

Open NerveCenter

Learning How to Create Behavior Models

UNIX and Windows

Disclaimer

The information contained in this publication is subject to change without notice. OpenService, Inc. makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. OpenService, Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this manual.

Copyright

Copyright © 1994-2004 OpenService, Inc. All rights reserved.

Trademarks

OpenService and NerveCenter are registered in the U.S. Patent and Trademark Office. The Open logo and Open NerveCenter are trademarks of OpenService, Inc. All other trademarks or registered trademarks are the property of their respective owners.

Printed in the USA, v4.0.

Open NerveCenter *Learning How to Create Behavior Models*

OpenService, Inc.

110 Turnpike Rd., Suite 308

Westborough, MA 01581

Phone 508-366-0804

Fax 508-366-0814

<http://www.openservice.com>

1 Introduction

Overview of this Book	2
NerveCenter Documentation	3
Using the Online Help	3
Printing the Documentation	3
The NerveCenter Documentation Library	4
UNIX Systems	5
Document Conventions	5
Documentation Feedback	6
Open Technical Support	7
Professional Services	7
Educational Services	7
Contacting the Customer Support Center	8
For telephone support:	8
For e-mail support:	8
For electronic support	8
For Online KnowledgeBase Access	8

2 How to Start Using NerveCenter Client

Before You Begin	10
What You Need to Know	11
What You Need to Use	11
What is the NerveCenter Client?	12
The Components of NerveCenter's Architecture	12
NerveCenter's User Interfaces	13
How to Use the NerveCenter Client	14
Starting the NerveCenter Client	14
Connecting to a NerveCenter Server	16
Quitting the NerveCenter Client	17
Review and Summary	18
Review Exercises	19
Review Questions	19

	Summary of What You Learned	20
3	How to Define Property Groups and Properties	
	What is a Property Group?	22
	How to Create Property Groups and Properties	23
	Creating a New Property Group	24
	Creating a New Property	26
	Assigning a Property Group to a Set of Nodes	27
	Review and Summary	30
	Review Exercises	30
	Review Questions	30
	Summary of What You Learned	31
4	How to Use Polls	
	What is a Poll?	34
	How to Create a Poll	34
	Creating a New Poll	35
	Writing a Poll Condition	37
	Enabling a Poll	40
	Modifying a Poll Condition	41
	Review and Summary	44
	Review Exercises	44
	Review Questions	44
	Summary of What You Learned	46
5	How to Use Alarms	
	What is an Alarm?	48
	How to Create an Alarm	49
	Creating a New Alarm	49
	Designing a State Diagram	52
	Enabling an Alarm	55
	Modifying an Alarm's State Diagram	58
	Adding Another State to an Alarm	60
	Review and Summary	62
	Review Exercises	62
	Review Questions	63
	Summary of What You Learned	64

6 How to Use Trap Masks

What is a Trap Mask?	66
How to Create a Trap Mask.	67
Creating a New Trap Mask.	67
Creating an Alarm to be Triggered by a Mask.	70
Using Trappgen to Generate a Trap.	73
Defining a Trigger Function.	76
Review and Summary	79
Review Exercises	79
Review Questions.	80
Summary of What You Learned.	81

7 How to Use Behavior Models

What is a Behavior Model?	84
How to Create Useful Behavior Models	85
Using a Counter to Detect Persistence	85
Notifying of a Persistent Condition	88
Using a Timer to Detect Persistence	91
Detecting Unresponsive Nodes	94
Testing for an Unresponsive Node.	96
Review and Summary	99
Review Exercises	99
Review Questions.	100
Summary of What You Learned.	101

8 How to Use Alarm Scope in Behavior Models

What is Alarm Scope?	104
Enterprise Scope (Fleet)	106
Node Scope (Car)	106
SubObject Scope (Tire)	106
Instance Scope (Specific Tire and Specific Brake Pads)	107
How to Use Alarm Scope in a Multi-alarm Behavior Model	107
Creating the First Alarm of a Multi-alarm Behavior Model	108
Creating the Second Alarm of a Multi-alarm Behavior Model	110
Adding Reset Capabilities to a Multi-alarm Behavior Model	112
Review and Summary	115
Review Exercises	115
Review Questions.	116

	Summary of What You Learned	116
9	How to Define Conditional Actions with Action Router	
	What is Action Router?	118
	How to use Action Router	119
	Defining a Rule Condition in Action Router	119
	Defining a Rule Action in Action Router	123
	Using Action Router in an Alarm	125
	Review and Summary	127
	Review Exercises	127
	Review Questions	128
	Summary of What You Learned	129
10	How to Reset Your Environment	
	How to Reset Your Environment	131
	Deleting NerveCenter Objects	132
	Deleting Property Groups	134
	Summary of What You Learned	136
A	Answers to Questions	
	Answers to the Chapter 1 Review Questions	140
	Answers to the Chapter 2 Review Questions	140
	Answers to the Chapter 3 Review Questions	141
	Answers to the Chapter 4 Review Questions	142
	Answers to the Chapter 5 Review Questions	143
	Answers to the Chapter 6 Review Questions	143
	Answers to the Chapter 7 Review Questions	144
	Answers to the Chapter 8 Review Questions	144
	Index	145

Welcome to *Learning How to Create Behavior Models*. This chapter introduces the audience and purpose of this guide, and how you can best use it.

