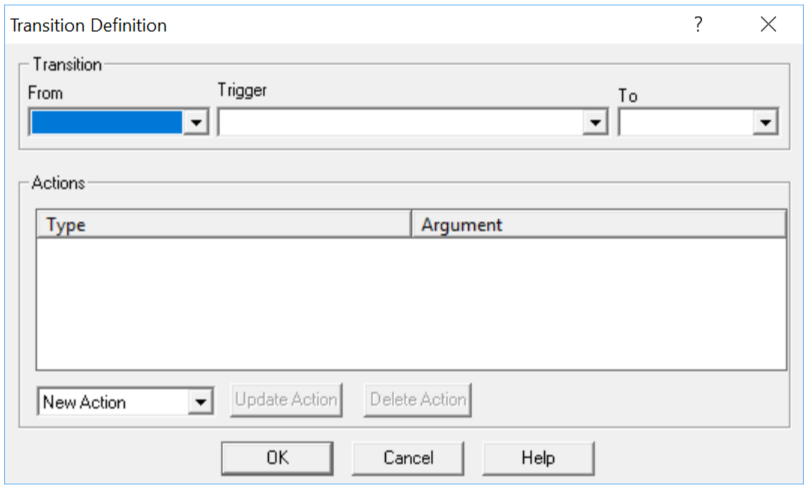


2. In the **Name** field, type Busy.
3. Under **Severity > Traffic**, select **Low**.
4. Select **OK**.

   The State Definition window closes. An icon appears in the top left corner with the name of your new state, Busy, and a bright green color, indicating a low severity.

5. Position the Busy state icon in the middle of the state diagram.
6. Right-click on the background of the alarm state diagram and select **Add Transition**.

   The Transition Definition window appears.



7. In the **From** list, select **Ground**. In the **To** list, select **Busy**.

   You are creating an instance when the alarm will transition from the Ground state to the Busy state.

8. In the **Trigger** list, select **portTraffic**.

   You are telling the alarm to transition to the Busy state whenever NerveCenter fires the portTraffic trigger. In "How to Use Polls" on page 25 you created a poll that would fire portTraffic under certain high-traffic conditions.

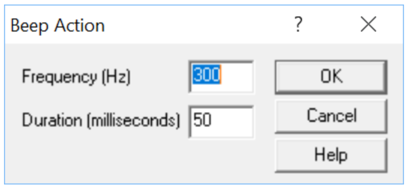
   The **Trigger** list in the Transition Definition window contains all the predefined triggers and user-defined triggers in NerveCenter's database.

9. In the **Actions** area, select **New Action**.

   A list of alarm actions appears. You can associate one or more of these actions with each transition in a state diagram. A transition does not need an action.

10. From the alarm action list, select **Beep**.

    The Beep Action window appears.



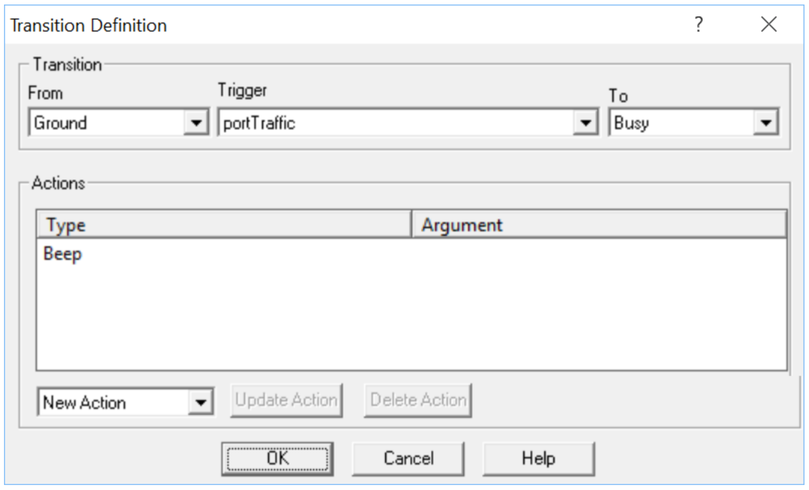
Frequency (Hz) 300

Duration (milliseconds) 50

Some actions require additional information. For the Beep action, the beep's frequency and duration can be changed. Notice the fields include default values.

11. In the Beep Action window, select **OK**.

    Beep now appears under the list of actions for this transition.



From	Trigger	To
Ground	portTraffic	Busy

Type	Argument
Beep	

12. In the Transition Definition window, select **OK**.

    In the state diagram, a transition named portTraffic connects the states Ground and Busy.

    Notice the transition name, portTraffic, is the same as the trigger that causes this transition.

13. In the Alarm Definition window, select **Save**.
14. Select **Cancel** to close the window.

You have just completed creating your first alarm. In the next activity you will enable the alarm and the poll to test it out.

What is a transition?

In this last activity you added a transition to the 1HighTraffic alarm called portTraffic.

A *transition* tells the alarm to move from one state to another whenever a particular trigger is fired. Each transition can have an *action* or *group of actions* associated with it. The transition is always named the same as the associated trigger.

Enabling an Alarm

You have just completed creating the 1HighTraffic alarm. You designed the state diagram to transition the alarm from the Ground state to the Busy state when NerveCenter fires the trigger portTraffic.

This next activity will step you through the process of enabling your new 1HighTraffic alarm.

TO ENABLE AN ALARM

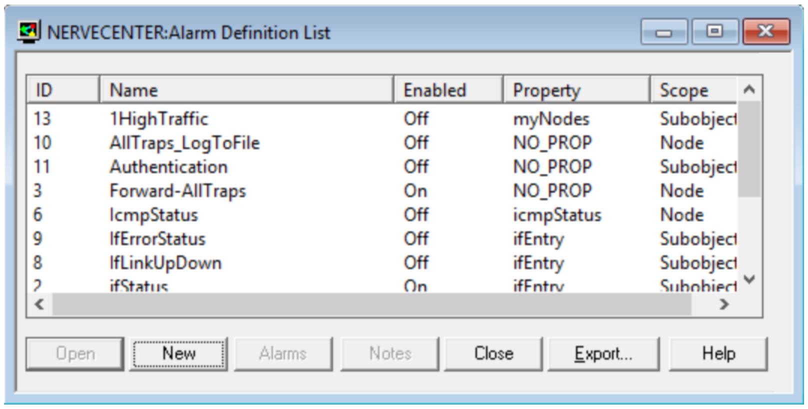
1. From the **Admin** menu, choose **Poll List**.

   NerveCenter displays the Poll List window.

   The 1CheckTraffic poll should have Enabled listed as **On**.

2. If the 1CheckTraffic poll is not enabled, turn it on.
3. From the **Admin** menu, choose **Alarm Definition List**.

   NerveCenter displays the Alarm Definition List window.



ID	Name	Enabled	Property	Scope
13	1HighTraffic	Off	myNodes	Subobject
10	AllTraps_LogToFile	Off	NO_PROP	Node
11	Authentication	Off	NO_PROP	Subobject
3	Forward-AllTraps	On	NO_PROP	Node
6	IcmpStatus	Off	icmpStatus	Node
9	IfErrorStatus	Off	ifEntry	Subobject
8	IfLinkUpDown	Off	ifEntry	Subobject
2	ifStatus	On	ifEntry	Subobject

The Alarm Definition list should include the new 1HighTraffic alarm.

4. Right-click on the alarm **1HighTraffic**.

   Notice the pop-up menu gives you several ways to alter the alarm without entering the Alarm Definition window.

5. Select **On**.

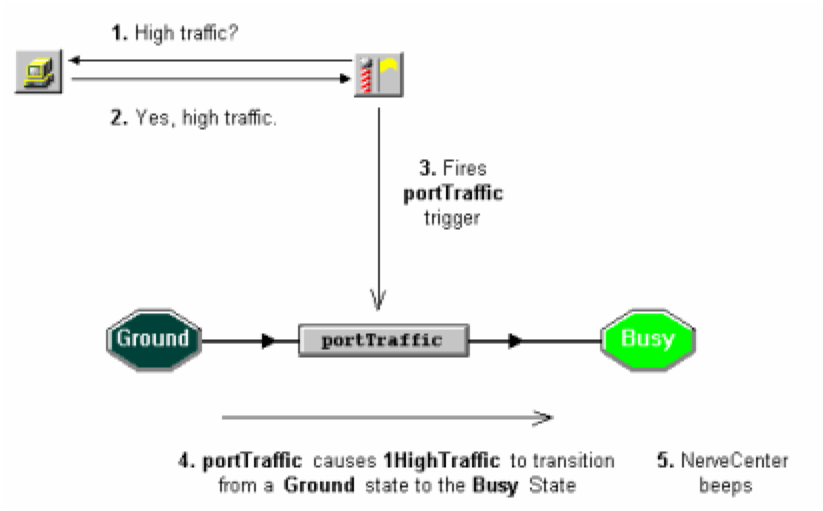
   This enables the 1HighTraffic alarm.

   You soon should hear beeps.

What is NerveCenter doing?

The poll 1CheckTraffic polls the nodes in CriticalDevices for high traffic conditions.

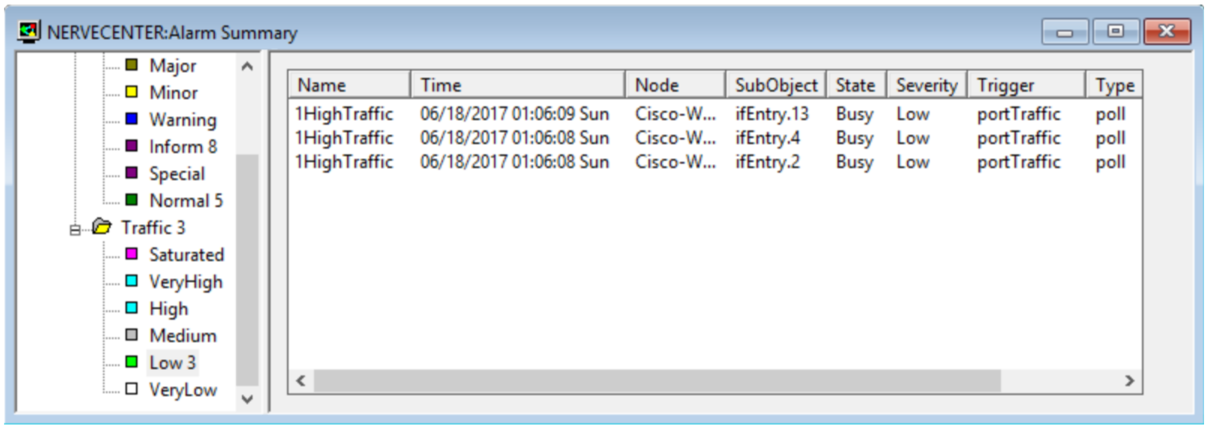
1. The agents respond that the nodes are experiencing high traffic conditions.
2. The poll 1CheckTraffic fires the trigger portTraffic.
3. The trigger portTraffic causes the alarm 1HighTraffic to transition from the Ground state to the Busy state.
4. NerveCenter performs the action associated with the transition. In this case, it beeps.


1. High traffic?
2. Yes, high traffic.
3. Fires portTraffic trigger
Ground
portTraffic
Busy
4. portTraffic causes 1HighTraffic to transition from a Ground state to the Busy State
5. NerveCenter beeps

Note: When the alarm is in the Busy state, NerveCenter will not send the poll out to the node again because of smart polling.

5. From the **Admin** menu, choose **Alarm Summary**.

   NerveCenter displays the Alarm Summary window.



Name	Time	Node	SubObject	State	Severity	Trigger	Type
1HighTraffic	06/18/2017 01:06:09 Sun	Cisco-W...	ifEntry.13	Busy	Low	portTraffic	poll
1HighTraffic	06/18/2017 01:06:08 Sun	Cisco-W...	ifEntry.4	Busy	Low	portTraffic	poll
1HighTraffic	06/18/2017 01:06:08 Sun	Cisco-W...	ifEntry.2	Busy	Low	portTraffic	poll

The Alarm Summary window presents information about the active alarms for the active server. The alarm summary tree on the left displays folders for the different levels of alarm severity.

On the right is a list of all the current alarm *instances*. It includes the following information:

- The name of the alarm
- The time and date the alarm was instantiated
- The node and subobject which had the conditions of interest
- The name and the severity of the alarm's current state
- The name of the trigger that caused the transition
- The type and name of trigger generator

In the alarm summary tree, there should be numbers beside Low, Traffic, and Severity. In the alarm instances list, 1HighTraffic probably has several alarm instances.

You originally set the alarm's Scope to **SubObject**. The alarm 1HighTraffic, therefore, monitors each of your node's ports *separately*. Because of this there may be more than one alarm instance per node in your CriticalDevices group. If you had set Scope to **Node**, it would have shown no more than one instance per node.

6. Turn the **1CheckTraffic** poll and the **1HighTraffic** alarm to **Off**.

You just created, enabled, and viewed instances of your first poll. Now it is time to modify the 1HighTraffic alarm to make it easier to read and more useful. The next two activities will show you how to do that.

Modifying Alarm State Diagrams

There may be times you will want to adjust the icons of the state diagram to make them easier to read. This next activity will step you through the process of resizing the icons in the state diagram of the 1HighTraffic alarm.

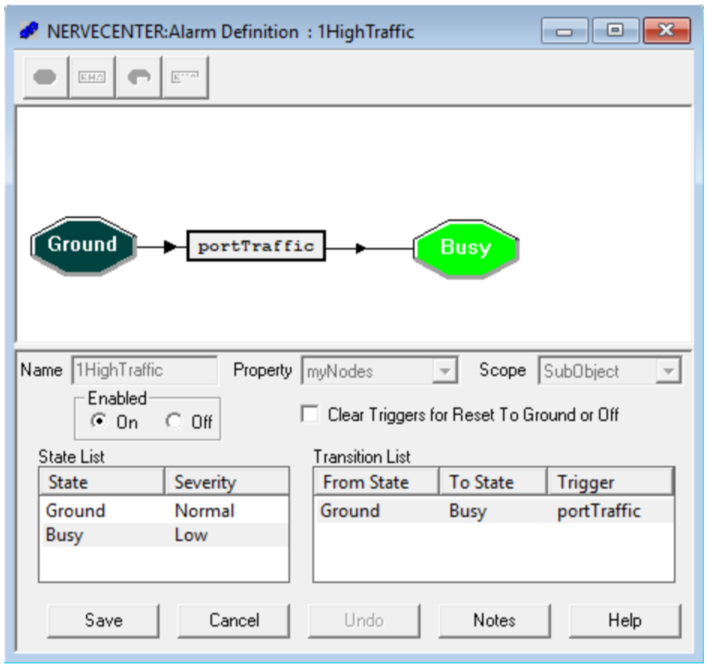
TO MODIFY AN ALARM'S STATE DIAGRAM

1. From the **Admin** menu, select **Alarm Definition List**.

   NerveCenter displays the Alarm Definition List window.

2. From the Alarm Definition list, select **1HighTraffic**. Then select **Open**.

   The Alarm Definition window for 1HighTraffic appears.



Name: 1HighTraffic
Property: myNodes
Scope: SubObject
Enabled: [✓ On] Off
☐ Clear Triggers for Reset To Ground or Off

State List

State	Severity
Ground	Normal
Busy	Low

Transition List

From State	To State	Trigger
Ground	Busy	portTraffic

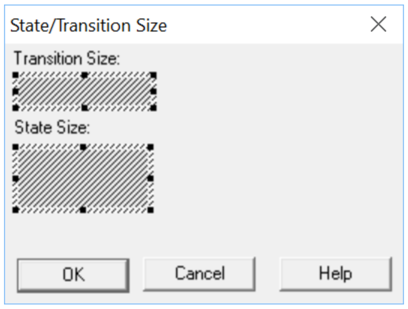
3. If the alarm is on, select **Off**, then select **Save**.

   Before you can modify an alarm, you must turn it off.

   More than likely, the icons for the states and the transitions will be too small to display the entire names. You will need to resize them.

4. Right-click anywhere within the state diagram. Select **Size**.

   The State/Transition Size window appears.

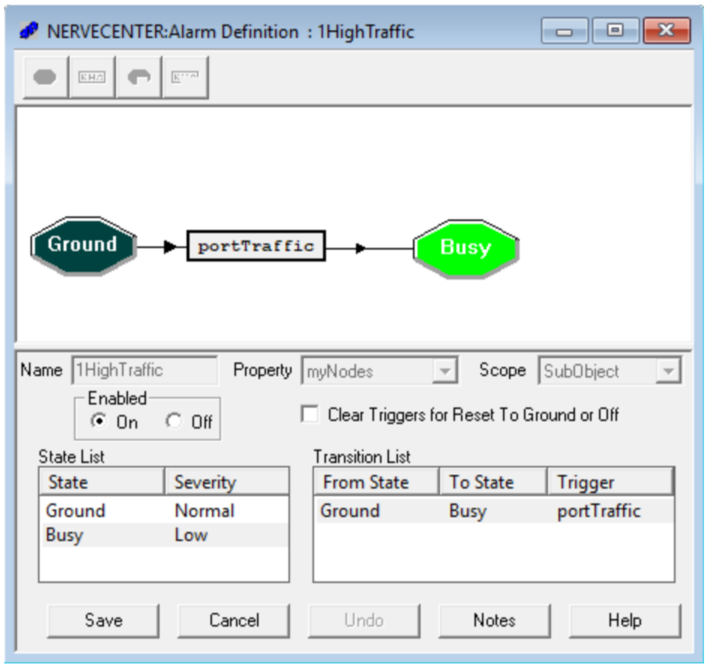


   The State Size and Transition Size rectangles indicate the current icon size.

5. Drag the handles of the State Size rectangle to change the height and width of the rectangle. Repeat for the Transition Size rectangle.

6. Select **OK**.

   The State/Transition Size window closes and the icons in the state diagram are now the size you specified.



Name 1HighTraffic Property myNodes Scope SubObject

Enabled: [✓ On] Off

Clear Triggers for Reset To Ground or Off

State List

State	Severity
Ground	Normal
Busy	Low

Transition List

From State	To State	Trigger
Ground	Busy	portTraffic

7. Select **Save**.

Now that you have made the state diagram for 1HighTraffic easier to read, it is time to add another state to it. The next activity will step you through this process.

Adding Another State to an Alarm

Now that you have resized the icons in your alarm's state diagram, it is easier to read. There is, however, an even bigger problem. Currently, 1HighTraffic notifies you whenever high traffic occurs. But you expect occasional high traffic and don't want to be bothered every time there is an occasional traffic spike.

This next activity steps you through the process of adding another state to the 1HighTraffic alarm so that it will only notify you after a second occurrence of high-traffic conditions.

TO ADD ANOTHER STATE TO AN ALARM

1. In the Alarm Definition window, make sure **1HighTraffic** is turned off.
2. Right-click on the background of the alarm state diagram and select **Add State**.

   NerveCenter displays the State Definition window.

3. In the **Name** field of the State Definition window, type **TooBusy**. From the Traffic severity list, select **Very High**. Then select **OK**.

   The State Definition window closes. An icon appears in the top left corner with the name of your new state, TooBusy, and a light blue color, indicating a very high severity.