This chapter includes the following sections:

Section	Description
<i>Overview of this Book</i> on page 2	Includes an overview of the contents of this guide and what you need to know before you use the guide.
<i>NerveCenter Documentation</i> on page 3	Lists and describes the components of the Open NerveCenter support system, including printed guides, online guides, help, and links to the Open NerveCenter Web site and the Open technical support Web site.
<i>Open Technical Support</i> on page 7	Describes how to access the NerveCenter knowledge base and other Open support services.

Overview of this Book

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

Learning How to Create Behavior Models contains the following sections:

Title	Description
Chapter 2, <i>How to Start Using NerveCenter Client</i>	Explains how to start the NerveCenter Client and connect to a NerveCenter Server.
Chapter 3, <i>How to Define Property Groups and Properties</i>	Explains property groups and their use within NerveCenter behavior models.
Chapter 4, <i>How to Use Polls</i>	Explains polls and their use within NerveCenter behavior models.
Chapter 5, <i>How to Use Alarms</i>	Explains alarms and their use within NerveCenter behavior models.
Chapter 6, <i>How to Use Trap Masks</i>	Explains trap masks and their use within NerveCenter behavior models.
Chapter 7, <i>How to Use Behavior Models</i>	Explains how to create useful behavior models which test for persistence.
Chapter 8, <i>How to Use Alarm Scope in Behavior Models</i>	Explains alarm scope and how to create multi-alarm behavior models.
Chapter 9, <i>How to Define Conditional Actions with Action Router</i>	Explains how to use the NerveCenter Action Router to have behavior models perform actions conditionally.
Chapter 10, <i>How to Reset Your Environment</i>	Explains how to reset your environment following the completion of this tutorial.
Appendix A, <i>Answers to Questions</i>	Answers the questions found in the Review and Summary sections of each chapter.

NerveCenter Documentation

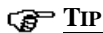
This section describes the available NerveCenter documentation, which explains important concepts in depth, describes how to use NerveCenter, and provides answers to specific questions.


The documentation set is provided in online (HTML) format, as well as PDF for printing or on-screen viewing. See the following topics for more information:

- ◆ *Using the Online Help* on page 3
- ◆ *Printing the Documentation* on page 3
- ◆ *The NerveCenter Documentation Library* on page 4
- ◆ *UNIX Systems* on page 5
- ◆ *Document Conventions* on page 5
- ◆ *Documentation Feedback* on page 6

Using the Online Help

You can use Microsoft Internet Explorer, Mozilla, or Netscape Navigator to view the documentation. Refer to the *NerveCenter 4.0.3 Release Notes* for the browser versions supported with this release.

**TIP**

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 <http://www.adobe.com>.

The NerveCenter Documentation Library

The following documents ship with NerveCenter.

Book Title	Description	Application	Audience	PDF for Print
<i>NerveCenter 4.0.3 Release Notes</i>	Describes new NerveCenter v4.0 features and includes late-breaking information, software support, corrections, and instructions.	All	All	relnotes.pdf
<i>Getting a Quick Start with NerveCenter</i>	Provides a preview prior to installing and configuring NerveCenter for your network. Takes approximately one to two hours to complete.	All	New users	QuickStart.pdf
<i>Upgrading NerveCenter</i>	Explains how to upgrade your current NerveCenter version.	All	Installation team	upgrading.pdf
<i>Installing NerveCenter</i>	Helps you plan and carry out your NerveCenter installation. Use the <i>Release Notes</i> in conjunction with this book.	All	Installation team	install.pdf
<i>Managing NerveCenter</i>	Explains how to customize and tune NerveCenter after it has been installed.	NerveCenter Administrator	Administrator	managing_nervecenter.pdf
<i>Integrating NerveCenter with a Network Management Platform</i>	Explains how to integrate NerveCenter with network management platforms.	NerveCenter Administrator	Administrator	integratingNC.pdf
<i>Learning How to Create Behavior Models</i>	Provides step-by-step instructions and examples for creating behavior models.	NerveCenter Client	Users with administrative privileges	learningModel.pdf
<i>Designing and Managing Behavior Models</i>	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
<i>Monitoring Your Network</i>	Explains how NerveCenter works and how you can most effectively monitor your network.	NerveCenter Client and Web Client	Users	monitoringNet.pdf
<i>Behavior Models Cookbook</i>	Describes each behavior model shipped with Open NerveCenter.	NerveCenter Client	Users with administrative privileges	modsCookbook.pdf

Book Title	Description	Application	Audience	PDF for Print
Quick reference cards	<p>Quick reference cards provide convenient reference material for common NerveCenter tasks. The following cards are provided (PDF only):</p> <ul style="list-style-type: none"> ◆ Monitoring NerveCenter reference. ◆ Installing NerveCenter reference. ◆ Using Behavior Models reference. 	NerveCenter Client and Administrator	All	quickreference.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
<ul style="list-style-type: none"> ◆ A variable you substitute with a specific entry ◆ Emphasis ◆ Heading or Publication Title 	<i>Italic</i>	Enter <i>.installdb -f IDBfile</i>
Code samples, code to enter, or application output	Code	<code>iifInOctets > 0</code>
Messages in application dialog boxes	Message	Are you sure you want to delete?
An arrow (>) indicates a menu selection	>	Choose Start > Programs > Open NerveCenter

Element	Convention	Example
A link to a section in the same book	<i>Blue Italic</i>	For more information, see <i>Correlating Conditions</i> .
A link to a section in a different book	<i>Green Italic</i>	For more information, see <i>Correlating Conditions in Monitoring Your Network with NerveCenter</i> .

Note: If you are viewing this document in a PDF viewer, you may need to use the **Go to Previous View** button to return to the original PDF file.

**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.

**TIP**

A tip provides extra information that supplements the current topic. Often, tips offer shortcuts or alternative methods for accomplishing a task.



If toolbar buttons are available, they are displayed in the margin next to the step in which you can use them. Other shortcuts are noted as tips. Also, shortcut (accelerator) keys are displayed on application menus next to their respective options.

Documentation Feedback

OpenService, 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
 OpenService, Inc.
 110 Turnpike Road, Suite 308
 Westborough, MA 01581

documentation@openservice.com

Open Technical Support

Open 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 network management software, or Security Threat Manager, our security threat management suite.

Professional Services

Open 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

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

Contacting the Customer Support Center

For telephone support:

Phone: 1-888-886-1085, menu option 1 or 1-508-599-2000

For e-mail support:

E-mail: techsupport@openservice.com.

For electronic support

Open uses a Web-based customer call tracking system, TeamShare, where you can enter questions, log issues, track the status of logged incidents, and check the knowledge base.

When you purchased your product or renewed your maintenance contract, you received a user name and password to access the TeamShare system. If you have not received or have forgotten your log-in credentials, please e-mail us with a contact name and company specifics at techsupport@openservice.com.

For Online KnowledgeBase Access

For additional NerveCenter support, you can access the following information on the Open website, <http://www.openservice.com>:

- ♦ **Patches and Updates** - latest installation files, patches, and updates including documentation for NerveCenter.
- ♦ **Software Alerts** - latest software alerts relative to NerveCenter.
- ♦ **KnowledgeBase Search** - search the NerveCenter KnowledgeBase for answers to your questions whether relating to installation, usage, or operation.

How to Start Using NerveCenter Client

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

This chapter includes the following sections:

Section	Description
<i>Before You Begin</i> on page 10	Contains important information about this tutorial.
<i>What is the NerveCenter Client?</i> on page 12	Explains that the NerveCenter Client is the interface through which the user can view and create NerveCenter behavior models.
<i>How to Use the NerveCenter Client</i> on page 14	Steps you through the process of starting the client and connecting to the server.
<i>Review and Summary</i> on page 18	Gives you the opportunity to review what you learned.

Before You Begin

Learning How 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* 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* will step you through setting your network back to normal.



TIP

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 Appendix A, *Answers to Questions*.

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)
- ◆ The Windows or UNIX environment



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 machine
- ◆ A NerveCenter Server installed on your local machine or a machine accessible to 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)

See *Installing NerveCenter* for information on how to install the NerveCenter components you need to complete this tutorial.

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.

The Components of NerveCenter's Architecture

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. On Windows, the database can be either a Microsoft Access database or a Microsoft SQL Server database.

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

NerveCenter's 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 log on as the Windows Administrator, the UNIX supervisor, or a member of the NerveCenter administrators group.

TABLE 2-1. 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* on page 14
2. *Connecting to a NerveCenter Server* on page 16
3. *Quitting the NerveCenter Client* on page 17

Starting the NerveCenter Client

This first activity steps you through the process of opening the NerveCenter Client.

UNIX

TO START THE NERVECENTER CLIENT IN UNIX

1. Open a terminal window.
 2. Source one of the following environment variables:
installation_path/userfiles/ncenv.sh
installation_path/userfiles/ncenv.csh
installation_path/userfiles/ncenv.ksh
 3. Type **client &**.
NerveCenter displays the Open NerveCenter Client window.
-

WIN**TO START THE NERVECENTER CLIENT IN WINDOWS**

1. Select the **Start** menu.
2. Select **Programs > Open Service NerveCenter > Client**.

**NOTE**

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

NerveCenter displays the NerveCenter Client window.

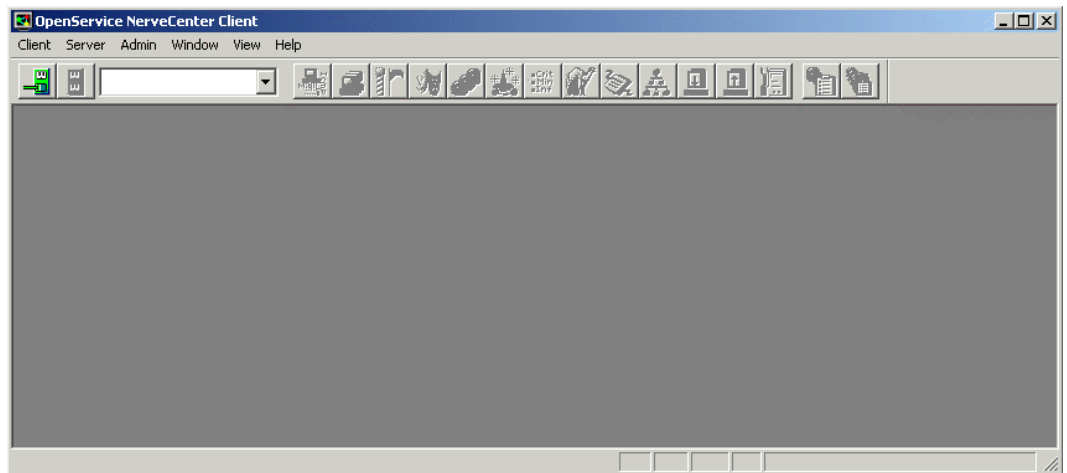


FIGURE 2-1. The NerveCenter Client Window

You have now just started the NerveCenter Client. The next activity will explain how to connect to a 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 either your machine or 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.