4. In the state diagram, position the icon for the TooBusy state to the right of the icon for the Busy state.
5. Right-click on the background of the alarm state diagram and select **Add Transition**.

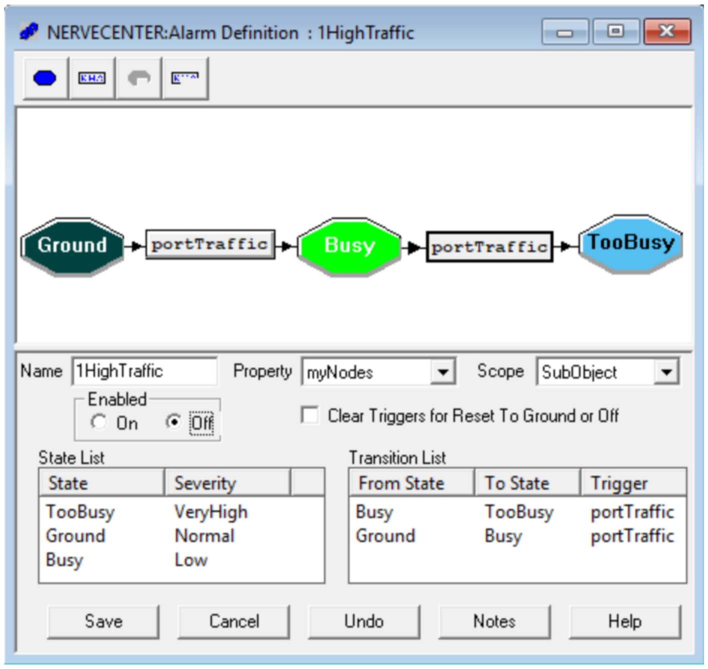
   The Transition Definition window appears.

6. In the **From** list, select **Busy**. In the **To** list, select **TooBusy**. In the **Trigger** list, select **portTraffic**.
7. Select **New Action**. From the **Action Alarm** list, select **Beep**. In the Beep Action window, select **OK** to keep the default values and close the window.

   Beep is added to the actions list.

8. In the Transition Definition window, select **OK**.

   The Transition Definition window closes, and an icon for the second portTraffic transition appears between the icons for the Busy state and the TooBusy state.



Name: 1HighTraffic
Property: myNodes
Scope: SubObject
Enabled: On [✓ Off]
Clear Triggers for Reset To Ground or Off

State List

State	Severity
TooBusy	VeryHigh
Ground	Normal
Busy	Low

Transition List

From State	To State	Trigger
Busy	TooBusy	portTraffic
Ground	Busy	portTraffic

You may need to adjust the state diagram to make it easier to read.

9. In the state diagram, double-click the **portTraffic** transition between the Ground state and the Busy state.

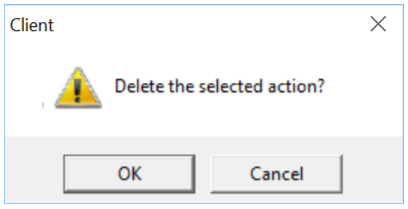
   The Transition Definition window appears. Notice that Beep is still listed in the Actions list. Since you now want 1HighTraffic to notify you only when there have been two occurrences of high traffic, you must delete this action.

10. In the **Actions** list, select **Beep**.

    The **Delete Action** button is enabled.

11. Select **Delete Action**.

    A warning box appears asking if you want to delete the selected action.



Delete the selected action?

12. Select **OK**.

    The action is deleted from the Actions list.

13. In the Transition Definition window, select **OK**.
14. In the Alarm Definition window, select **Save**.

Now, when you enable the 1CheckTraffic poll and the 1HighTraffic alarm, it will only beep after the second occurrence of high traffic.

In future chapters you will explore other ways to check the persistence of a condition on your network. But, first you will need to learn how to use NerveCenter to filter traps in "How to Use Trap Masks" on page 55.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter.

1. Create a new alarm that will always fire true triggers. Some of the actions you will take include:
   a. Naming the alarm **1Always**
   b. Including two states, **Ground** and **True**
   c. Including a transition between the two states prompted by the trigger **TrueTrigger**
   d. Leaving the actions empty and the alarm off
2. Modify the **1HighTraffic** alarm, by changing **Scope** to **Node**. Now turn on the **1HighTraffic** alarm and the **1CheckTraffic** poll.

   What is now different about what the Alarm Summary window displays?

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 5 Review Questions on page 122.

1. What causes an alarm to move from one state to another?

2. When is it unnecessary to specify a property for an alarm definition?

3. You need an alarm that first checks for high traffic on an interface and only then checks for a high error rate. Assume that you've already defined the necessary polls to collect the appropriate data. Draw an example of the state diagram below:

Summary of What You Learned

In this chapter you learned how to create, modify, and enable alarms in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to create an alarm
- How to design a state diagram
- How to enable an alarm
- How to modify an alarm's state diagram
- How to add an additional state to an alarm
- You also were introduced to the following concepts:
- Alarm
- State diagram
- Transition

How to Use Trap Masks

6

In the last chapter you learned how to create, modify, and enable alarms. An alarm will only transition from one state to another when a trigger generator fires a trigger. You have already created one trigger generator, a poll. In this chapter you will create another type of trigger generator, a trap mask.

This chapter explains:

- What a trap mask is and how it fits in a behavior model
- How to create a new mask
- How to create an alarm associated with a mask
- How to generate an artificial trap
- How to modify a mask's trigger function

What is a Trap Mask?

Alarms transition from one state to another when NerveCenter fires a trigger. There are five different types of trigger generators:

- A poll
- A trap mask
- An alarm instance
- A Perl subroutine action
- A fire Trigger action

Trap masks screen SNMP traps. These traps may be sent by managed nodes or by other sources, such as a NerveCenter server.

A trap mask is similar to a poll in its ability to trigger one or more alarms. Whereas a poll actively monitors conditions, the mask passively waits until it receives a relevant SNMP trap to act.

review - there was no image?

Figure 6: The Role of a Trap Mask within a Behavior Model

You, the user, define the complexity of the trigger each mask fires.

How to Create a Trap Mask

In this next scenario, you want to know if any of the nodes in the CriticalDevices group are communicating without proper authorization. You will create a trap mask to “mask out” or filter out any SNMP trap that warns of an authentication failure. Since the mask will be useless without an associated alarm, you will create an alarm as well.

The scenario includes four activities:

1. Creating a New Trap Mask below
2. Creating an Alarm to be Triggered by a Mask on page 59
3. Using Trapgen to Generate a Trap on page 62
4. Defining a Trigger Function on page 64

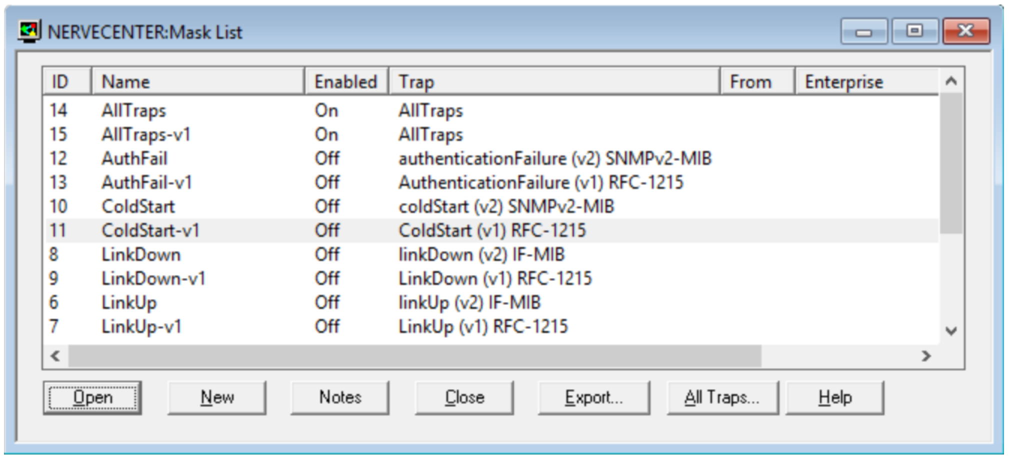
Creating a New Trap Mask

This first activity will step you through the process of creating a new trap mask that will listen for an SNMP trap signaling an authentication failure.

To Create a New Trap Mask

1. Open a NerveCenter Client and connect to an appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Mask List**.

   NerveCenter displays the Mask List window.

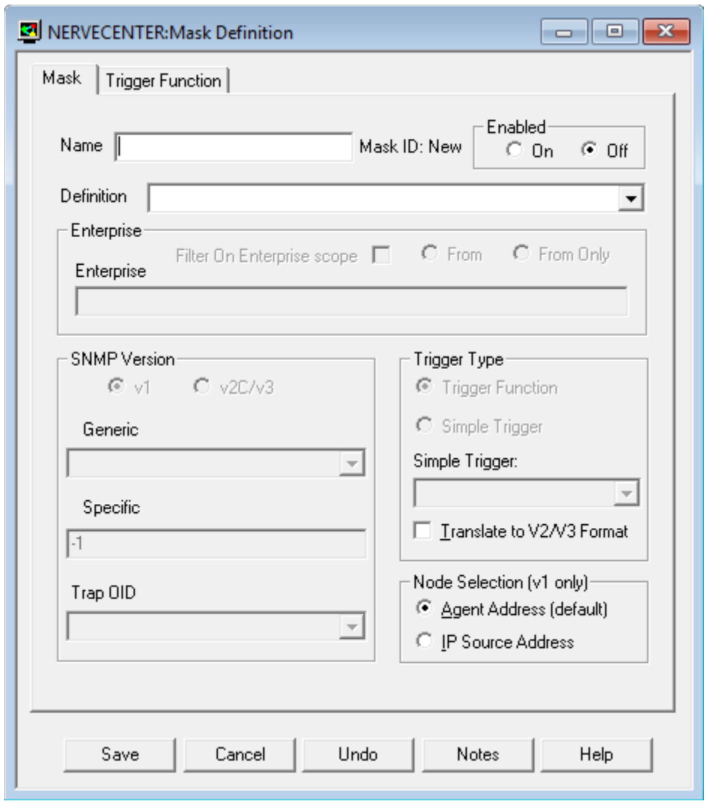


ID	Name	Enabled	Trap	From	Enterprise
14	AllTraps	On	AllTraps		
15	AllTraps-v1	On	AllTraps		
12	AuthFail	Off	authenticationFailure (v2) SNMPv2-MIB		
13	AuthFail-v1	Off	AuthenticationFailure (v1) RFC-1215		
10	ColdStart	Off	coldStart (v2) SNMPv2-MIB		
11	ColdStart-v1	Off	ColdStart (v1) RFC-1215		
8	LinkDown	Off	linkDown (v2) IF-MIB		
9	LinkDown-v1	Off	LinkDown (v1) RFC-1215		
6	LinkUp	Off	linkUp (v2) IF-MIB		
7	LinkUp-v1	Off	LinkUp (v1) RFC-1215		

The Mask List window contains a list of the trap masks in the NerveCenter database for the active server.

3. From the Mask List window, select **New**.

   The Mask Definition window appears.



Name:
Mask ID: New
Enabled: On [✓ Off]
Definition:
Enterprise
Filter On Enterprise scope ☐ From / From Only
Enterprise:
SNMP Version: [✓ v1] v2C/v3
Generic:
Specific: -1
Trap OID:
Trigger Type: [✓ Trigger Function] Simple Trigger
Simple Trigger:
☐ Translate to V2/V3 Format
Node Selection (v1 only): [✓ Agent Address (default)] IP Source Address

The Mask Definition window allows you to examine, create, or change a trap mask definition.

4. In the **Name** field, type **1CheckAuth**.
5. From the **Generic** list, select the generic trap number **AuthFail = 4**.

   You are telling the mask only to respond to an agent sending a generic SNMP trap 4. An agent sends a trap 4 when it receives an SNMP message with a bad community string.

6. Skip the **From** and **From Only** buttons. Also skip the **Enterprise** and **Specific** fields.

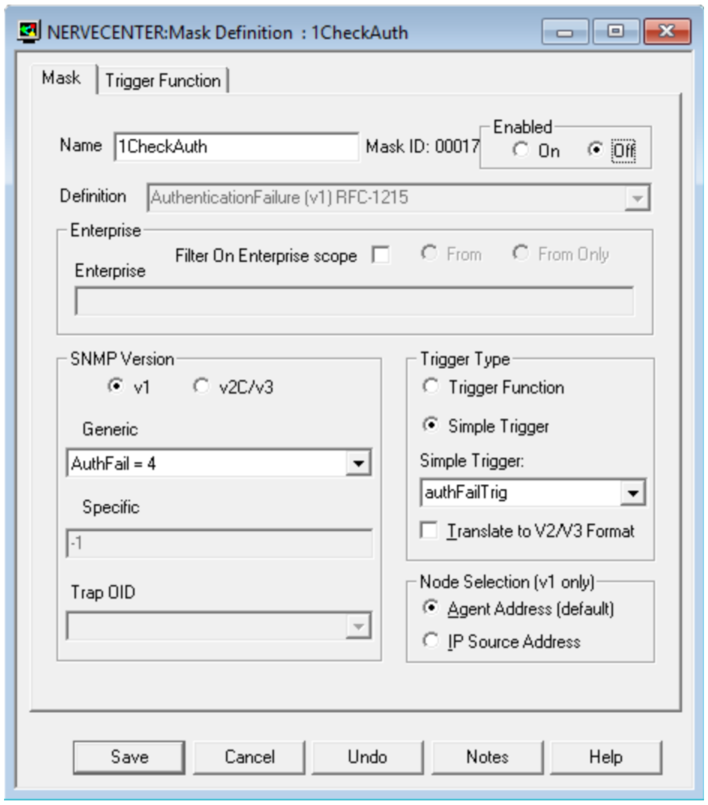
   These fields will be explained later.

7. In the **Trigger Type** field, select **Simple Trigger**.

   The **Simple Trigger** field is enabled.

   You would use the Trigger Function option only if you need to conditionally fire a trigger based on the contents of the trap's variable bindings. For this activity, a simple trigger will do.

8. In the **Simple Trigger** field, type **authFailTrig**.


NERVECENTER:Mask Definition : 1CheckAuth

Name 1CheckAuth
Mask ID: 00017
Enabled: ( ) On [✓ Off]
Definition AuthenticationFailure (v1) RFC-1215
Enterprise
Filter On Enterprise scope ☐ ( ) From ( ) From Only
Enterprise
SNMP Version [✓ v1] ( ) v2C/v3
Generic AuthFail = 4
Specific -1
Trap OID
Trigger Type ( ) Trigger Function [✓ Simple Trigger]
Simple Trigger: authFailTrig
☐ Translate to V2/V3 Format
Node Selection (v1 only) [✓ Agent Address (default)] ( ) IP Source Address

9. In the **Enabled** frame, select **On**.
10. Select **Save**, then **Cancel** to close.

The mask will now be listening for AuthFail traps. However, as we saw, with the 1CheckTraffic poll, the trap mask is useless until we have an alarm associated with it. In the next activity you will create an alarm for the mask.

What is a trap?

In the last activity you created a mask to detect an SNMP trap.

A *trap* is an unsolicited message sent by an SNMP agent on a managed node.

Traps generated according to the rules of SNMPv1 include the following information:

- A "generic" identification code, which is always a number from 0-6
- An "enterprise-specific" identification code, which is always a non-negative number
- An enterprise identifer
- Extra data in a trailing structure called "variable bindings"

Traps generated according to SNMPv3 and SNMPv3 include the following information:

- A "trap identifier"
- Extra data in a trailing structure called "variable bindings"

Creating an Alarm to be Triggered by a Mask

In the last activity, you created a mask called 1CheckAuth to detect for SNMP AuthFail traps. You now need an alarm that will be triggered when the trap is detected.

This next activity steps you through the process of creating an alarm that will alert you when an authentication failure is detected.

TO CREATE AN ALARM TO BE TRIGGERED BY A MASK

1. From the **Admin** menu, choose **Alarm Definition List**.

   NerveCenter displays the Alarm Definition window.

2. From the Alarm Definition List window, select **New**.

   The Alarm Definition window appears.

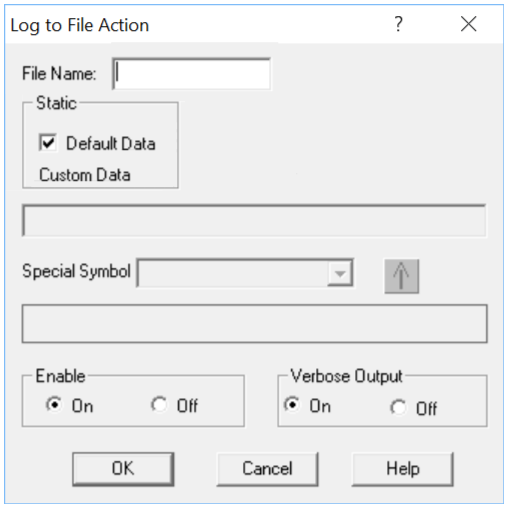
3. In the **Name** field, type **1FailedAuth**.
4. Set the **Property** field to **myNodes** and the **Scope** field to **Node**.
5. In the alarm's state diagram, add a state with a severity of Major and the name **Trap4Received**.
6. Size and position the state icon so that it is easy to read.
7. Create a transition from the Ground state to the Trap4Received state that is triggered by authFailTrig.

8. In the Transition Definition window, select **New Action**.

   A list of action alarms appears.

9. From the alarm action list, select **Log to File**.

   NerveCenter displays the Log to File Action window.



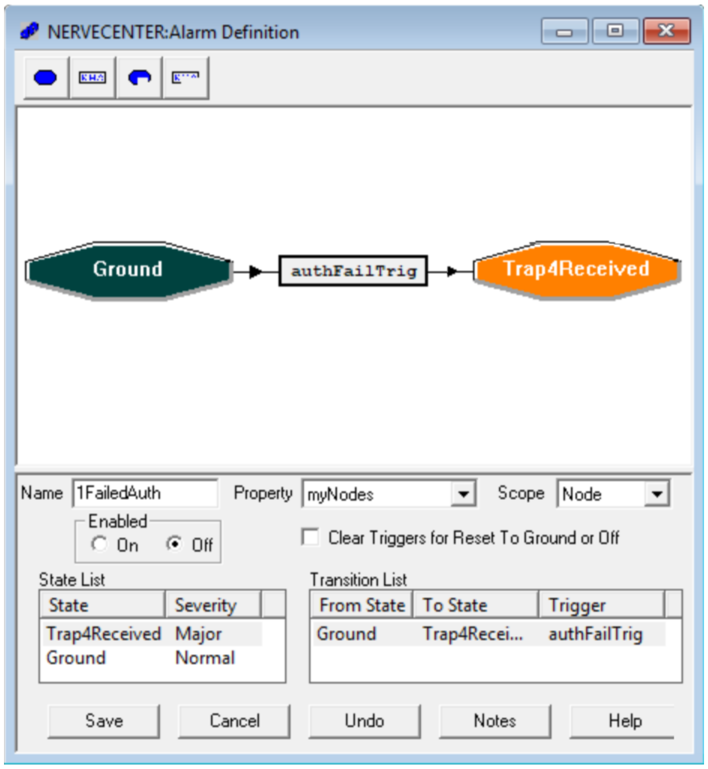
10. In the **File Name** field, type **myLog**.
11. Leave **Default Data** checked.
12. Select **On** in the **Enable** and **Verbose Output** fields.
13. Select **OK**.