A screenshot of a dialog box titled "Connect to Server". It has a blue title bar with a question mark and a close button. The dialog contains three input fields: "Server Name" (a drop-down list box), "User ID" (a text box), and "Password" (a text box). At the bottom, there are three buttons: "Connect", "Cancel", and "Help".

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 be able to complete the activities and exercises in this book, you must be a member of a group with administrator rights.

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.

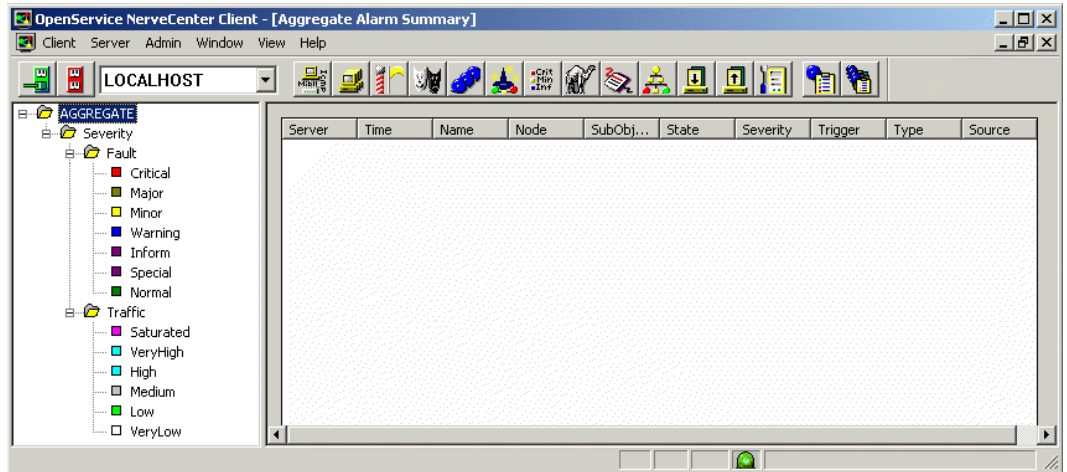


FIGURE 2-2. The NerveCenter Client Window after the User Has Connected to a Server

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.

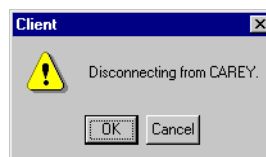
This last activity will step you through the process of disconnecting from the server and exiting the NerveCenter Client.

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.

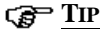


2. In the warning message, select **OK**.

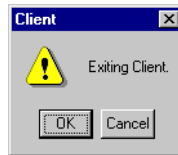
The NerveCenter Client disconnects from the active server.

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

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

**TIP**

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.



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 isolate some devices on your network so you can create a test group to work with. Chapter 3, *How to Define Property Groups and Properties* 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 1 Review Questions](#) on page 140.

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?

3. What time-saving feature is available for Windows users logging on to a NerveCenter Server with a 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 checklist 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

As you work through the exercises in this book, you will want to isolate certain devices to use. You can target NerveCenter's monitoring activities to a particular set of nodes by assigning them a unique 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

This chapter includes the following sections:

Section	Description
<i>What is a Property Group?</i> on page 22	Explains that a property group is a named container for strings, called properties.
<i>How to Create Property Groups and Properties</i> on page 23	Steps you through the process of creating and assigning a unique property group.
<i>Review and Summary</i> on page 30	Gives you the opportunity to review what you learned.

What is a Property Group?

Often you will create 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 NerveCenter's monitoring activity to a specific node or type of nodes. You can limit a behavior model's focus using property groups.

Each node in NerveCenter's node list is assigned a property group, which contains one or more properties. Each property is a string which can be:

- ◆ 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.

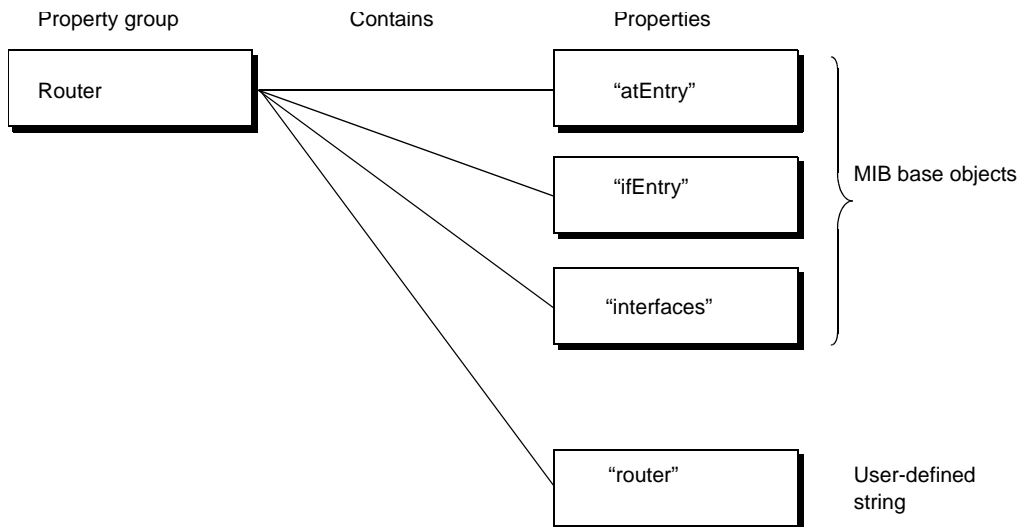


FIGURE 3-1. Property Groups and Properties

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 page 24
2. *Creating a New Property* on page 26
3. *Assigning a Property Group to a Set of Nodes* on page 27

Creating a New Property Group

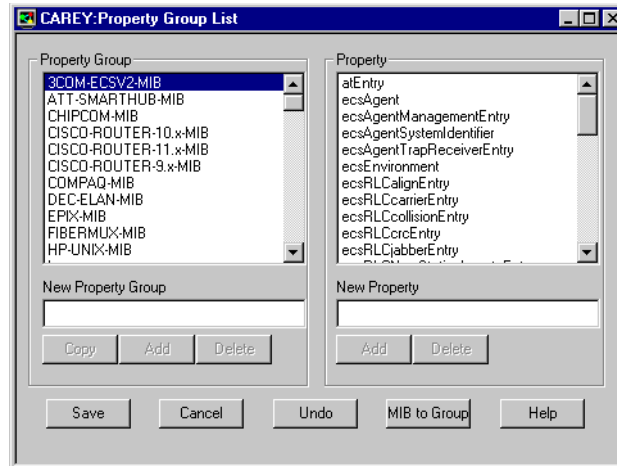
This first activity will step you through the process of creating a unique property group called **CriticalDevices**.

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**.

TIP

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.

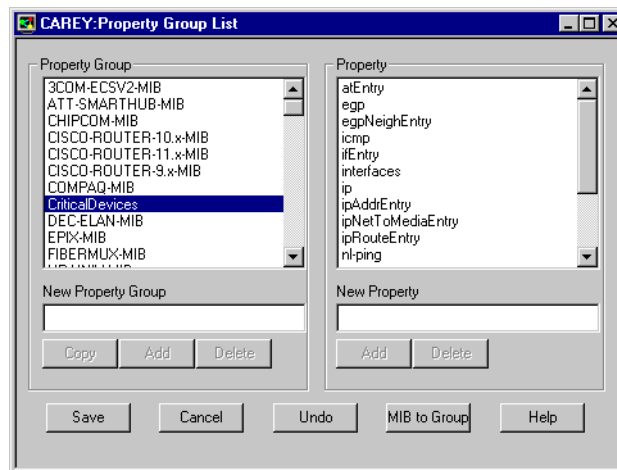
Creating a New Property

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.



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 isolating 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.

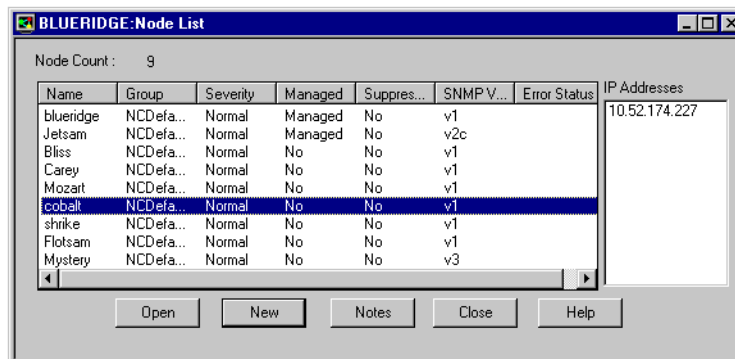
In our example network we would want to isolate the critical routers. For this tutorial, you will be isolating three of your department's managed workstations.

TO ASSIGN A PROPERTY GROUP TO A SET OF NODES

1. Identify at least three of your department's managed workstations that you know are running SNMP agents. Write their names on the following lines:

If you have questions about which devices to use, see *Before You Begin* on page 10.

2. From the **Admin** menu, choose **Node List**.
NerveCenter displays the Node List window.



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.

The screenshot shows the 'BLUERIDGE: Node Definition' window. The 'Node' tab is selected. The 'Name' field contains 'Carey'. The 'Property Group' dropdown is set to 'NCDefaultGroup'. The 'IP Address' field contains '10.52.174.24'. The 'Managed' checkbox is checked, and the 'Autodelete' checkbox is also checked. The 'Suppressed' and 'Platform Node' checkboxes are unchecked. The 'IP Address List' is currently empty. The 'Save', 'Cancel', 'Update', and 'Delete' buttons are visible.

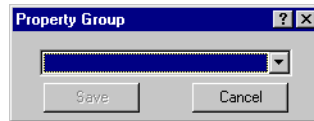
The Node Definition window enables you to change the various elements of a managed device, such as its name, IP address, community string, deletion strategy, and property group.

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 down the **Ctrl** key and select the remainder of the 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.



In the Property Group window, select the **CriticalDevices** property group.

9. Select **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. Chapter 4, *How to Use Polls* 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 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 2 Review Questions](#) on page 140.

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 checklist 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

3

How to Define Property Groups and Properties

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

This chapter includes the following sections:

Section	Description
<i>What is a Poll?</i> on page 34	Explains that a poll monitors a particular set of conditions on your network.
<i>How to Create a Poll</i> on page 34	Steps you through the process of creating a new poll.
<i>Review and Summary</i> on page 44	Gives you the opportunity to review what you learned.

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.