    The Log to File Action window closes. The Log to File action is included in the **Actions** list.

    The Log to File alarm action writes information about an alarm transition to an ASCII text file. Since you entered a file name, the log file will be written to /opt/OSInc/userfiles/logs on the NerveCenter Server host.

14. In the Transition Definition window, select **OK**.

    The authFailTrig transition now appears in the state diagram. Size and position the icons as needed.



Name 1FailedAuth Property myNodes Scope Node

Enabled: On [✓ Off]

Clear Triggers for Reset To Ground or Off

State List

State	Severity
Trap4Received	Major
Ground	Normal

Transition List

From State	To State	Trigger
Ground	Trap4Recei...	authFailTrig

15. In the Enabled frame, select **On.**
16. Select **Save**.

You have just completed creating an alarm that will respond to the 1CheckAuth mask. The only step left is to have a trap for the mask to detect. The next activity will step you through one way you can use NerveCenter to generate a trap.

Using Trapgen to Generate a Trap

In the last activity, you created the 1FailedAuth alarm to respond to the 1CheckAuth mask. Previously, you created the 1CheckAuth mask to listen for a generic 4 trap. Since SNMP traps are unsolicited and sent in response to specific conditions on your network, it may be some time until one of your managed nodes sends this particular trap.

This next activity will step you through the process of artificially creating a generic 4 trap using the NerveCenter *trapgen* utility.

TO USE TRAPGEN TO GENERATE A TRAP

1. Check to make sure the 1CheckAuth mask and the 1FailedAuth alarm are On.
2. From the command line of the NerveCenter Server host, type:

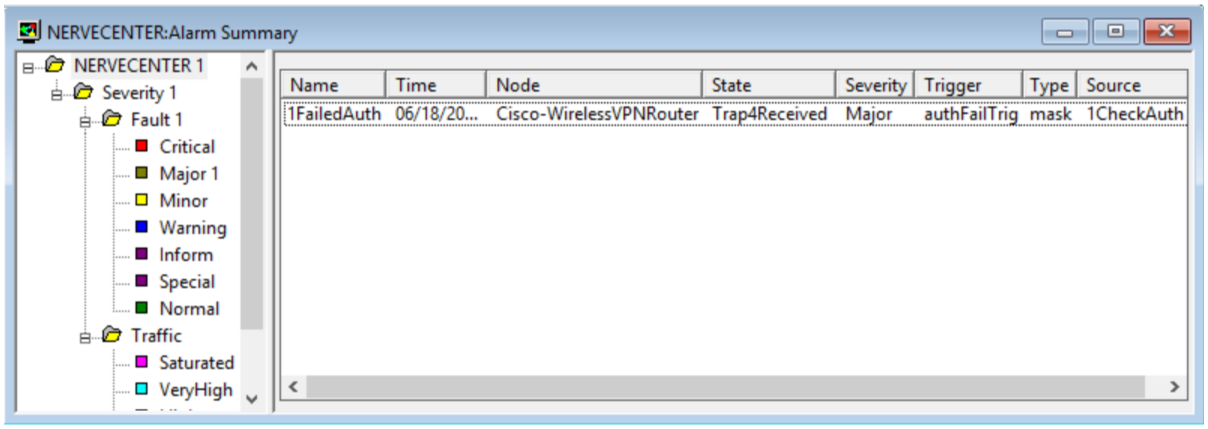
   ```
   /opt/OSInc/bin/trapgen server_name - node_name 4 0 ip_address -
   ```

   Note the trailing dash at the end of the command.

   You are commanding your platform to generate a trap with the following specifications:

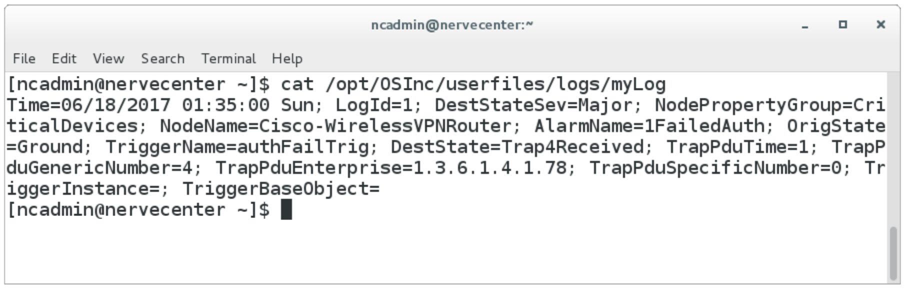
   - `server_name` should be the name of the current NerveCenter server
   - – for enterprise sends the trap with a default value
   - `node_name` should be the name of one of your nodes in the CriticalDevices property group
   - `4` for generic_trap sends an SNMP trap 4 (AuthFail)
   - `0` for specific_trap is required, since the generic trap is not an enterprise-specific trap (6)
   - – for time_stamp sends the trap with a default value
3. Press **Enter**.
4. Return to the NerveCenter client window. From the **Admin** menu, choose **Alarm Summary**.

   NerveCenter displays the Alarm Summary window. The instance for the alarm 1FailedAuth should appear in the Alarm Summary list.



Name	Time	Node	State	Severity	Trigger	Type	Source
1FailedAuth	06/18/20...	Cisco-WirelessVPNRouter	Trap4Received	Major	authFailTrig	mask	1CheckAuth

5. Locate your log file **myLog** at /opt/OSInc/userfiles/logs/ on the NerveCenter Server host. Open this file with an ASCII text editor such as Notepad (Windows) or vi (UNIX).



```
[ncadmin@nervecenter ~]$ cat /opt/OSInc/userfiles/logs/myLog
Time=06/18/2017 01:35:00 Sun; LogId=1; DestStateSev=Major; NodePropertyGroup=Cri
ticalDevices; NodeName=Cisco-WirelessVPNRouter; AlarmName=1FailedAuth; OrigState
=Ground; TriggerName=authFailTrig; DestState=Trap4Received; TrapPduTime=1; TrapP
duGenericNumber=4; TrapPduEnterprise=1.3.6.1.4.1.78; TrapPduSpecificNumber=0; Tr
iggerInstance=; TriggerBaseObject=
[ncadmin@nervecenter ~]$
```

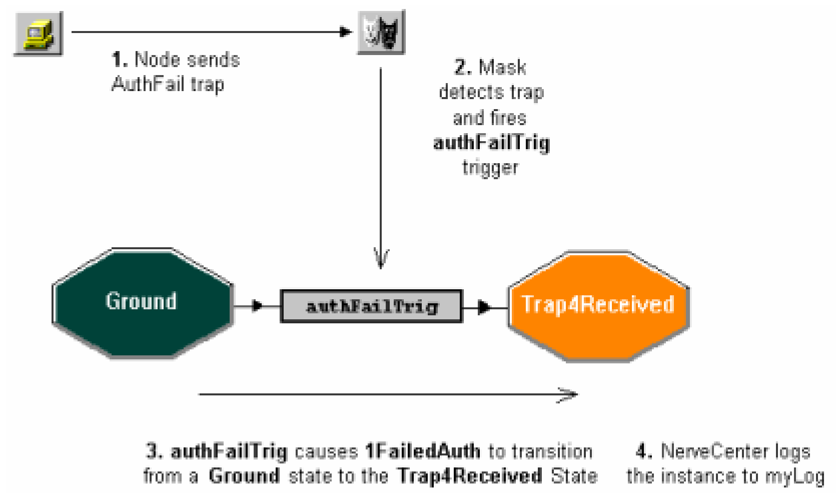
Figure 7: myLog as it Appears in Notepad

In the last activity, you specified that when the alarm 1FailedAuth transitioned from the Ground state to the Trap4Received state that it would perform the Log to File action. The myLog file represents the results of that action.

What is NerveCenter doing?

The SNMPv1AuthFailTrap.pl script uses trapgen to simulatean SNMPv1 Authentication Failure trap (Generic ID of '4').

1. The 1CheckAuth mask detects the trap 4 and fires the authFailTrig trigger.
2. The trigger authFailTrig causes the 1FailedAuth alarm to transition from the Ground state to the Trap4Received state.
3. NerveCenter performs the action associated with the transition. In this case, it logs the alarm instance to the file myLog.


1. Node sends AuthFail trap
2. Mask detects trap and fires authFailTrig trigger
Ground
authFailTrig
Trap4Received
3. authFailTrig causes 1FailedAuth to transition from a Ground state to the Trap4Received State
4. NerveCenter logs the instance to myLog

You have just created a behavior model that will log to a file any instance of an authentication failure on any of the nodes in the CriticalDevices property group. To do this you created a mask that detects SNMP Authentication Failure traps.

In the next activity you will modify this mask to enhance its filtering process, using the trigger function of the mask.

What are generic and enterprise-specific trap numbers?

In the last activity you generated an Authentication Failure trap. This trap was encoded and sent using the generic trap number 4 and the enterprise-specific number 0.

Generic trap numbers distinguish between seven major types of traps. The first six are SNMP-defined traps. Trap number 6 is reserved for enterprise-specific traps.

Enterprise-specific trap numbers are vendor-defined positive integers that identify particular traps. The meaning of specific trap numbers should be included in the documentation provided by the vendor of the device.

Defining a Trigger Function

In the previous activities you created a mask that would alert you when the agent of a node in your CriticalDevices property group sent an SNMP Authentication Failure trap. But suppose one of the devices in this group generates frequent authorization failures; the constant string of authorization traps may become annoying.

In this next activity you will use the trigger function of a mask to filter out all failed authorization traps for a particular node.

TO DEFINE A TRIGGER FUNCTION

1. From the **Admin** menu, choose **Mask List**.

   NerveCenter displays the Mask List window.

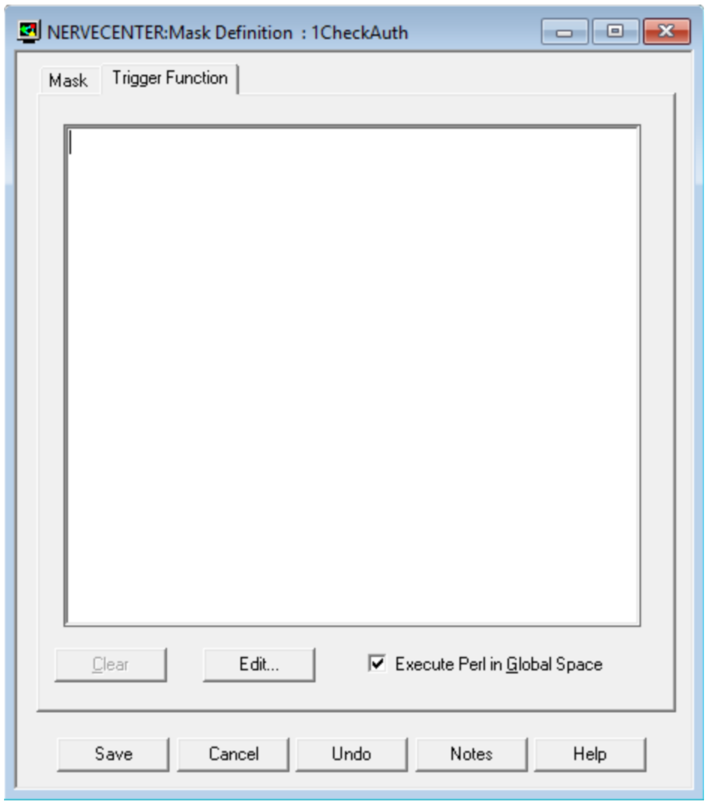
2. Highlight the **1CheckAuth** mask, and select **Open**.

   The Mask Definition window for 1CheckAuth appears. If 1CheckAuth is still enabled, all the fields will be grayed out.

3. If the mask is on, in the Enable frame, select **Off**.
4. In the Trigger Type area, select **Trigger Function**.

5. In the Mask Definition window, select the **Trigger Function** tab.

   The Trigger Function page appears.


NERVECENTER:Mask Definition : 1CheckAuth
Mask
Trigger Function
Clear
Edit...
Execute Perl in Global Space
Save
Cancel
Undo
Notes
Help

6. In the text box, type the following Perl script:

```
if ($NodeName ne “your_node_name”) {
FireTrigger (“authFailTrig”);
}
```

   Be sure to substitute the full name of one of your own managed nodes for *your_node_name* in the first line of the function.

Caution: If the full name of your node includes periods, they should be preceded by an escape character, which is the backward slash (\). For example: carey.open.com should be typed carey\.open\.com

This trigger function tests the name of the node that caused the trap and fires the trigger only if the node does not match the one you specified.

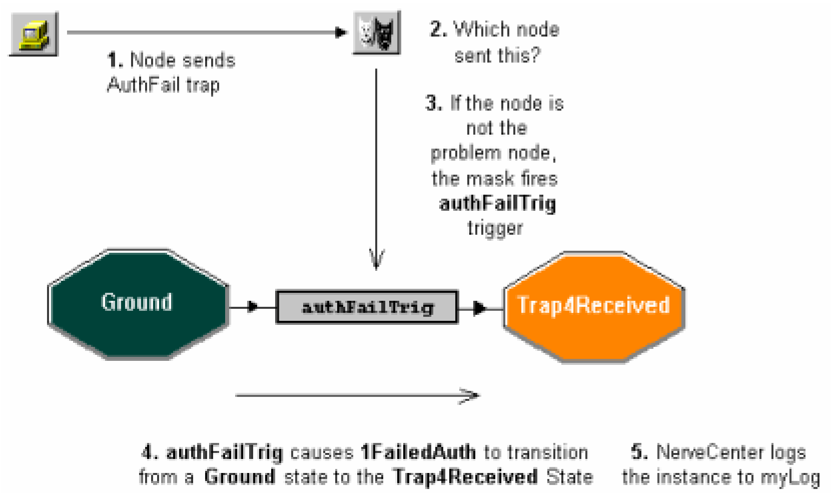
7. Select the **Mask** tab.
8. In the Enable frame, select **On**.
9. Select **Save**.

10. Open the Alarm Summary window.
11. Follow the steps in the last activity, Using Trapgen to Generate a Trap on page 62 to test the modified mask:
    a. First, generate a trap specifying for a_node_name the name of a device in the CriticalDevices group that is different from the node excluded by the trigger function. If everything is working properly, that alarm instance should appear in the Alarm Summary window.
    b. Next, generate a trap specifying for a_node_name the name of the device that you excluded by the trigger function. If everything is working properly, nothing should appear in the Alarm Summary window.

What is NerveCenter doing?

NerveCenter uses trapgen to simulate a node sending a generic 4 (AuthFail) trap.

1. The 1CheckAuth mask detects the Authentication Failure trap and determines if the specified source node is the problem node.
2. If the node is not the problem node, it fires the authFailTrig trigger.
3. The authFailTrig trigger causes the 1FailedAuth alarm to transition from the Ground state to the Trap4Received state.
4. NerveCenter performs the action associated with the transition. In this case, it logs the alarm instance to the myLog file.


1. Node sends AuthFail trap
2. Which node sent this?
3. If the node is not the problem node, the mask fires authFailTrig trigger
Ground
authFailTrig
Trap4Received
4. authFailTrig causes 1FailedAuth to transition from a Ground state to the Trap4Received State
5. NerveCenter logs the instance to myLog

You now know how to use alarms, polls, and masks, the main elements in NerveCenter behavior models. In "How to Use Behavior Models" on page 69 you will begin learning how to use behavior models to achieve monitoring of your network that is smart and relevant.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter.

1. Create an alarm that will notify you when a link is down and when it comes back up. Some of the actions you will take include:
   a. Creating a mask named 1LinkDown, that fires the simple trigger downLink when it detects an SNMP LinkDown trap
   b. Creating a mask named 1LinkUp, that fires the simple trigger upLink when it detects an SNMP LinkUp trap
   c. Creating an alarm named 1LinkDownUp, that includes the following transitions:

```
Ground --> downLink --> LinkDown

LinkDown --> upLink --> Ground
```

   d. Having the alarm log the instance to **myLog** at each transition
   e. Testing the masks and alarm by using trapgen

Note: This alarm is similar to the alarm IfLinkUpDown, included with NerveCenter. Generating generic traps may trigger it and other alarms.

2. Test these two Trap Masks and this Alarm by using the scripts provided at /opt/OSInc/Samples/trap-generation. From a command-line prompt on the NerveCenter Server host, enter these two commands:

```
# /opt/OSInc/Samples/trap-generation/SNMPv1LinkDown localhost 100
# /opt/OSInc/Samples/trap-generation/SNMPv1LinkUp localhost 100
```

3. These generate the SNMPv1 LinkDown and LinkUp trap notifications, respectively.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 6 Review Questions on page 123.

1. What are the similarities and differences between polls and masks?

2. What is the difference between a generic and an enterprise-specific trap number?

3. Why is it often necessary to use trapgen when testing a mask?

Summary of What You Learned

In this chapter you learned how to create, modify, and use trap masks in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to create a new mask
- How to create an alarm to be triggered by a mask
- How to use trapgen to generate a trap
- How to define a trigger function
- You also were introduced to the following concepts:
- Trap Mask
- Trap
- Generic and enterprise-specific trap numbers

How to Use Behavior Models

7

In the previous chapters, you learned how to create, modify, and enable several key objects in NerveCenter, such as polls, masks, and alarms. These elements are combined to create behavior models that monitor and correlate network events.

This chapter explains:

- What a behavior model is
- Two ways to develop behavior models to test for a persistent network condition
- How to create a behavior model to detect unresponsive nodes

What is a Behavior Model?

NerveCenter is designed to detect and correlate network conditions. You, the user, determine how NerveCenter will detect and react to those conditions by designing a set of specifications called a *behavior model*.

NerveCenter ships with some predefined behavior models. You can develop others to handle site-specific conditions.

Each behavior model has three components:

- Detecting network conditions

  Each behavior model is interested in a specific set of network conditions. NerveCenter detects these conditions actively by polling managed nodes. It also passively listens to SNMP traps that agents send, filtering out relevant traps with masks.

- Interpreting network conditions

  Once NerveCenter detects relevant conditions, the behavior model also defines how these conditions should be interpreted. NerveCenter uses one or more alarms to correlate such events. As network conditions develop, alarms move in and out of states the behavior model defines. An alarm state depicts the condition of your network.

- Responding to network conditions

  As certain conditions develop, NerveCenter will need to perform certain actions. The behavior model determines what actions should be taken and when they should occur. NerveCenter uses alarms and Action Router to define this component of the behavior model.

How to Create Useful Behavior Models

Over the course of the previous chapters you have created several behavior models. One behavior model detects high traffic. Another detects authorization failures. You also created other behavior models that were variations of these.

In this next scenario you don't want to be notified every time there is a problem. Instead, you want to know only when there is a persistent problem. Also, you want to know which nodes are not responding to NerveCenter's monitoring activities.

This scenario includes five activities:

1. Using a Counter to Detect Persistence below
2. Notifying of a Persistent Condition on page 73
3. Using a Timer to Detect Persistence on page 77
4. Detecting Unresponsive Nodes on page 82
5. Testing for an Unresponsive Node on page 84

Using a Counter to Detect Persistence

In a previous chapter you developed a behavior model for detecting high-traffic conditions on the CriticalDevices nodes.