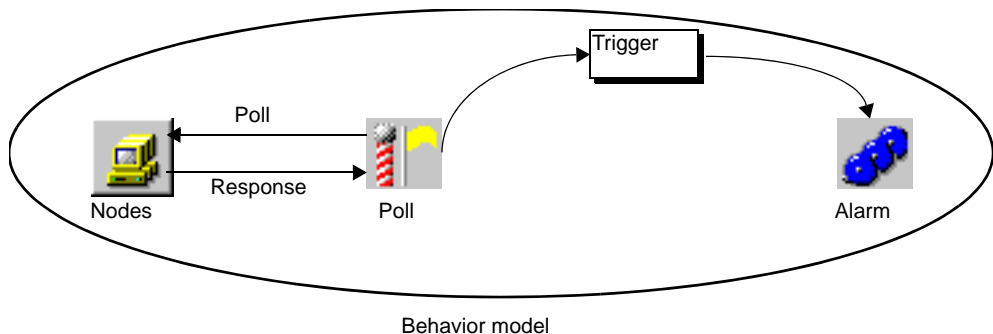


FIGURE 4-1. The Role of a Poll within a Behavior Model

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:

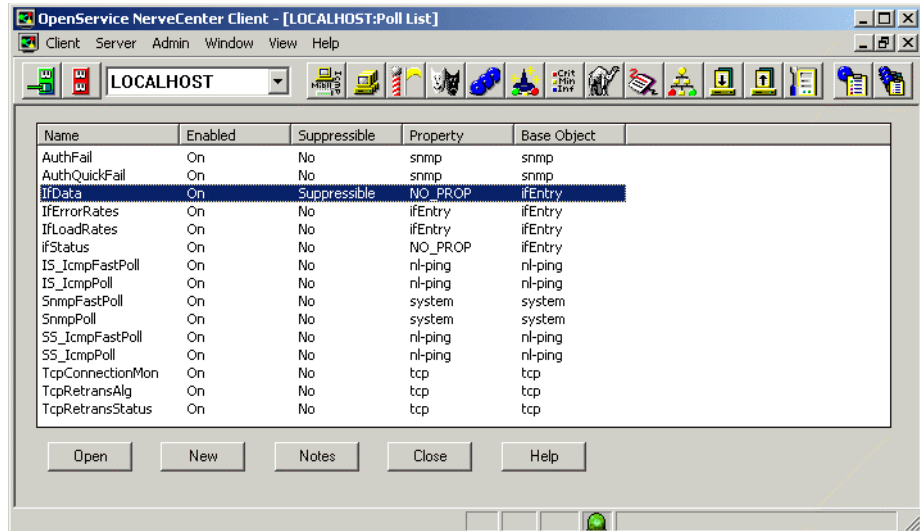
1. *Creating a New Poll* on page 35
2. *Writing a Poll Condition* on page 37
3. *Enabling a Poll* on page 40
4. *Modifying a Poll Condition* on page 41

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

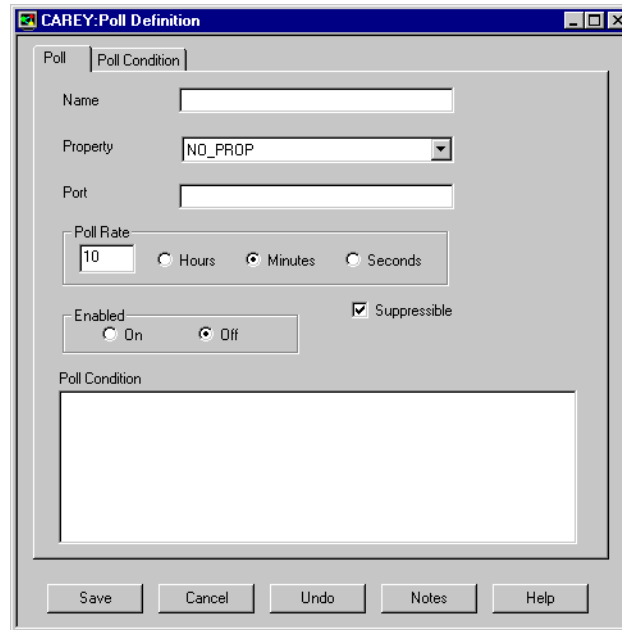
1. Open the NerveCenter Client and connect to the appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Poll List**.
NerveCenter displays the Poll List window.



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.



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.

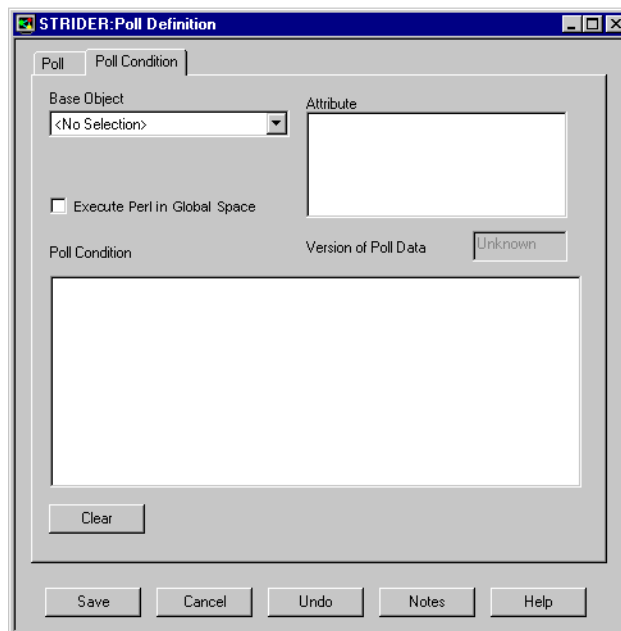
Writing a Poll Condition

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 ICheckTraffic poll.