This first activity will step you through the first stage of using a counter to check for the persistence of high traffic.

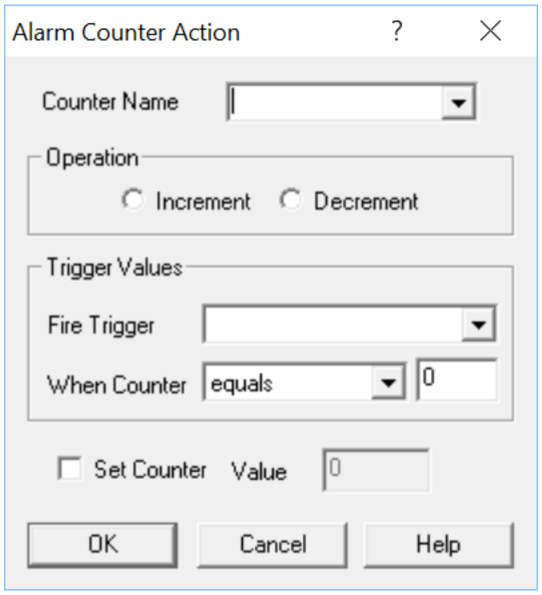
TO USE A COUNTER TO DETECT PERSISTENCE

1. Create a new alarm with the following characteristics:
   - Name: **1PersistentTraffic**
   - Property: **myNodes**
   - Scope: **SubObject**
2. In the state diagram, add a state naming it **Busy**.

   The state should have a traffic severity of Medium.
3. In the state diagram, add a transition to move from the Ground state to the Busy state when the trigger portTraffic is fired.
4. Add another transition to be triggered by portTraffic. This transition should be circular, transitioning from the Busy state to the Busy state.

5. In the Transition Definition window for the circular transition, select **New Action**. From the list of alarm actions, select **Alarm Counter**.

   NerveCenter displays the Alarm Counter Action window.



Alarm Counter Action

Counter Name

Operation
Increment
Decrement

Trigger Values
Fire Trigger
When Counter: equals 0

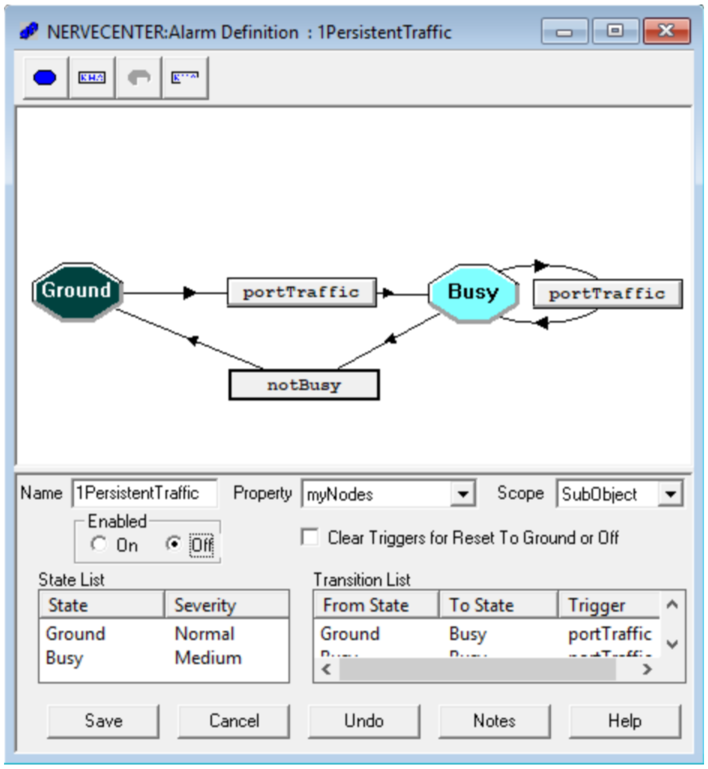
Set Counter Value 0

6. In the **Name** field, type **BusyCount**.
7. In the Operation area, select **Increment**.
8. In the **FireTrigger** field, type **tooBusy**.
9. In the **when counter equals** field, type **3**.

10. In the Alarm Counter Action window, select **OK**. In the Transition Definition window, select **OK**.

    The state diagram now includes a circular transition called portTraffic. Resize and position the icons to make the state diagram easier to read.

11. Add another transition to move from the Busy state to the Ground state when the notBusy trigger is fired.



Name 1PersistentTraffic Property myNodes Scope SubObject

Enabled: On [✓ Off]

Clear Triggers for Reset To Ground or Off

State List

State	Severity
Ground	Normal
Busy	Medium

Transition List

From State	To State	Trigger
Ground	Busy	portTraffic
Busy	Busy	portTraffic

Note: The notBusy trigger was created in Review and Summary on page 35. You must complete that exercise before you can create this second transition.

12. In the Transition Definition window, select **New Action**. From the list of alarm actions, select **Alarm Counter**.

    The Alarm Counter Action window is displayed.

13. In the **Counter Name** field, select **BusyCount**.
14. Check the **Set Counter** checkbox, leaving the new counter value at **0**.

    If the 1HighTraffic poll fires the notBusy trigger before the alarm counter BusyCount reaches three, the alarm will return to the Ground State. It will also return the count for BusyCount to zero. This way, the counter starts over any time the alarm receives a notBusy trigger.

15. In the Alarm Counter Action window, select **OK**. In the Transition Definition window, select **OK**.

    You may need to resize and position the icons to be able to make the state diagram easier to read.

16. In the Alarm Definition Window, select **Save**.

Note: It is important to save at this point so that NerveCenter has a chance to put the tooBusy trigger into its database. Otherwise you will not be able to use it in the next activity.

You have completed the first stage of creating a behavior model to detect persistent high traffic conditions. The next activity will step you through the process of creating an action to inform your network management platform of a high-traffic event.

What is a counter?

In the last activity you created a counter to detect persistent high traffic conditions on your network.

A *counter* tracks the same event over several occurrences to determine if a problem is persistent or just an isolated instance. Unlike a timer, a counter tests the frequency of an event.

NerveCenter alarms use the Alarm Counter action to both set and reset a counter.

Notifying of a Persistent Condition

In the last activity you began creating a behavior model with a counter that would test for the persistence of high-traffic conditions. Should traffic decrease on the interface before the counter reached three, the alarm would be returned to normal, and you would not be bothered.

This next activity steps you through the process of completing the behavior model. Should the high-traffic conditions persist, the alarm will move to a new state. It will also notify your network management platform using the alarm action inform.

Note: The portion of this activity from Step 3 to Step 6 assumes you are using NerveCenter to notify a network management platform. If you are using NerveCenter by itself, you may substitute for the Inform action the alarm action of your choice.

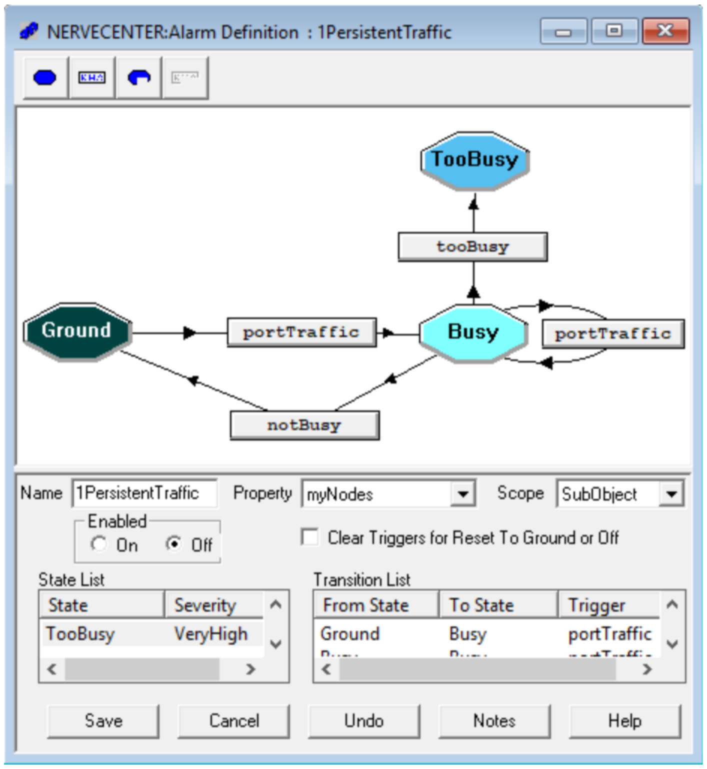
To Use Inform

1. From the toolbar of 1PersistentTraffic's state diagram, select **Add State**.

   Name the state **TooBusy** and assign it a traffic severity of **Very High**.

   Resize and position the icon as necessary.

2. Add a transition, which moves from the **Busy** state to the **TooBusy** state when the tooBusy trigger is fired.



Name: 1PersistentTraffic Property: myNodes Scope: SubObject

Enabled: On [✓ Off]

Clear Triggers for Reset To Ground or Off: ☐

State List

State	Severity
TooBusy	VeryHigh

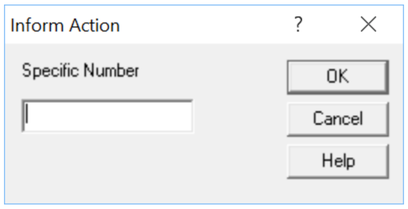
Transition List

From State	To State	Trigger
Ground	Busy	portTraffic

If the tooBusy trigger is not available, make sure you saved at the end of the activity Using a Counter to Detect Persistence on page 70.

3. In the Transition Definition window, select **New Action**. From the alarm action list, select **Inform**.

   The Inform Action window appears.



   The Inform action sends a trap-like message to either a network management platform or another NerveCenter. The only argument is a specific trap number.

   NerveCenter users are permitted to use any number in the range 100000 to 199999. Numbers below 100000 are used in predefined behavior models.

4. In the **Specific Number** text field, type the number **111111** or any other number in the appropriate range.

   If you do not enter a number in the **Specific Number** field, NerveCenter will assign a default number that will send a message that is not particularly informative.

5. In the Inform Action window, select **OK**. In the Transition Definition window, select **OK**.
6. In your network management platform, configure an event to display the message "Persistent high traffic detected" in the event browser.

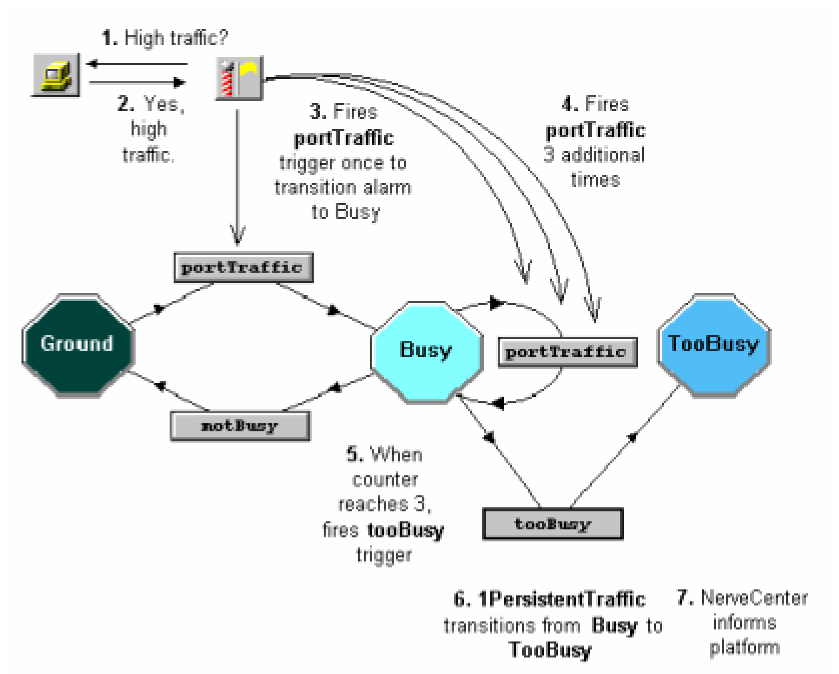
   The new event should be configured for the specific trap number 111111 or the number you entered in Step 4. See your network management platform's documentation for further instructions.

7. In the Alarm Definition window, select **Save**.
8. Turn the 1CheckTraffic poll and the 1PersistentTraffic alarm to **on**.

What is NerveCenter doing?

The 1CheckTraffic poll polls the nodes in the CriticalDevices group for high-traffic conditions.

1. The agents respond that the nodes are experiencing high-traffic conditions.
2. The 1CheckTraffic poll fires the portTraffic trigger, causing the 1PersistentTraffic alarm to transition from the Ground state to the Busy state. The counter BusyCount is set to 0.
3. Each time 1CheckTraffic fires the portTraffic trigger, the alarm transitions to the Busy state and the counter BusyCount increments by 1.
4. As soon as the counter BusyCount reaches 3, it fires the tooBusy trigger.
5. The tooBusy trigger causes 1PersistentTraffic to transition from the Busy state to the TooBusy state.
6. NerveCenter performs the Inform action. It is only at this point that your network management platform is notified of the event.


1. High traffic?
2. Yes, high traffic.
3. Fires portTraffic trigger once to transition alarm to Busy
4. Fires portTraffic 3 additional times
portTraffic
Ground
Busy
portTraffic
TooBusy
notBusy
5. When counter reaches 3, fires tooBusy trigger
tooBusy
6. 1PersistentTraffic transitions from Busy to TooBusy
7. NerveCenter informs platform

You have just completed creating a behavior model which will only notify you when a persistent problem occurs. The next activity will step you through the process of using another NerveCenter feature to detect persistence: the timer.

Using a Timer to Detect Persistence

In a previous chapter, you created a behavior model to detect unauthorized communication attempts. However, you don't want to be bothered every time a glitch in the network causes an Authentication Failure trap to be sent out. You want your behavior model to notify you only of repeated attempts to breach security.

Earlier in this chapter you used a counter to detect a persistent condition. In this activity you will use a timer.

TO USE A TIMER TO DETECT PERSISTENCE

1. Create a new alarm called 1PersistentAuthFail.

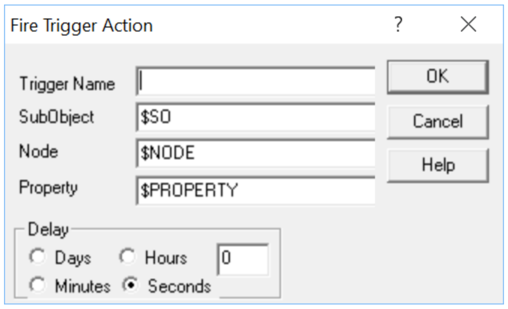
   Set the **Property** field to **myNodes** and the **Scope** field to Node.

2. In the alarm's state diagram, add a state.

   Name it Trap4Received and assign it a fault severity of Minor.

3. In the state diagram, add a transition that moves from the Ground state to the Trap4Received state when the trigger authFailTrig is fired.
4. In the Transition Definition window, select **New Action**. From the alarm action list, select **Fire Trigger**.

   The Fire Trigger Action window appears.

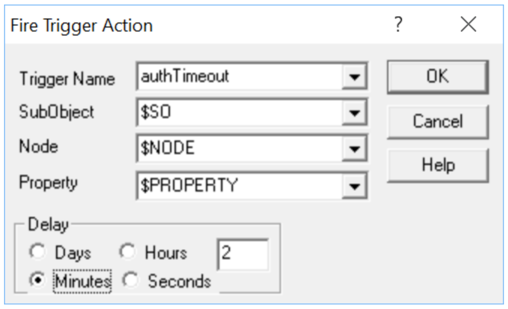


Trigger Name
SubObject $SO
Node $NODE
Property $PROPERTY
Delay
Days [ ] Hours [ ] 0
Minutes [ ] [✓ Seconds]

   Similar to polls and masks, alarms can fire triggers. This trigger will carry the name you give it, the name of the object instance, the node for the alarm instance, and the property associated with the alarm.

5. In the **Trigger Name** field, type **authTimeout**. In the Delay area, type **2** and select **Minutes**.

   When the behavior model receives its first authentication failure trap, it starts a two-minute timer. After two minutes, the alarm will fire the authTimeout trigger.



Trigger Name: authTimeout
SubObject: $SO
Node: $NODE
Property: $PROPERTY
Delay: ○ Days ○ Hours [● Minutes] ○ Seconds 2

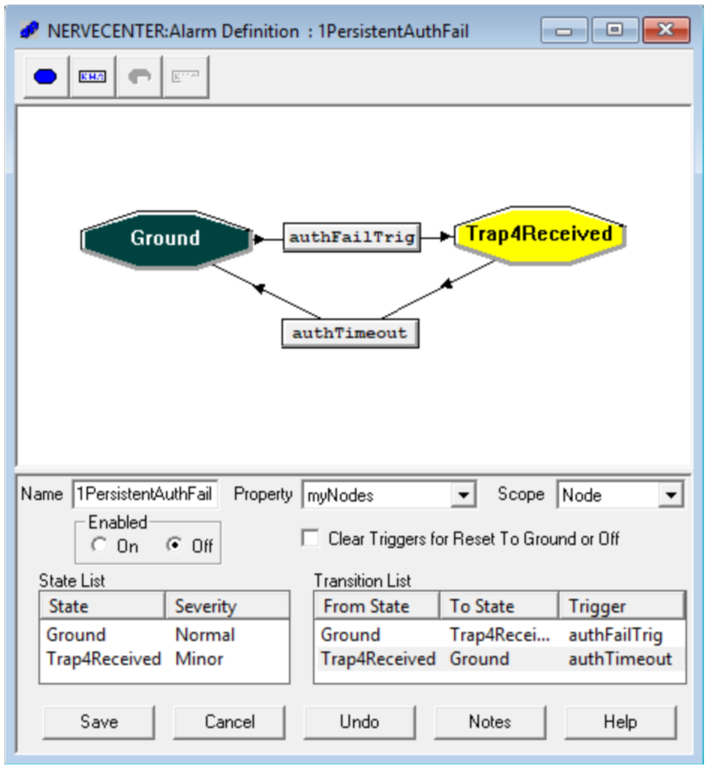
6. In the Fire Trigger Action window, select **OK**. In the Transition Definition window, select **OK**. In the Alarm Definition window, select **Save**.

   It is important to save at this point, so that you will be able to use the authTimeout trigger in other parts of the alarm.

7. Add a transition that moves from the Trap4Received state to the Ground state when the trigger authTimeout is fired.

   This transition moves the alarm back to ground two minutes after the first authentication failure.

   Resize and position the icons to make the state diagram easier to read.



Name 1PersistentAuthFail Property myNodes Scope Node

Enabled: On [✓ Off]

Clear Triggers for Reset To Ground or Off

State List

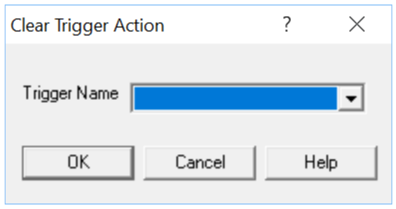
State	Severity
Ground	Normal
Trap4Received	Minor

Transition List

From State	To State	Trigger
Ground	Trap4Recei...	authFailTrig
Trap4Received	Ground	authTimeout

8. Add another transition that moves from Trap4Received to itself, caused by the authFailTrig trigger. Add to this transition the Inform action, using an appropriate trap number.
9. In the Transition Definition window, select **New Action**. From the alarm action list, select **Clear Trigger**.