TO WRITE A POLL CONDITION

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



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.

**NOTE**

Although you may include as many attributes as you like in a poll condition, each poll may use only one MIB base object.

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**.

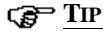
The word **FireTrigger** and a left parenthesis appears in the expression.

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 look like this:

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

**TIP**

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 `lCheckTraffic` can begin polling, you must complete a few more steps as explained in [Enabling a Poll](#).

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

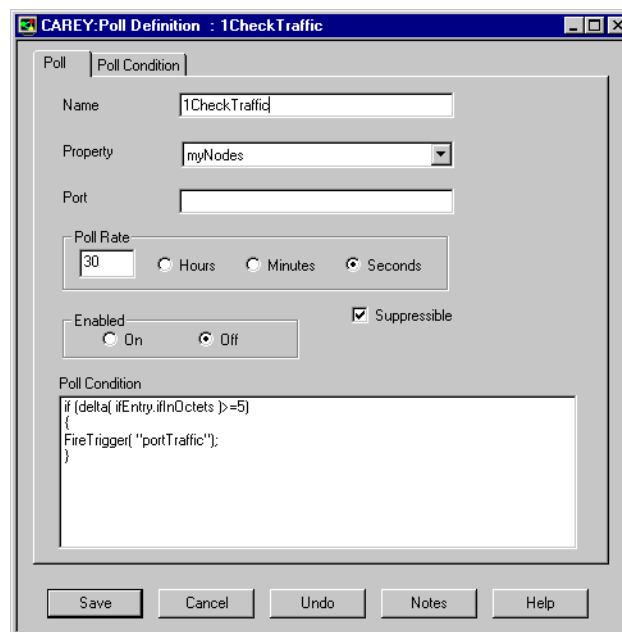
You have just completed creating the 1CheckTraffic poll, which will poll all the nodes in the CriticalDevices property group for the number of ifEntry.ifInOctets. When the poll detects high-traffic conditions (what we've defined as a difference of five), it will fire the trigger portTraffic.

For this poll to begin actively polling, it must be enabled. This activity steps you through that process.

TO ENABLE A POLL

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

NerveCenter displays the Poll page.



Notice that the poll condition you just entered now appears in the poll condition window on the Poll page.

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 47.

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

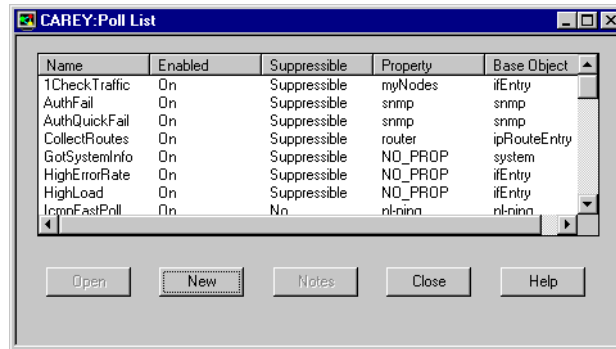
Modifying a Poll Condition

Before you create an alarm associated with the `1CheckTraffic` poll, you should modify the poll condition to improve the poll. This will give 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.



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**.

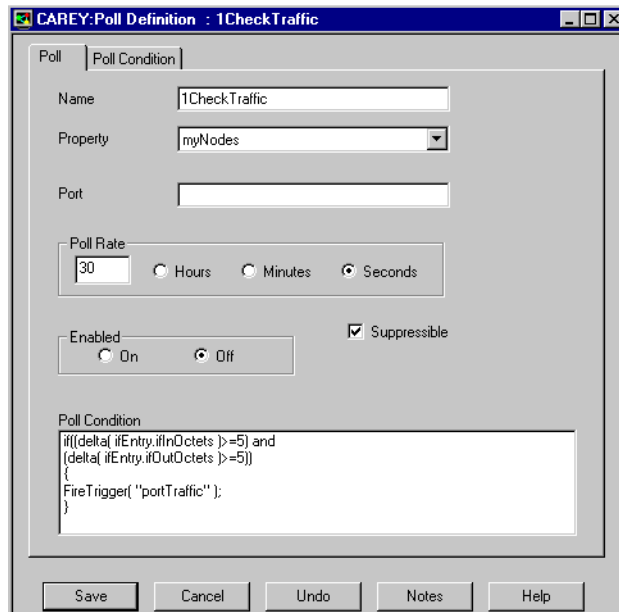
5. In the Poll Definition window, select the **Poll Condition** tab.

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**.
9. Select **Save**.

Now you have created, modified, and enabled the 1CheckTraffic poll. In Chapter 5, *How to Use Alarms* you will create an alarm that will transition states when your new poll generates the portTraffic trigger.

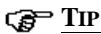
Review and Summary

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.

**TIP**

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 3 Review Questions](#) on page 141.

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

2. What is a trigger?

3. How many different MIB base objects can you use to form a complex poll condition?

4. 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 checklist 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

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

This chapter includes the following sections:

Section	Description
What is an Alarm? on page 48	Explains that an alarm is a NerveCenter object that correlates one or more detected network events and performs certain actions in response to those events.
How to Create an Alarm on page 49	Steps you through the process of creating, enabling, and modifying a new alarm.
Review and Summary on page 62	Gives you the opportunity to review what you learned.