The Clear Trigger Action window appears.



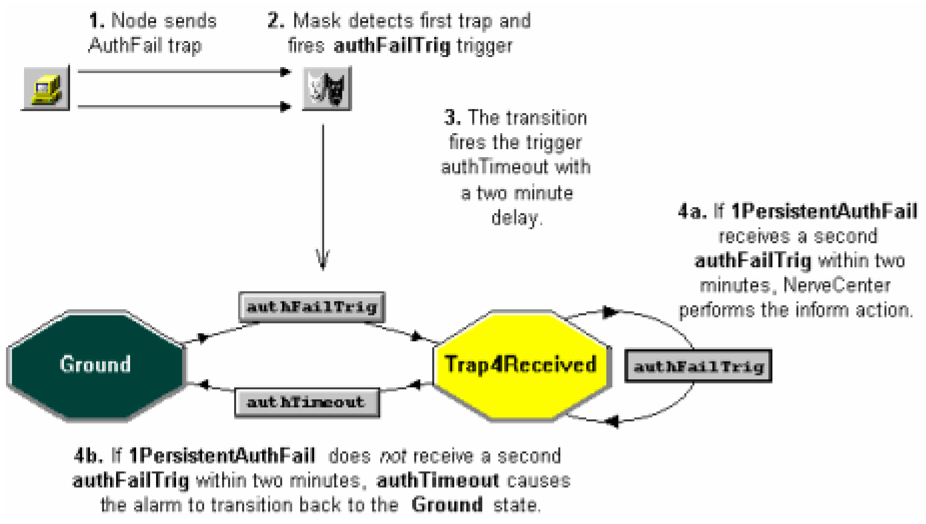
10. From the **Trigger Name** list, select **authTimeout**.

    Should another authentication failure occur for this node within two minutes of the first, this action will delete the authTimeout trigger. The alarm will not be able to return to the Ground state. The Inform action will occur for every authentication failure after the first.

11. In the Clear Trigger Action window, select **OK**. In the Transition Definition window, select **OK**. In the Alarm Definition window, select **Save**.
12. Turn the mask **1CheckAuth** and the alarm **1PersistentAuthFail** on.
13. Use the **trapgen** utility or the SNMPv1AuthFailTrap.pl script to simulate two Authentication Failure traps from the same node within two minutes of each other.

    The Alarm Summary window will display the progress of the 1PersistentAuthFail alarm.

Table 2: The Process that Occurs when 1PersistentAuthFail is Enabled

What is NerveCenter doing?
NerveCenter handles the arriving AuthFail traps as follows: 1. The mask 1CheckAuth detects the Authentication Failure trap and fires the trigger authFailTrig, causing the alarm 1PersistentAuthFail to transition to the Trap4Received state. 2. The transition fires the trigger authTimeout with a two-minute delay. 3. The next action depends on what occurs in the next two minutes: a. If another authFailTrig is fired, then the authTimeout trigger is deleted and NerveCenter performs the Inform action. b. If no other authFailTrig is fired, then the authTimeout trigger causes the alarm to return to the Ground state.  **1.** Node sends AuthFail trap **2.** Mask detects first trap and fires **authFailTrig** trigger **3.** The transition fires the trigger authTimeout with a two minute delay. **4a.** If **1PersistentAuthFail** receives a second **authFailTrig** within two minutes, NerveCenter performs the inform action. authFailTrig Ground Trap4Received authFailTrig authTimeout **4b.** If **1PersistentAuthFail** does *not* receive a second **authFailTrig** within two minutes, **authTimeout** causes the alarm to transition back to the **Ground** state.

You have now created two behavior models that inform your network management platform *only* when there is a persistent problem.

The next activity will explain how to create a behavior model that detects unresponsive nodes.

What is a timer?
In the last activity you created a timer to detect more than one failed authorization within a two-minute period. A *timer* detects the occurrence of the same event over a set period. Unlike a counter, a timer tests the duration of time between events. NerveCenter uses the Fire Trigger and Clear Trigger alarm actions to set and reset a timer.

Detecting Unresponsive Nodes

In previous activities you have created behavior models that monitor nodes within the CriticalDevices group. But what if one of the nodes in the group is not responding?

This activity steps you through the process of creating a behavior model to place unresponsive nodes into another property group.

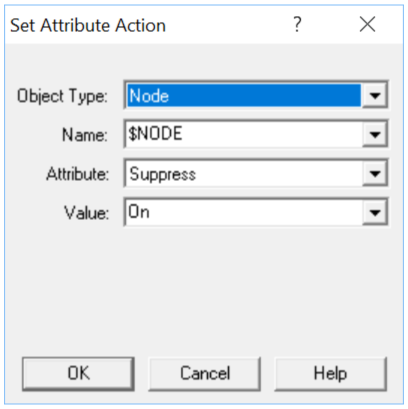
TO HANDLE UNRESPONSIVE NODES

1. Create a new alarm, naming it 1DeadNode. Set the **Property** field to **myNodes** and the **Scope** field to Node.
2. In the alarm's state diagram, add a state. Name it **NoResponse** and assign it a fault severity of **Warning**.

   Resize and position the icon to make the state diagram easier to read.
3. In the state diagram, add a transition that will move from the Ground state to the NoResponse state.
4. In the **trigger** field, select **ERROR**.

   The ERROR trigger is one of the predefined triggers provided by NerveCenter. Like all the built-in triggers, ERROR is in capital letters so that you can distinguish it from user-defined triggers. The ERROR trigger automatically fires whenever NerveCenter doesn't get a response from an SNMP get request when polling a device. This condition is usually interpreted to mean the device is down.
5. In the Transition Definition window, select **New Action**. From the alarm action list, select **Set Attribute**.

   The Set Attribute Action window appears.



Object Type: Node
Name: $NODE
Attribute: Suppress
Value: On

The Set Attribute action allows you to change certain attributes of a poll, mask, alarm, or node. In this case you want to assign a new property group to the unresponsive node.

6. In the Set Attribute Action window, set the **Object Type** field to **Node** and leave the **Name** field at the default **$NODE**. From the **Attribute** list, select **Property Group**.

   This tells NerveCenter to change the property group of the unresponsive node that instantiated the alarm.

7. From the **Value** list, select the **Troublemakers** property group.

Note: The Troublemakers property group was created in the Review Exercises on page 23. You must complete that exercise before you can define this action.

8. In the Set Attribute Action window, select **OK**. In the Transition Definition window, select **OK**.
9. In the Alarm Definition window, select **On** and **Save**.

You have just finished creating a behavior model that will assign a different property group to an unresponsive node. The next activity will explain how to test such a behavior model.

Testing for an Unresponsive Node

In the last activity you created a behavior model to detect an unresponsive node.

This activity will step you through the process of testing that behavior model.

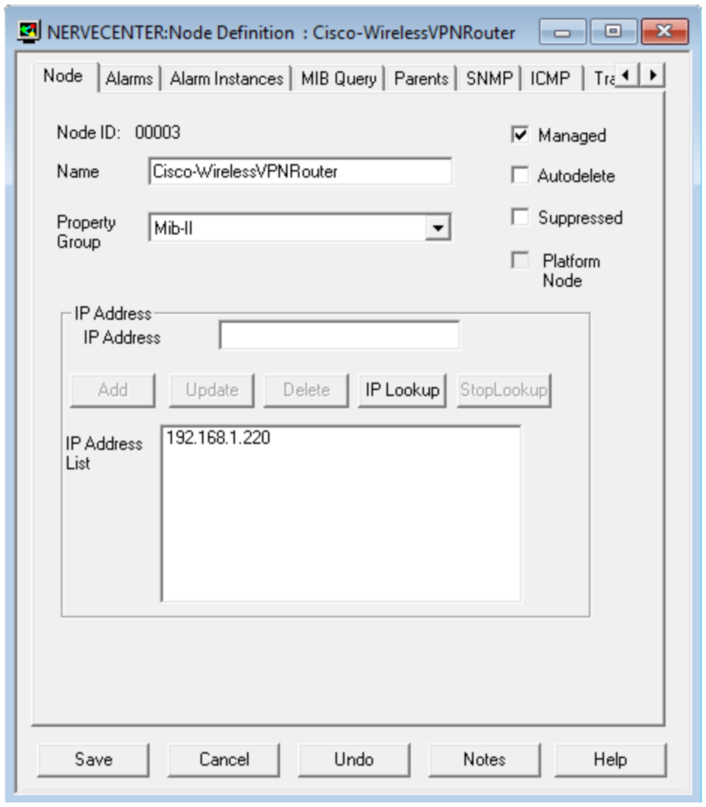
TO TEST FOR AN UNRESPONSIVE NODE

1. From the **Admin** menu, choose **Node List**.

   NerveCenter displays the Node List window.

2. In the Node List, highlight one of the devices in the CriticalDevices group. Select **Open**.

   The Node Definition window appears.



Node ID: 00003

Name: Cisco-WirelessVPNRouter

Property Group: Mib-II

[✓] Managed

[ ] Autodelete

[ ] Suppressed

[ ] Platform Node

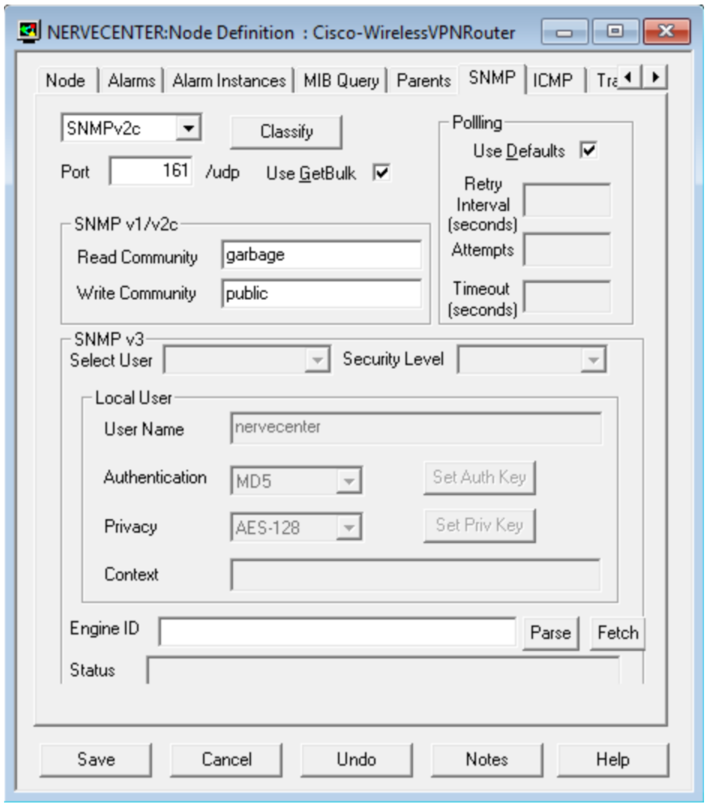
IP Address

IP Address:

IP Address List: 192.168.1.220

3. In the Node Definition window, select the **SNMP** tab.

   NerveCenter displays the SNMP page.



NERVECENTER:Node Definition : Cisco-WirelessVPNRouter

SNMPv2c

Port 161 /udp

Use GetBulk ✓

Polling: Use Defaults ✓

SNMP v1/v2c

Read Community: garbage

Write Community: public

SNMP v3

Local User

User Name: nervecenter

Authentication: MD5

Privacy: AES-128

4. In the **Read Community** field, delete the current value and type in **garbage**. Select **Save**.

   A bad community string causes an SNMP timeout error, which your new alarm 1DeadNode will interpret as your device being down.

Caution: Remember to write down the correct name of the Read Community. You will need it later to restore the node to its original setting.

5. Enable the **1CheckTraffic** poll and the **1HighTraffic** alarm.

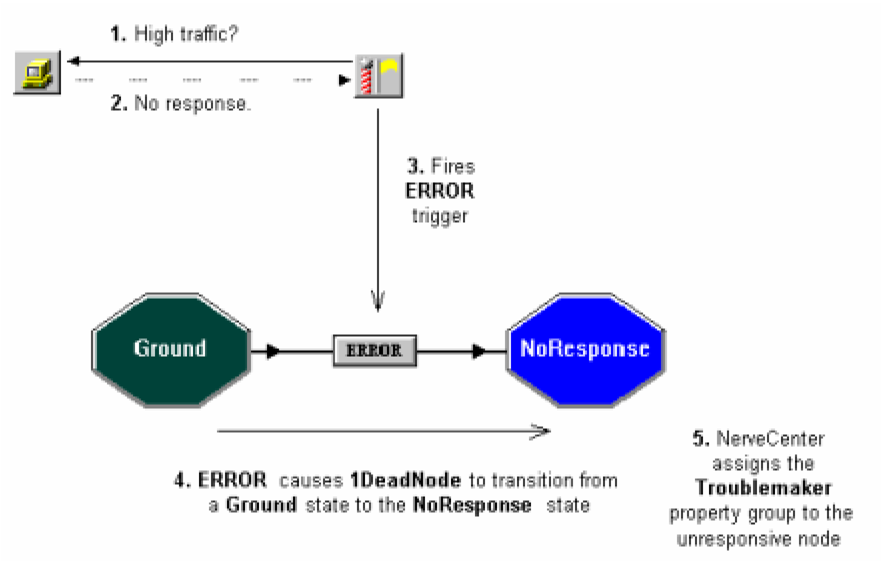
   The ERROR trigger will not be fired unless an unresponsive node is polled. This is why you must first turn on 1CheckTraffic. Because of smart polling, the 1CheckTraffic poll will not work unless an associated alarm, such as 1HighTraffic, is enabled.

6. In a few moments, examine the **Node** List. The device with the bad community string should now be in the Troublemakers property group.

What is NerveCenter doing?

The 1CheckTraffic poll polls the nodes in CriticalDevices for high traffic conditions.

- The node with the bad community string does not respond.
- NerveCenter fires an ERROR trigger.
- The 1DeadNode alarm transitions from the Ground state to the NoResponse state.
- NerveCenter assigns the property group Troublemakers to the unresponsive node.


1. High traffic?
2. No response.
3. Fires ERROR trigger
Ground
ERROR
NoResponse
4. ERROR causes 1DeadNode to transition from a Ground state to the NoResponse state
5. NerveCenter assigns the Troublemaker property group to the unresponsive node

1. Follow the instructions from Step through Step 4, to return the proper value to the node's **Read Community** field.
2. From the **Group** list, select **CriticalDevices**.

You have just created several useful behavior models designed to detect persistent conditions or unresponsive nodes. Each of these models involved a single level of alarms.

"How to Use Alarm Scope in Behavior Models" on page 89 will explain how to create multi-alarm behavior models so that you will have an even more refined approach to monitoring your network.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter.

1. Create an alarm that uses a counter to detect a node that is persistently unresponsive. Some of the actions you will take include:
    a. Creating a new alarm, naming it 1PersistentDeadNode, that includes the states **Ground**, **NoResponse**, **DeadNode**
    b. Adding a transition **Ground** > **ERROR** > **NoResponse**
    c. Adding a circular transition for the **NoResponse** state that starts an **Alarm Counter** that will fire the deadNode trigger after three transitions
    d. Adding a transition from the **NoResponse** state to the **Ground** state that removes the **Alarm Counter** should it receive NerveCenter's built-in **RESPONSE** trigger
    e. Adding a transition **NoResponse** > **deadNode** > **DeadNode** that uses the **Set Attribute** action to assign the **Troublemakers** property group to the unresponsive node
2. Assign a bad community name to a device and test the **1PersistentDeadNode** alarm.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 7 Review Questions on page 123.

1. Describe the three components of a behavior model.

____

____

____

2. When creating a behavior model with a timer component, why is the Clear Trigger action necessary?

____

____

____

3. When creating behavior models, what are the similarities and differences between counters and timers?

Summary of What You Learned

In this chapter you learned how to create useful behavior models in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to create behavior models that detect a persistent problem
- How to create a behavior model to detect unresponsive nodes
- How to use a counter in a behavior model
- How to use a timer in a behavior model
- How to use built-in triggers
- You also were introduced to the following concepts:
- Behavior model
- Counter
- Timer

How to Use Alarm Scope in Behavior Models

8

In previous chapters you learned how to create behavior models using NerveCenter objects such as alarms, polls, and masks. For certain circumstances you may want to create a behavior model that involves more than one alarm.

This chapter explains:

- What alarm scope is
- What a multi-alarm behavior model is
- How to use alarm scope in multi-alarm behavior models
- How to add reset capabilities to your behavior models

What is Alarm Scope?

Each alarm has a scope, which is a setting that determines which triggers are applied to that particular alarm. An alarm scope can be one of the following:

- *Enterprise Scope* manages one alarm instance for all the applicable nodes (that is, all nodes with the alarm's property) in the enterprise.
- *Node Scope* manages only one alarm instance for a single node.
- *SubObject Scope* manages multiple alarm instances for a single node. One instance is managed for each occurrence of the managed object. A subobject scope alarm is limited by both the object and the instance. Each interface, port, or other managed object can cause an alarm instance.
- *Instance Scope* tracks alarm instances for each instance of any managed object, regardless of the base object polled. An instance scoped alarm is limited by the instance, but not the object being tracked.

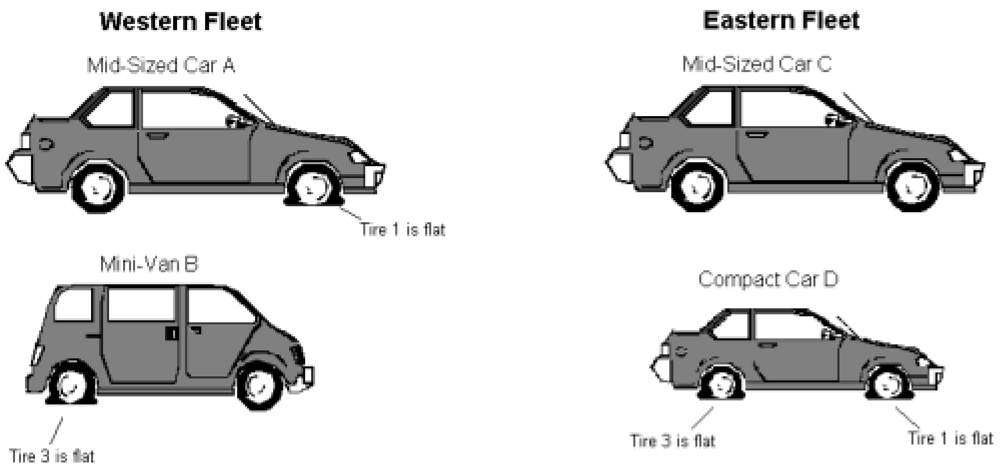
Another way to describe scope is to illustrate which trigger components need to match in order for an alarm to advance to the next state, as shown in Table 3.The scopes are listed on the left and the properties required to match the scope are listed across the top.

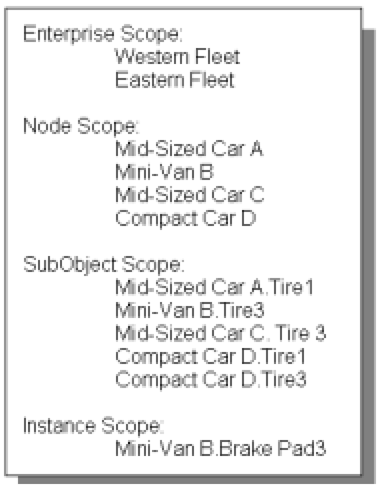
Table 3: Properties Needed to Match Scope

	Trigger Name	Property	Node	SubObject	Instance
Enterprise	✓	✓			
Node	✓	✓	✓		
Subobject	✓	✓	✓	✓	✓
Instance	✓	✓	✓		✓

For example, if you assign an instance scope to an alarm, the trigger name, property, node and instance must all match to advance the alarm to the next state. The subobject may vary.

Consider the following example: suppose a car rental company were to use NerveCenter to keep track of flat tires. Mid-size Car A and Mini-Van B have one flat and Compact Car D has two. How does varying the scope change the results?




Enterprise Scope:
Western Fleet
Eastern Fleet

Node Scope:
Mid-Sized Car A
Mini-Van B
Mid-Sized Car C
Compact Car D

SubObject Scope:
Mid-Sized Car A.Tire1
Mini-Van B.Tire3
Mid-Sized Car C. Tire 3
Compact Car D.Tire1
Compact Car D.Tire3

Instance Scope:
Mini-Van B.Brake Pad3

Figure 8: A Car Rental Company Uses Alarm Scope to Keep Track of Flats

Enterprise Scope (Fleet)

- Polls the Western or Eastern fleet to see if at least one flat tire exists.
- Does not care which car or which tire has a flat. Does not care how many flats exist.
- Notifies the tire repair company since the fleet contains at least one flat tire.

Enterprise Scope is not a good choice for monitoring flat tires, because you receive one notification whenever a tire is flat. If you are monitoring hundreds of cars, knowing that at least one car, but not which car or cars, has a flat tire is not helpful. However, Enterprise Scope is a good choice for monitoring car availability. If you were monitoring car availability, you would know that the Eastern Fleet has at least one car available.

Node Scope (Car)

- Returns information that Mid-Sized Car A has at least one flat, Mini-Van B has at least one flat, and Compact Car D has at least one flat.
- Does not care how many or which tires are flat.
- Notifies the reservation system that it is down one compact car, one mini-van and one mid-size car.

Node Scope is also not a good choice for monitoring flat tires because it doesn’t tell you what tire on the car is flat. However, Node Scope is useful for monitoring parts of the car of which there is only one, such as an engine or air conditioner. You could also poll to check if the insurance and registration for each car are valid.

SubObject Scope (Tire)

- Returns information that Mid-Size Car A and Mini-Van B have one flat and Compact Car D has two flats.
- Notifies the inventory department, so it can keep track of the number of spare parts needed for each car.

SubObject Scope is the best choice for monitoring flat tires because it provides the information you need to order new parts and let you know which cars are unavailable.

Instance Scope (Specific Tire and Specific Brake Pads)

- Returns information about the flat tires for Cars A, B and D, but can then monitor other conditions, such as wheel alignment, the condition of brake pads, or shocks.
- Notifies the service department of each condition detected.

Instance Scope is not a good choice for monitoring for flat tires, but it is very useful for seeing what else might be wrong if you have a flat tire.

As the above example illustrates, scope limits what an alarm monitors. The best scope depends on what information is important for your behavior model.

How to Use Alarm Scope in a Multi-alarm Behavior Model

In this next scenario you want to monitor a specific node within the CriticalDevices property group because it has been experiencing unusually high traffic. The overworked device has four key ports. You don't care to be notified when just one port is busy. However, you would like to know when all four ports are busy.

Creating just one alarm with SubObject scope will not be enough, since NerveCenter would then only track one instance on each individual port. Creating an alarm with Node scope will not be enough, since you need to track more than one instance on the same node.

For this scenario, you will need to create a multi-alarm behavior model that uses NerveCenter's alarm scope feature to keep track of the different instances.

The scenario includes the following three activities:

1. Creating the First Alarm of a Multi-alarm Behavior Model below
2. Creating the Second Alarm of a Multi-alarm Behavior Model on page 94
3. Adding Reset Capabilities to a Multi-alarm Behavior Model on page 96

Creating the First Alarm of a Multi-alarm Behavior Model

This first activity will step you through the process of creating an alarm that monitors your network at the SubObject Scope level.

TO CREATE THE FIRST ALARM OF A MULTI-ALARM BEHAVIOR MODEL

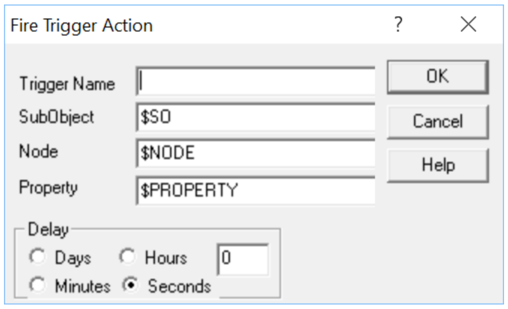
1. Create a new alarm, naming it **1BusyPort** and setting the property to **myNodes**.
2. Set the **Scope** field of the new alarm to **SubObject**.

   By setting the alarm scope to SubObject, you are telling 1BusyPort to track a separate instance for each port on the overworked node.

3. In the alarm's state diagram, add a state, naming it **BusyOnce** with a traffic severity of **Medium**.
4. Add another state, naming it **BusyTwice** with a traffic severity of **Very High**.
5. In the alarm's state diagram, add a transition that moves from Ground to BusyOnce when the trigger portTraffic is fired.
6. Add another transition that moves from BusyOnce to BusyTwice when the trigger portTraffic is fired.

7. In the Transition definition window, select **New Action**. Then, select **FireTrigger**.

   The Fire Trigger Action window appears.



Fire Trigger Action

Trigger Name	
SubObject	$SO
Node	$NODE
Property	$PROPERTY

Delay: Days Hours Minutes [✓ Seconds] 0

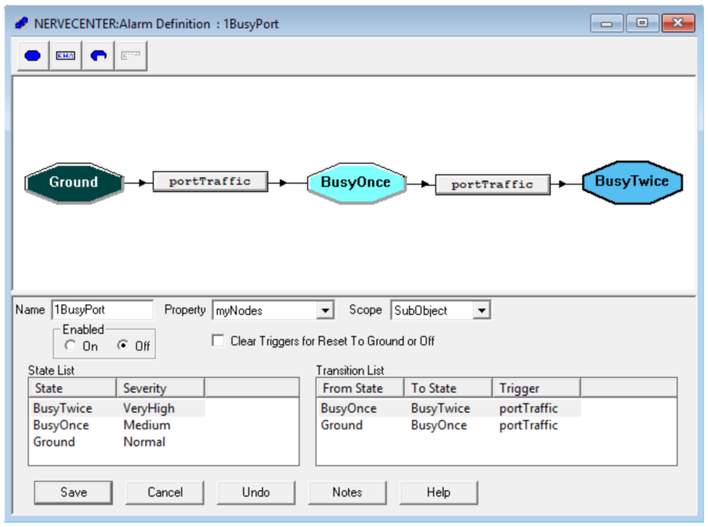
8. In the **Trigger Name** field, type **portTooBusy**.
9. Leave the NerveCenter defaults in the **SubObject**, **Node**, and **Property** fields.

   The defaults ensure that the resulting trigger will drive only alarm instances that monitor the same subobject and node as the current alarm instances.

10. Since there is no need for a timer in this behavior model, leave the delay time set to zero.
11. In the Fire Trigger Action window, select **OK**.

    The Fire Trigger Action is added to the transition's **Actions** list.

12. In the Transition Definition window, select **OK**.
13. Resize and position the state diagrams icons as needed.



NERVECENTER:Alarm Definition : 1BusyPort

Name 1BusyPort Property myNodes Scope SubObject

Enabled: On [✓ Off]

Clear Triggers for Reset To Ground or Off

State List

State	Severity
BusyTwice	VeryHigh
BusyOnce	Medium
Ground	Normal

Transition List

From State	To State	Trigger
BusyOnce	BusyTwice	portTraffic
Ground	BusyOnce	portTraffic

Figure 9: The Completed State Diagram for 1BusyPort

14. In the Alarm Definition window, select **Save**.

In this activity you created the first alarm of a multi-alarm behavior model. This alarm will fire a trigger called portTooBusy. However, no alarm is currently listening for this trigger. The next activity will step you through the process of creating the second alarm of this behavior model.

What is a multi-alarm behavior model?

In the last activity you created an alarm that was part of a multi-alarm behavior model.

A *multi-alarm behavior model* is a behavior model that contains two or more alarm definitions. A transition in one alarm instance fires triggers to cause other transitions in other alarm instances.

Creating the Second Alarm of a Multi-alarm Behavior Model

In the last activity you created the first alarm of a multi-alarm behavior model for monitoring traffic on an overworked node. In that alarm, high-traffic conditions on a single subobject (port) would cause the portTooBusy trigger to be fired.

This next activity will step you through the process of creating a second-level node scope alarm involving the portTooBusy trigger.

TO CREATE THE SECOND ALARM OF A MULTI-ALARM BEHAVIOR MODEL

1. Create a new alarm, naming it **1BusyNode**. You can leave the property at **NO_PROP**.

   Setting the property to myNodes is unnecessary since all the triggers for this alarm will be fired by the alarm created in the last activity.

2. Set the alarm's **Scope** field to **Node**.

   By selecting Node Scope, you are indicating that this alarm should keep track of instances for the entire node.

3. In the state diagram, add the following states.

   a. 1PortBusy with a traffic severity of Low

   b. 2PortsBusy with a traffic severity of Medium

   c. 3PortsBusy with a traffic severity of High

   d. 4PortsBusy with a traffic severity of Very High

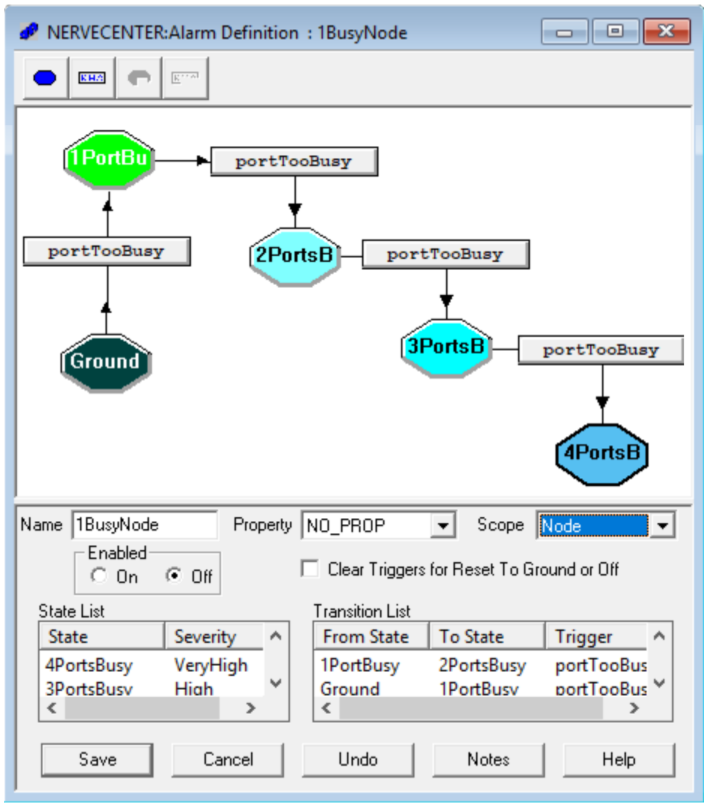
   If you add a new state and cannot see the state's icon, it may be hidden behind another state icon. Resizing and positioning your icons will make the state diagram easier to read.

4. In the state diagram, add a transition that moves from the Ground state to the 1PortBusy state, when the portTooBusy trigger is fired.
5. In the Transition Definition window, select **New Action**. From the alarm action list, select **Log to File**. In the **File Name** field, type BusyLog.
6. Repeat the last step to add the following transitions:

   a. 1PortBusy > portTooBusy > 2PortsBusy

   b. 2PortsBusy > portTooBusy > 3PortsBusy

7. Add another transition that moves from the 3PortsBusy state to the 4PortsBusy state when the portTooBusy trigger is fired.


NERVECENTER:Alarm Definition : 1BusyNode

Name 1BusyNode Property NO_PROP Scope Node

Enabled ( ) On (•) Off

☐ Clear Triggers for Reset To Ground or Off

State List

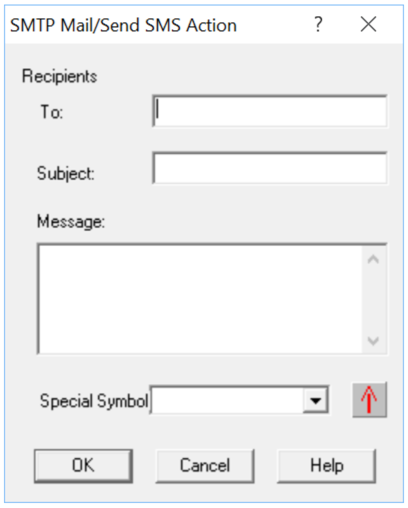
State	Severity
4PortsBusy	VeryHigh
3PortsBusy	High

Transition List

From State	To State	Trigger
1PortBusy	2PortsBusy	portTooBus
Ground	1PortBusy	portTooBus

8. In the Transition Definition window, select **New Action**. From the alarm action list, select **Send SMS**.

   The SMTP Mail/Send SMS Action window appears.


SMTP Mail/Send SMS Action

Recipients

To:

Subject:

Message:

Special Symbol

Note: Before you can use the Send SMS action, somebody must have configured NerveCenter to be able to connect to a SMTP mail service. If this has not been done, substitute an alarm action of your choice.

9. In the **To:** field, enter the SMTP gateway addressing for reaching your SMS target per the instructions of your carrier service.

   Optionally, fill the **Subject:** field. Any entry here will be included at the start of the resulting SMS message. As SMS messages need to be short (140 characters), this field can be skipped and left blank.

10. In the "Message:" field enter the content of the SMS message.

    The message text (limited to 140 characters, including the Subject you entered one) may be used as boilerplate wherein you insert a supported variable in the text, which is replaced at runtime with values taken from the live context.

    For this message, enter the sentence "Too many ports are busy on $NodeName ($NodeAddress)." When this occurs at runtime, NerveCenter will replace $NodeAddress and $NodeName. As you type in the message, you can either type the variable inclusions directly or select a variable from the **Special Symbol** list and then click the 'up' arrow to insert the selected variable into the text.

    A longer message could be "Too many ports are busy on $NodeName ($NodeAddress). Sent from $NCHostName for $AlarmName"

11. Select the **OK** button.

    The new action appears in the list of actions in the Transition Definition window.

12. Select the **OK** button in the Transition Definition window.
13. Select the **Save** button in the Alarm Definition window.

You have now created the second alarm of the multi-alarm behavior model to monitor high-traffic conditions on an overworked node.

However, what if the other ports are free again by the time the fourth port becomes too busy? The next activity will step you through the process of modifying the two alarms to allow for the behavior model to be reset should lower traffic conditions arise.

Adding Reset Capabilities to a Multi-alarm Behavior Model

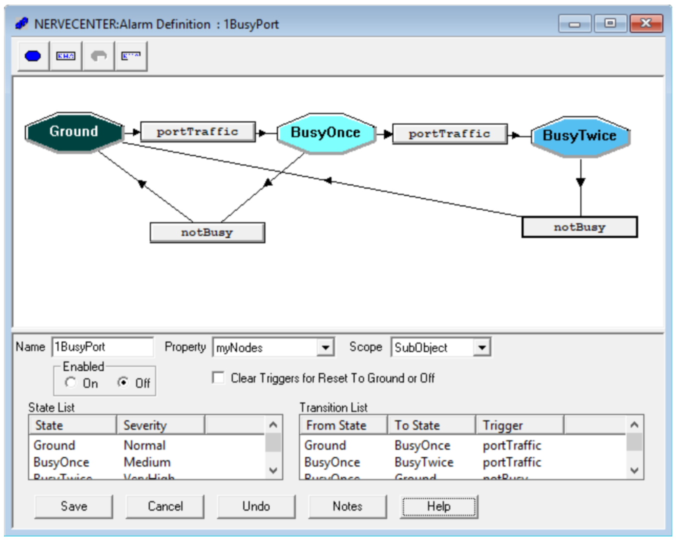
In the previous two activities you created a multi-alarm behavior model that will monitor high-traffic conditions on different ports of a single node.

NerveCenter will page you as soon as the fourth port on a node reports high-traffic conditions. However, what if traffic on one of the ports drops off? You need some way for your behavior model to reset itself to keep track of the ebb and flow of traffic conditions on your network.

This next activity will step you through the process of modifying the first and second alarm of your multi-alarm behavior model to allow for it to reset conditions.

To Add Reset Capabilities to a Multi-alarm Behavior Model

1. In the Alarm Definition window of 1BusyPort, add a transition that moves from BusyOnce to Ground when the trigger notBusy is fired.
2. Add another transition that moves from BusyTwice to Ground when the notBusy trigger is fired. Have the transition fire the backUpOne trigger.
3. In the Alarm Definition window, arrange the state diagrams so they are easier to read. Then select **Save**.



Name: 1BusyPort Property: myNodes Scope: SubObject

Enabled: On [✓ Off]

☐ Clear Triggers for Reset To Ground or Off

State List

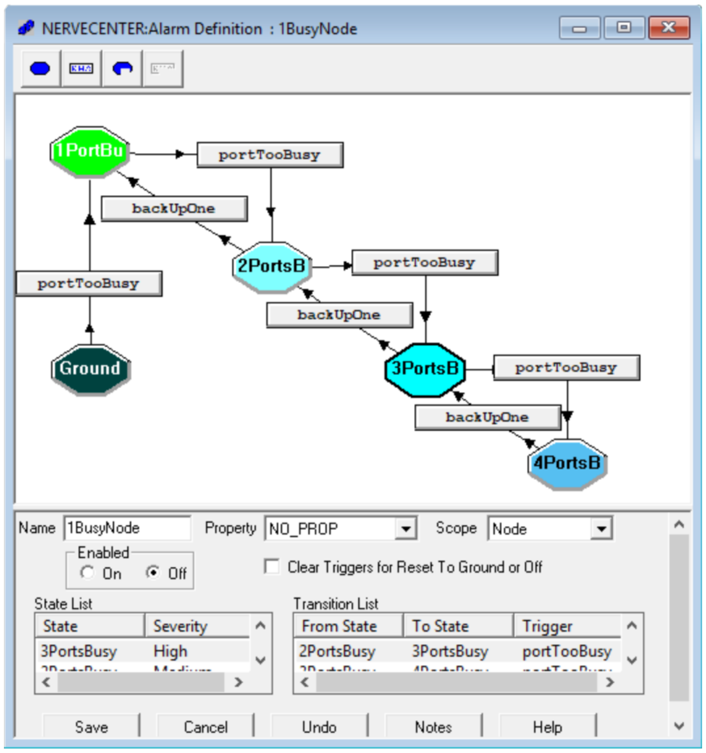
State	Severity
Ground	Normal
BusyOnce	Medium
BusyTwice	VeryHigh

Transition List

From State	To State	Trigger
Ground	BusyOnce	portTraffic
BusyOnce	BusyTwice	portTraffic
BusyOnce	Ground	notBusy

4. In the Alarm Definition window of 1BusyNode, add the following transitions:
   a. 4PortsBusy > backUpOne > 3PortsBusy
   b. 3PortsBusy > backUpOne > 2PortsBusy
   c. 2PortsBusy > backUpOne > 1PortBusy
   d. 1PortBusy > backUpOne > Ground

5. In the state diagram, resize and position the icons as necessary.



NERVECENTER:Alarm Definition : 1BusyNode

Name 1BusyNode Property NO_PROP Scope Node

Enabled On [✓ Off]

Clear Triggers for Reset To Ground or Off

State List

State	Severity
3PortsBusy	High

Transition List

From State	To State	Trigger
2PortsBusy	3PortsBusy	portTooBusy

6. In the Alarm Definition window, select **Save**.
7. Turn the 1CheckTraffic poll and the 1BusyPort and 1BusyNode alarms on.

   Unless one of the nodes in the CriticalDevices property group has four interfaces, the alarm 1BusyNode will not reach the final state.

You have just completed creating a sophisticated multi-alarm behavior model using alarm scope to limit NerveCenter's monitoring activities.

Table 4: The Process that Occurs when the Multi-Alarm Behavior Model is Enabled

What is NerveCenter doing?
The 1CheckTraffic poll polls the node for high-traffic conditions. 1. Each port on the node responds, causing 1CheckTraffic to fire the portTraffic trigger. 2. Since 1BusyPort is set to the SubObject scope, each port causes an alarm instance. 3. Each alarm instance fires the portTooBusy trigger. 4. Each instance of the portTooBusy trigger causes 1BusyNode to transition up one level. 5. As 1BusyNode transitions to 4PortsBusy, it performs the SMS action.

"How to Define Conditional Actions with Action Router" on page 103 will explain how to use the Action Router to tell NerveCenter when to perform certain actions.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter.

1. Modify this chapter's multi-alarm behavior model so that it notifies you whenever three nodes on your network are experiencing high traffic. Some of the actions you will take include:
   a. Modifying the 1BusyNode alarm so that, as it transitions into 4PortsBusy, rather than paging you it fires the trigger nodeTooBusy
   b. Creating a new alarm called 1BusyEnterprise
   c. Setting the Scope of the new alarm to **Enterprise**
   d. Adding three states, 1NodeBusy, 2NodesBusy, 3NodesBusy
   e. Adding the following transitions:

      Ground --> **nodeTooBusy** --> **1NodeBusy**

      1NodeBusy --> **nodeTooBusy** --> **2NodesBusy**

      2NodesBusy --> **nodeTooBusy** --> **3NodesBusy**
   f. Adding to the last transition an appropriate action to notify you of the high traffic on your network
2. Modify the alarms 1BusyNode and 1BusyEnterprise to be reset in the event that a node returns to normal traffic conditions.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 8 Review Questions on page 124.

1. What are the differences between using property and alarm scope to limit a behavior model?

2. Why was it necessary to use more than one alarm to create the multi-alarm behavior model in this chapter?

Summary of What You Learned

In this chapter you learned how to use alarm scope to create a multi-alarm behavior model in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to use alarm scope
- How to create a multi-alarm behavior model
- You also were introduced to the following concepts:
- Alarm scope
- Multi-alarm behavior model

How to Define Conditional Actions with Action Router

9

Up to this point, every behavior model you have created tells NerveCenter to perform its actions unconditionally. As long as the poll or mask detects the right conditions, the alarm will perform the associated action. NerveCenter's Action Router allows you to have NerveCenter perform certain actions conditionally.

This chapter explains:

- What Action Router does
- How to use Action Router to define conditional alarm actions
- How to use Action Router in a behavior model

What is Action Router?

All alarm actions we have seen so far have been unconditional. This means that whenever a transition causes an alarm action to be performed, the action is performed no matter what the circumstances.

Action Router is a feature that allows you to dictate which actions should be performed under particular circumstances. The node in question, the alarm that calls the action, or even the day of the week are all conditions that could determine whether an action is performed.

Action Router fits in a behavior model in the following way:

1. An alarm transitions into a new state, which performs the *Action Router* alarm action.
2. The Action Router facility checks the current conditions against user-defined *rule conditions*.
3. Each rule whose rule condition evaluates to true performs one or more *rule actions*.

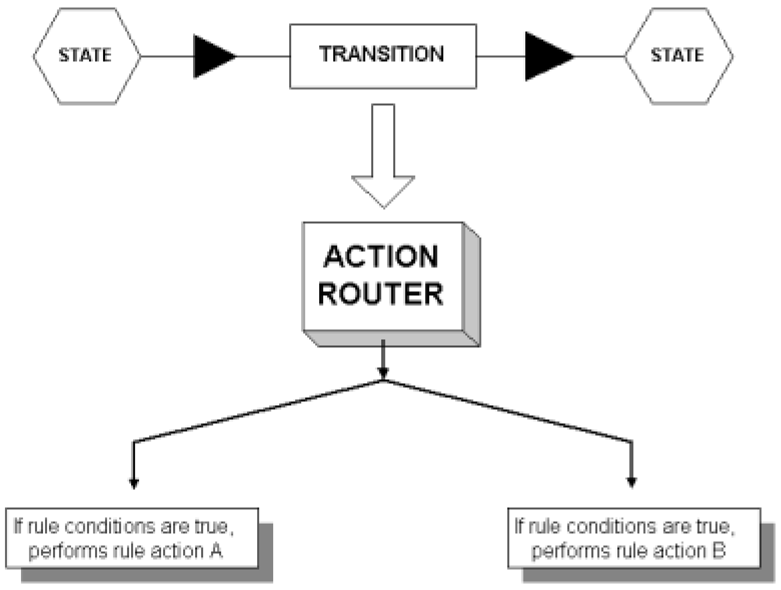

STATE
TRANSITION
STATE
ACTION ROUTER
If rule conditions are true, performs rule action A
If rule conditions are true, performs rule action B

Figure 10: The Role of Action Router within a Behavior Model

Whenever NerveCenter performs the Action Router action, all actions in the Action Router are carried out unless rule conditions have specified otherwise.

How to use Action Router

In this scenario, you keep receiving SMS messages for non-critical nodes. You want to reserve SMS messaging for times when specific nodes go down and use email for other nodes.

You will use Action Router to achieve this objective.

This scenario includes three activities:

1. Defining a Rule Condition in Action Router on the facing page
2. Defining a Rule Action in Action Router on page 108
3. Using Action Router in an Alarm on page 110

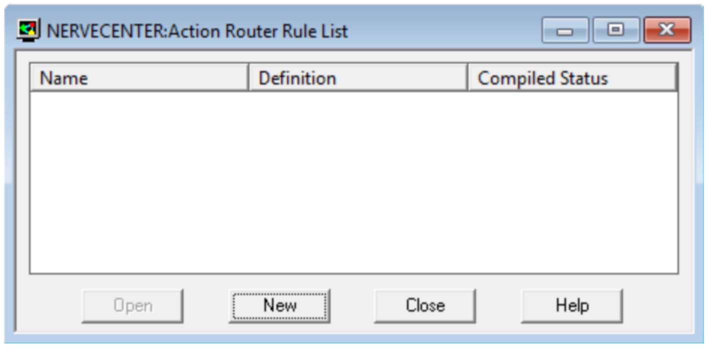
Defining a Rule Condition in Action Router

This first activity will step you through the process of defining a set of conditions under which the Action Router rule will perform an SMS message action.

To define a rule condition in Action Router

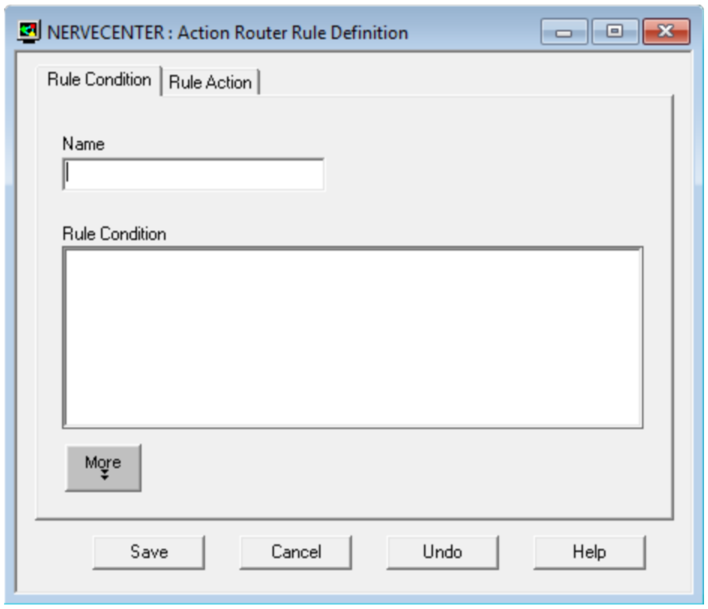
1. From the **Admin** menu, choose **Action Router Rule List**.

   NerveCenter displays the Action Router Rule List window.



2. In the Action Router Rule List window, select **New**.

   NerveCenter displays the Action Router Rule Definition window.



The Action Router Rule Definition window allows you to examine, change, and create a rule's conditions and actions.

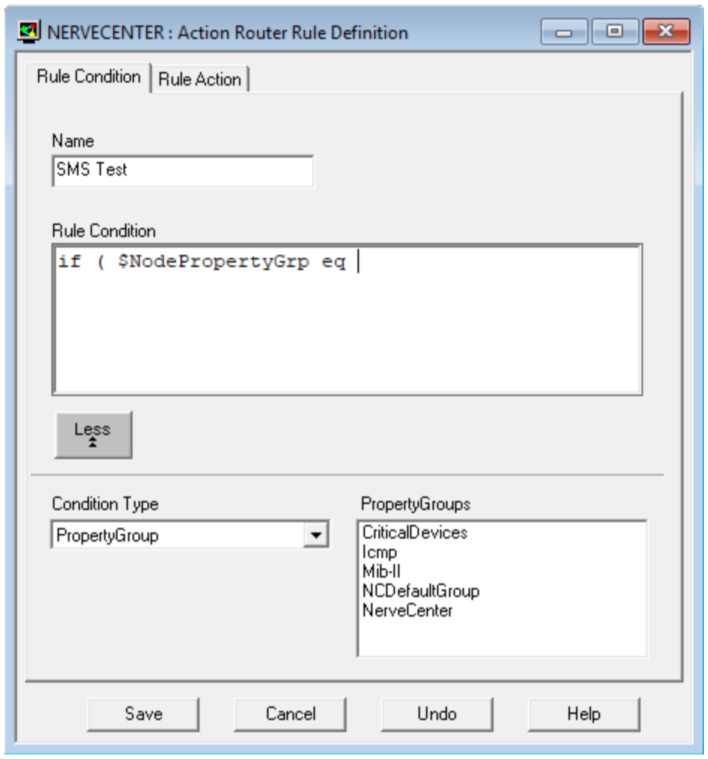
3. In the **Name** field, type **SMS Message Test**.

   This rule will determine by the property group of the node if NerveCenter should send an SMS message or send an email.

4. Type **if (** in the Rule Condition window.
5. Right-click in the Rule Condition window. From the pop-up menu, select **Node variables**. From the secondary pop-up menu, select **$NodePropertyGrp**.

   $NodePropertyGrp appears in the **Rule Condition** text field.

6. In the **Rule Condition** text field, type **eq** after $NodePropertyGrp.
7. Select **More** to expand the Rule Condition page.



Name: SMS Test

Rule Condition:

```
if ( $NodePropertyGrp eq
```

Condition Type: PropertyGroup

PropertyGroups:
CriticalDevices
Icmp
Mib-II
NCDefaultGroup
NerveCenter

The expanded Rule Condition page provides a list of all the available objects in the NerveCenter database, so that you do not need to worry about correctly typing the names.

8. In the **Condition Type** list, select **PropertyGroup**.

   A list of available property groups appears in the **PropertyGroups** text box.

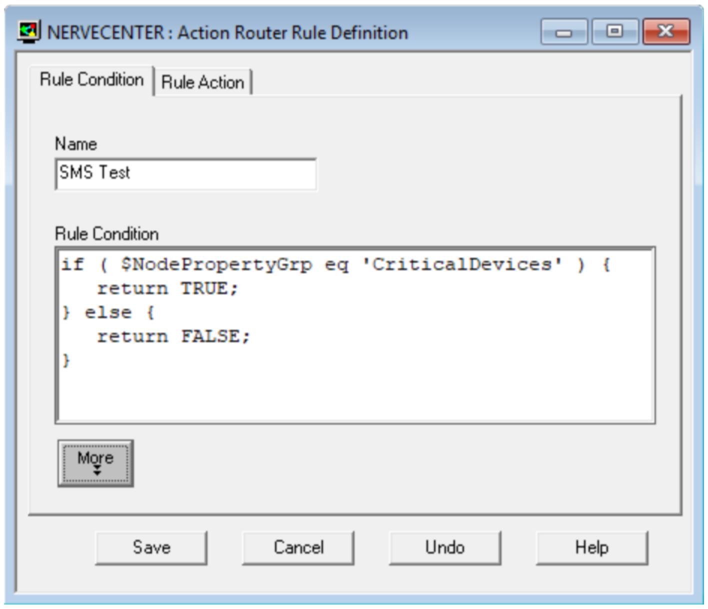
9. In the **PropertyGroups** list, double-click **CriticalDevices**.

   CriticalDevices appears in the expression in the **Rule Condition** text box.

10. Complete the Rule Condition to look like the following:

```
if( $NodePropertyGrp eq ‘CriticalDevices’) {
return TRUE;
}
else {return FALSE;
}
```

Note: As with building the poll condition expression, you can type all of this manually. This activity demonstrates some of the tools available to help you build a correct Rule Condition expression.



Name
SMS Test

Rule Condition

```
if ( $NodePropertyGrp eq 'CriticalDevices' ) {
   return TRUE;
} else {
   return FALSE;
}
```

Figure 11: The Completed Rule Condition of PagingTest

You have just completed building a Rule Condition that tells the Action Router to perform the Rule Action when the node does not belong to the property group CriticalDevices. The next activity will explain how you tell Action Router which actions to perform.

Defining a Rule Action in Action Router

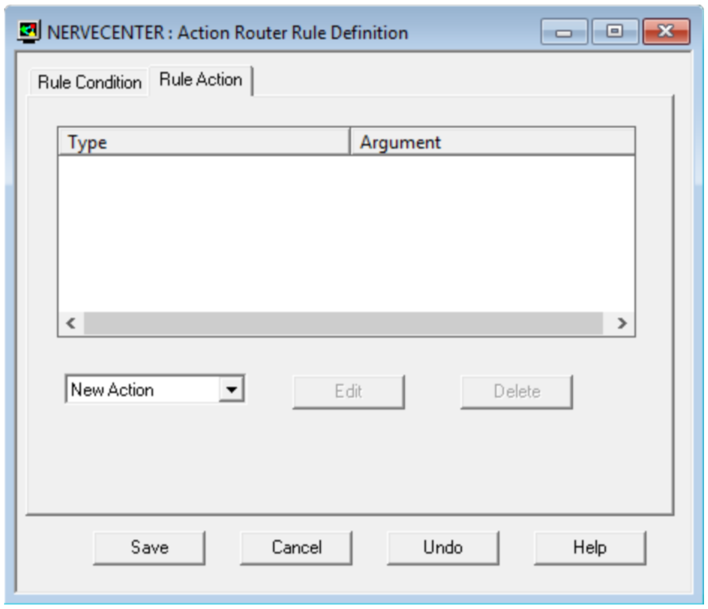
In the last activity you defined the conditions under which the Action Router would perform the actions of the SMS Message Test rule. You specified that it should carry out the rule actions if the node in question has the CriticalDevices property group.

This activity will step you through the process of defining the action that this rule will carry out should the rule conditions equal true.

TO DEFINE A RULE ACTION IN ACTION ROUTER

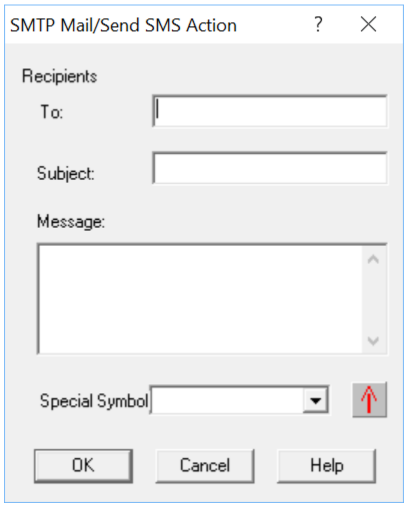
1. In the Action Router Rule Definition window, select the **Rule Action** tab.

   NerveCenter displays the Rule Action page.



2. In the Action Router Rule Definition window, select **New Action**. From the list of actions, select **Send SMS** .

   The SMTP Mail/Send SMS Action window appears.



Note: Before you can use the Send SMS action, somebody must have configured NerveCenter to be able to connect to a SMTP mail service. If this has not been done, substitute an alarm action of your choice.

3. In the **To:** field, enter the SMTP gateway addressing for reaching your SMS target per the instructions of your carrier service.

   Optionally, fill the **Subject:** field. Any entry here will be included at the start of the resulting SMS message. As SMS messages need to be short (140 characters), this field can be skipped and left blank.

4. In the “Message:” field enter the content of the SMS message.

   The message text (limited to 140 characters, including the Subject you entered one) may be used as boilerplate wherein you insert a supported variable in the text, which is replaced at runtime with values taken from the live context.

   For this message, enter the sentence “Too many ports are busy on $NodeName ($NodeAddress).” When this occurs at runtime, NerveCenter will replace $NodeAddress and $NodeName. As you type in the message, you can either type the variable inclusions directly or select a variable from the **Special Symbol** list and then click the ‘up’ arrow to insert the selected variable into the text.

   A longer message could be “Too many ports are busy on $NodeName ($NodeAddress). Sent from $NCHostName for $AlarmName”

5. Select the **OK** button.

   The new action appears in the list of actions in the Rule Definition window.

6. Select the **Save** button in the Alarm Definition window.
7. Select the **Save** button in the Action Router Rule Definition window.

   The rule now appears in the **Rule Definition** list.

The Action Router is now available to be used as an alarm action. The next activity will step you through the process of modifying an alarm to include this action.

Using Action Router in an Alarm

In the previous activity you defined a rule in Action Router that would page you only when the node in question is part of the CriticalDevices property group.

This activity will step you through the process of modifying the 1BusyNode alarm to include the Action Router action.

TO USE ACTION ROUTER IN AN ALARM

1. From the **Admin** menu, choose **Alarm Definition List**.

   NerveCenter displays the Alarm Definition List window.

2. Open the **1BusyNode** alarm.

   The Alarm Definition window for 1BusyNode appears.

3. In the state diagram, double-click on the portTooBusy transition between the 3PortsBusy state and the 4PortsBusy state.

   The Transition Definition window appears.

4. In the Transition Definition window, select **New Action**. From the alarm action list select **Action Router**.

   The Action Router action is added to the **Actions** list.

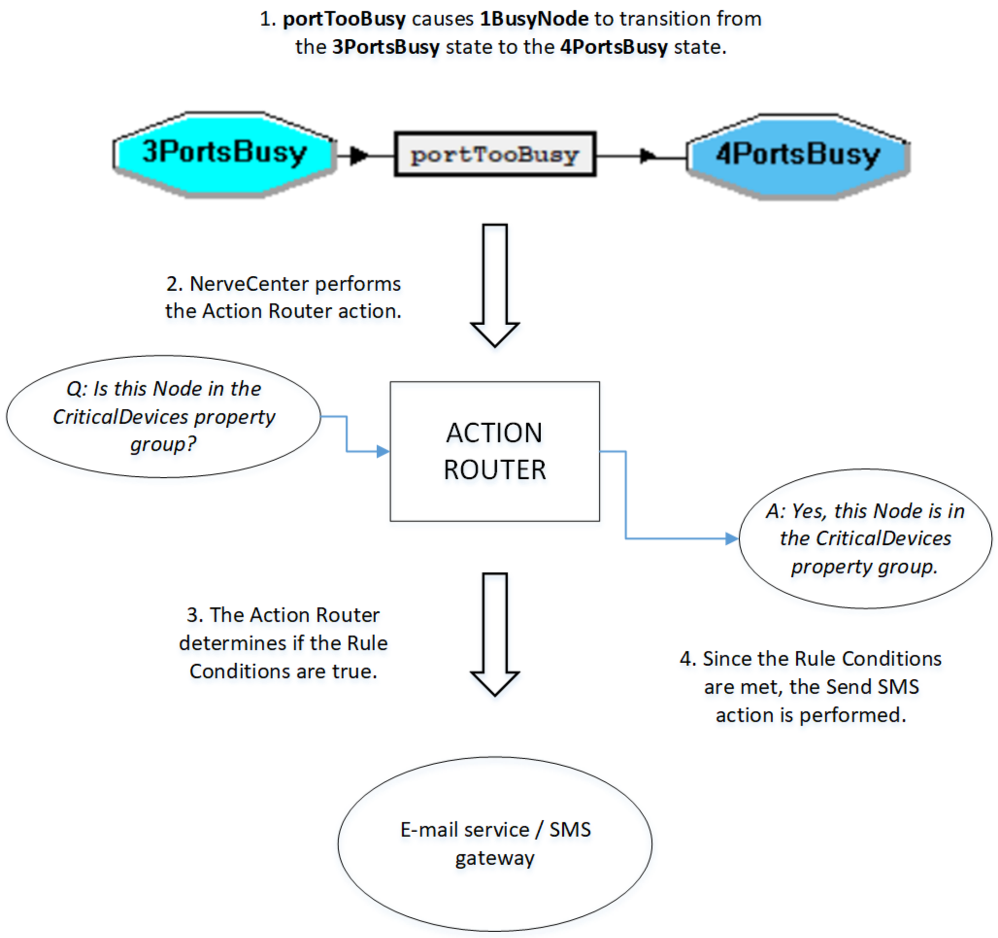
5. In the Transition Definition window, select **OK**.
6. In the Alarm Definition window, select **Save**.

   Now, any time one of the devices in the CriticalDevices property group becomes overwhelmed, in addition to sending the trigger nodeTooBusy to the alarm 1BusyEnterprise, it will also have the Action Router perform its actions.

What is NerveCenter doing?

High-traffic condition occurs on the fourth port on one of your nodes. The portTooBusy trigger causes the 1BusyNode alarm to transition from the 3PortsBusy state to the 4PortsBusy state.

1. The alarm performs the Action Router action.
2. Action Router determines if the current conditions correspond to the rule conditions, asking: Is the problem node part of the CriticalDevices property group?
3. Since the rule conditions evaluate to true, the rule actions are performed. In this case, you are sent an SMS message.


1. **portTooBusy** causes **1BusyNode** to transition from the **3PortsBusy** state to the **4PortsBusy** state.
3PortsBusy
portTooBusy
4PortsBusy
2. NerveCenter performs the Action Router action.
Q: Is this Node in the CriticalDevices property group?
ACTION ROUTER
A: Yes, this Node is in the CriticalDevices property group.
3. The Action Router determines if the Rule Conditions are true.
4. Since the Rule Conditions are met, the Send SMS action is performed.
E-mail service / SMS gateway

You have just used Action Router to define a conditional action in your behavior model.

"How to Reset Your Environment" on page 115 will explain how to reset your network environment.

Review and Summary

The following section includes exercises and questions to help you review what you have learned in this chapter.

Review Exercises

Complete the following exercises using the skills you learned in this chapter:

1. Have the Action Router email you if the problem node is not part of the CriticalDevices property group. Some of the actions you will take include:
   a. Naming the rule **MailTest**
   b. Setting the rule condition so that the node does not equal CriticalDevices
   c. Adding **SMTP Mail** as a new rule action
   d. Typing your email address in the **Receiver** text box

Note: Before you can use the SMTP action, someone must have configured NerveCenter to handle email actions correctly. If this has not been done, substitute an alarm action of your choice.

2. You want to clearly establish when you are responsible for messages and when your coworker is on call. You've suggested a schedule in which you'll receive SMS messages Sunday through Wednesday, and your coworker will be on call the rest of the week. Modify the rule condition of the SMS Message Test rule to work within this messaging schedule. Some of the actions you will take include:
   a. Add to the Rule Condition for SMS Message Test an expression telling Action Router that it is only to perform the SMS Message action Sunday through Wednesday.

Note: Remember to use the right-click and **More** button tools. Sunday is considered the first day of the week.

   b. Create a new rule called WeekendTest that resembles SMS Message Test except for the days of the week and the SMS Message destination.

Review Questions

Answer the following questions using the knowledge you learned in this chapter. See the Answers to the Chapter 9 Review Questions on page 124.

1. When does Action Router perform its actions?

____

____

____

2. What is the purpose of a rule’s Rule Condition?

3. When the Action Router facility is used, how many of its actions are performed?

Summary of What You Learned

In this chapter you learned how to use Action Router to perform conditional actions in NerveCenter. You also learned several new concepts. The list below should help you review once more before going to the next chapter..

You learned:

- How to define an Action Router rule condition
- How to assign Action Router rule actions
- How to use Action Router in an alarm
- You also were introduced to the following concepts:
- Action Router
- Rule condition
- Rule action

How to Reset Your Environment

As you have worked through this book, you have created quite a few polls, masks, and alarms. It is now time to return your NerveCenter environment to its original condition.

This scenario includes two activities:

1. "Deleting NerveCenter Objects" below
2. "Deleting Property Groups" on page 117

Deleting NerveCenter Objects

Normally you should not need to delete NerveCenter objects. Most of the time you will disable an object until you need it again.

To work through the activities and exercises found in this tutorial, you created fictitious alarms, polls, masks, and Action Router rules. Now that you have mastered the skills and concepts needed to create NerveCenter behavior models, these objects are unnecessary.

This next activity will step you through the process of deleting alarms, polls, and masks.

To delete NerveCenter objects

1. Close any open definition windows.
2. From the **Admin** menu, select **Alarm Definition List**.

   NerveCenter displays the Alarm Definition List window.

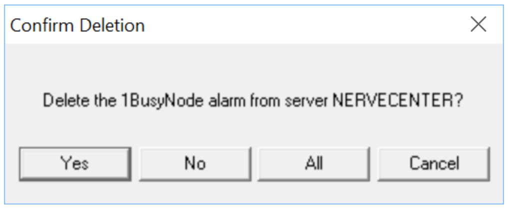
3. Hold down the Ctrl key and select each alarm you created while working though this tutorial.

   Each alarm should be highlighted as you select it.

Note: If you have used the names suggested in this tutorial, all of the alarms should begin with the number one. They should appear at the top of the list.

4. Right-click on one of the highlighted alarms. From the pop-up menu, select **Delete**.

   A Confirm Deletion window appears.



Delete the 1BusyNode alarm from server NERVECENTER?

5. In the Confirm Deletion window, select **Yes**.

   If you want to delete all the selected alarms at once, select **All**.

   The selected alarms disappear from the Alarm Definition List window.

6. From the **Admin** menu, select **Poll List**.

   NerveCenter displays the Poll List window.

7. Hold down the Ctrl key and select each poll you created while working though this tutorial.

   Each poll should be highlighted as you select it.

8. Right-click on one of the highlighted polls. From the pop-up menu, select **Delete**.

   A Confirm Deletion window appears.

9. In the Confirm Deletion window, select **Yes**.

   If you want to delete all the selected polls at once, select **All**.

   The selected polls disappear from the Poll List window.

10. From the **Admin** menu, select **Mask List**.

    NerveCenter displays the Mask List window.

11. Hold down the Ctrl key and select each mask you created while working though this tutorial.

    Each mask should be highlighted as you select it.

12. Right-click on one of the highlighted masks. From the pop-up menu, select **Delete**.

    A Confirm Deletion window appears.

13. In the Confirm Deletion window, select **Yes**.

    If you want to delete all the selected masks at once, select **All**.

    The selected masks disappear from the Mask List window.

14. From the **Admin** menu, select **Action Router Rule List**.

    NerveCenter displays the Action Router Rule List window.

15. Hold down the Ctrl key and select each Action Router rule you created while working through this tutorial.

    Each rule should be highlighted as you select it.

16. Right-click on one of the highlighted rules. From the pop-up menu, select **Delete**.

    A Confirm Deletion window appears.

17. In the Confirm Deletion window, select **Yes**.

    If you want to delete all the selected rules at once, select **All**.

    The selected rules disappear from the Action Router Rule List window.

You have now successfully deleted all of the alarms, polls, masks and Action Router rules. The next activity will explain how to delete a property group.

Deleting Property Groups

In the last activity you deleted all of the alarms, polls, masks, and Action Router rules you created while working through this tutorial.

You are now able to delete the property groups that you created.

Note: You cannot delete a property group until you have deleted all other objects that use that property group.

To Delete a Property Group

1. From the **Admin** menu, select **Node List**.

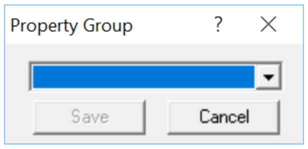
   NerveCenter displays the Node List window.

2. To have the nodes listed by Property Group, select the Group column header.
3. Hold down the Ctrl key and select each node that has a property group you created while working through this tutorial, such as CriticalDevices and Troublemakers.

   Each node should be highlighted as you select it.

4. Right-click on one of the highlighted nodes. From the pop-up menu, select **Property Group**.

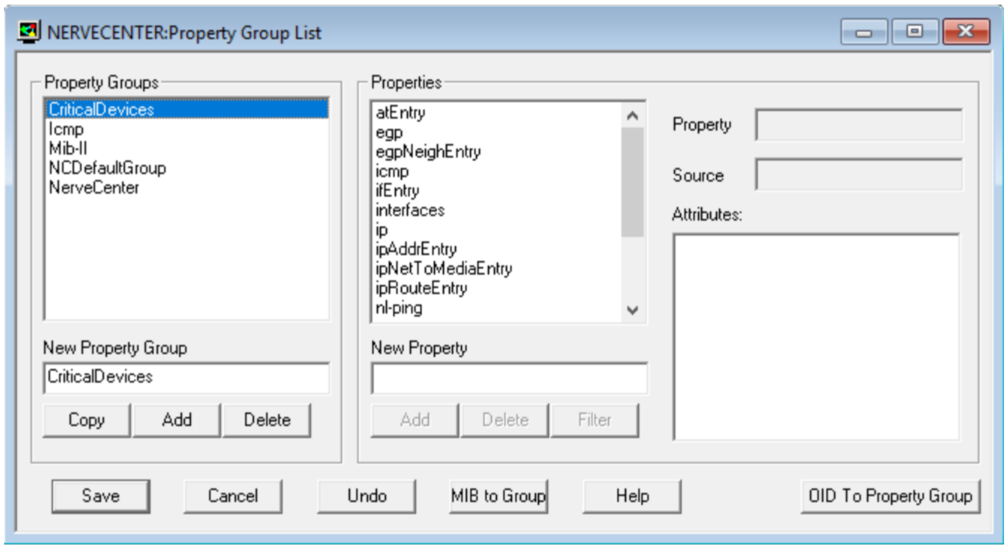
   The Property Group window appears.



5. In the Property Group window, select an appropriate property group.
6. Select **Save**.

7. From the **Admin** menu, select **Property Group List**.

   NerveCenter displays the Property Group List window.



8. Highlight one of the property groups you created for this tutorial.

   Unlike alarms, polls, and masks, you cannot select multiple property groups for deletion. You must delete each property group one at a time.

9. In the Property Group area, select **Delete**.

   The selected property group disappears from the **Property Group** list.

10. Repeat steps 2 and 3 to delete any other property groups you created.
11. In the Property Group List window, select **Save**.

    None of the property groups will be removed from the NerveCenter database until you select **Save**.

You have just completed deleting all the property groups you created for this tutorial.

Summary of What You Learned

Congratulations! You have covered a lot of ground. By completing the previous chapters, you have learned the following skills:

- Creating a new property group and a new property
- Assigning a property group to a particular set of nodes
- Creating, modifying, and enabling a new poll
- Creating, modifying, and enabling a new alarm
- Designing and modifying a state diagram
- Creating a new trap mask

- Using trapgen to generate a trap
- Defining a trigger function
- Creating behavior models that detect persistent problems
- Creating a behavior model to detect unresponsive nodes
- Using a counter in a behavior model
- Using a timer in a behavior model
- Using built-in triggers
- Using alarm scope
- Creating a multi-alarm behavior model
- Defining an Action Router rule condition and actions
- Using Action Router in an alarm
- Deleting NerveCenter objects

You also were introduced to the following concepts:

- Property group
- Property
- Poll
- Trigger
- Smart polling
- Alarm
- State diagram
- Transition
- Trap mask
- Trap
- Generic and enterprise-specific trap number
- Behavior model
- Counter
- Timer
- Alarm scope
- Multi-alarm behavior model
- Action Router
- Rule condition
- Rule action

Answers to Questions

A

This appendix contains the answers to the questions found in the Review and Summary sections of each chapter.

Answers to the Chapter 2 Review Questions

Following are the answers to the Review and Summary on page 13:

1. What are the three components of the NerveCenter client/server architecture?

   The three components of NerveCenter's architecture are the NerveCenter Server, the NerveCenter database, and the various NerveCenter user interfaces.

2. If you have administrator privileges, what can you do with the NerveCenter Client?

   Someone with administrator privileges can monitor active alarms, view an alarm's history, reset alarms, monitor the state of managed nodes, generate reports, create new behavior models, customize the predefined behavior models, manipulate objects in the NerveCenter database, and assign rights to a NerveCenter user.

Answers to the Chapter 3 Review Questions

Following are the answers to the Review Questions on page 23:

1. What is the purpose of property groups?

   Property groups and properties are used to limit the nodes which are monitored by behavior model objects.

2. What are the two types of properties?

   The two types of properties are the name of a MIB base object and a user-defined string.

Answers to the Chapter 4 Review Questions

Following are the answers to the Review Questions on page 35:

1. How can you configure a poll to monitor only the file servers on your network?

   A poll will monitor only those nodes whose property group contains its assigned property. To poll only file servers, you must create a property unique to your file servers and then assign that unique property to the poll.

2. What is a trigger?

   A trigger is a flag generated by NerveCenter objects that cause alarms to transition from one state to another.

3. What conditions must be met for a poll to be active?

   Smart polling ensures that NerveCenter will only poll a node if the poll is part of a behavior model designed to manage that node, can cause an immediate state transition in an alarm, and the poll's base object exists as a property in the node's property group.

Answers to the Chapter 5 Review Questions

Following are the answers to the Review Questions on page 23:

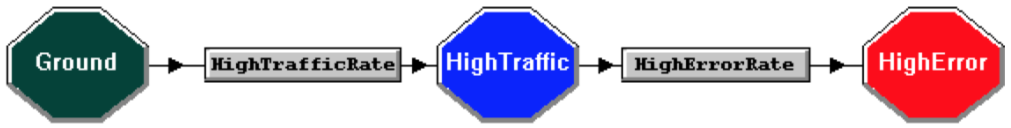
1. What causes an alarm to move from one state to another?

   An alarm moves from one state to another when a poll, mask, alarm, or other NerveCenter object fires a trigger whose key attributes match those of a pending alarm transition.

2. When is it unnecessary to specify a property for an alarm definition?

   It is unnecessary to specify a property for an alarm definition when all of the associated NerveCenter objects, such as polls or masks, are assigned that same property.

3. You need an alarm that first checks for high traffic on an interface and only then checks for a high error rate. Assume that you've already defined the necessary polls to collect the appropriate data. Draw an example of the state diagram below: (The names used in your diagram may differ, but the structure should be similar.)



Answers to the Chapter 6 Review Questions

Following are the answers to the Review Questions on page 35:

1. What are the similarities and differences between polls and masks?

   A trap mask is similar to a poll in its ability to trigger one or more alarms. Whereas a poll actively monitors conditions, the mask passively waits until it receives a relevant trap to act.

2. What is the difference between a generic and an enterprise-specific trap number?

   The first six generic trap numbers are SNMP-defined traps. Enterprise-specific trap numbers are vendor-defined numbers identifying particular conditions.

3. Why is it often necessary to use trapgen when testing a mask?

   If a mask needs testing, the trap it is masking will need to be generated. However, SNMP traps are unsolicited and sent in response to specific conditions on a network. Therefore, trapgen allows you to simulate a specific trap being sent.

Answers to the Chapter 7 Review Questions

Following are the answers to the Review Questions on page 53:

1. Describe the three components of a behavior model.

   A behavior model detects and interprets network conditions, and then responds to those conditions.

2. When creating a behavior model with a timer component, why is the Clear Trigger action necessary?

   A Clear Trigger action often prevents a trigger with a timer from returning an alarm to a previous state.

3. When creating behavior models, what are the similarities and differences between counters and timers?

   Both timers and counters are used in behavior models to detect persistence. A counter monitors for frequency of an occurrence, while a timer monitors the duration of time between events.

Answers to the Chapter 8 Review Questions

Following are the answers to the Review Questions on page 67:

1. What are the differences between using property and alarm scope to limit a behavior model?

   Property allows you to limit monitoring activity to a particular set of nodes. Alarm scope allows you to limit how many instances of alarms can occur.

2. Why was it necessary to use more than one alarm to create the multi-alarm behavior model in this chapter?

   It was necessary to create a multi-alarm behavior model because one alarm would not have been able to monitor the right conditions. The SubObject scope will only track alarm instances on each individual port. The Node scope would not allow NerveCenter to track more than one instance per node.

Answers to the Chapter 9 Review Questions

Following are the answers to the Review Questions on page 112:

1. When does Action Router perform its actions?

   Whenever a NerveCenter alarm performs the Action Router action, all actions in the Action Router are carried out.

2. What is the purpose of a rule's Rule Condition?

   A rule's Rule Condition tells the Action Router when it is to perform particular actions.

3. When the Action Router facility is used, how many of its actions are performed?

   When Action Router is used, all actions are performed that are not prohibited by criteria defined by the various rule conditions.

Index