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.

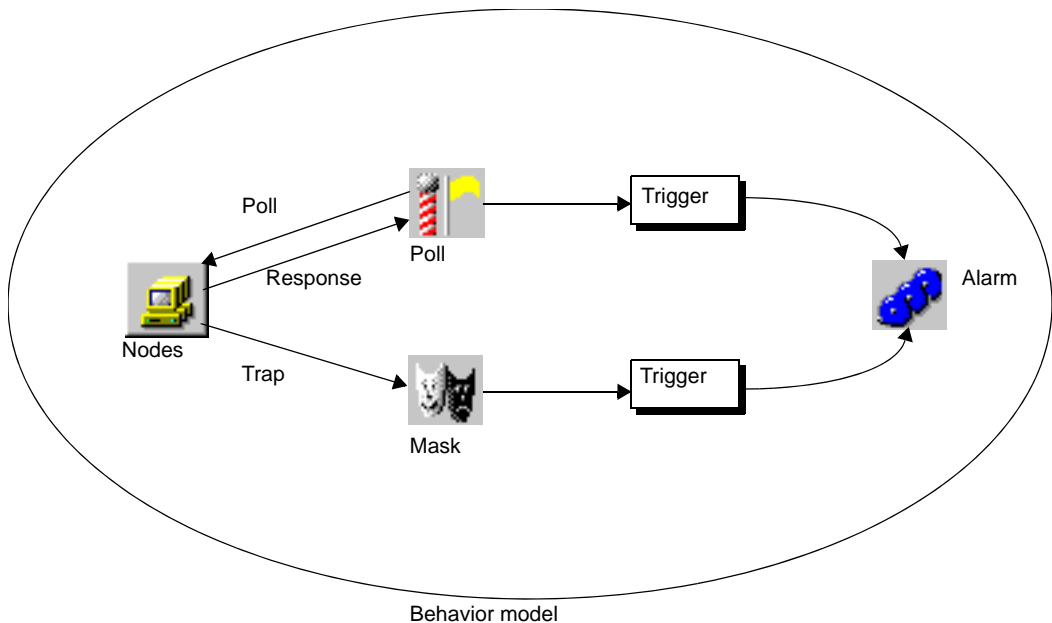


FIGURE 5-1. The Role of an Alarm within a Behavior Model

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 page 49
2. *Designing a State Diagram* on page 52
3. *Enabling an Alarm* on page 55
4. *Modifying an Alarm's State Diagram* on page 58
5. *Adding Another State to an Alarm* on page 60

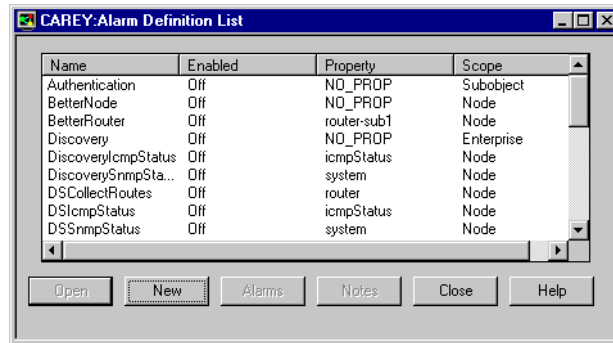
Creating a New Alarm

This first activity will step you through the first stage of creating a new alarm that monitors 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.



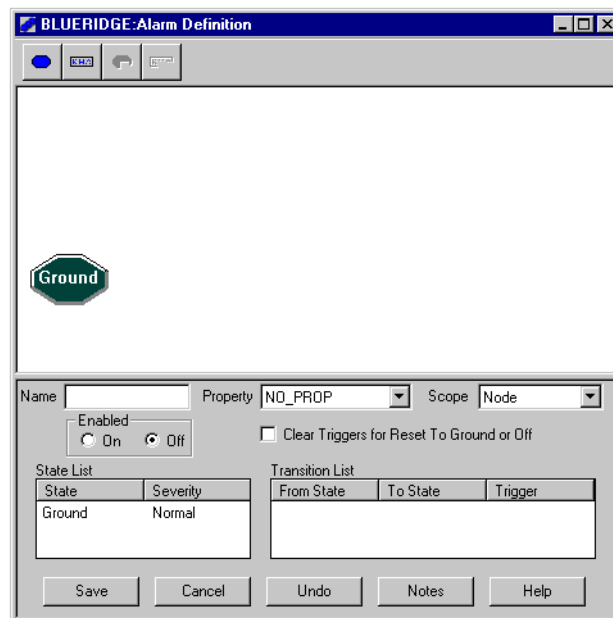


This window lists all the alarms in the NerveCenter database.

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. The definition is implemented through the state diagram. The next activity will step you through the process of designing the state diagram. First, you must name the alarm and limit the nodes it will monitor.



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 103
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.

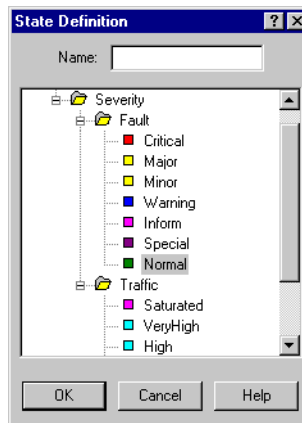
Designing a State Diagram

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

TO DESIGN A STATE DIAGRAM



1. From the toolbar at the top of the Alarm Definition window, 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**.

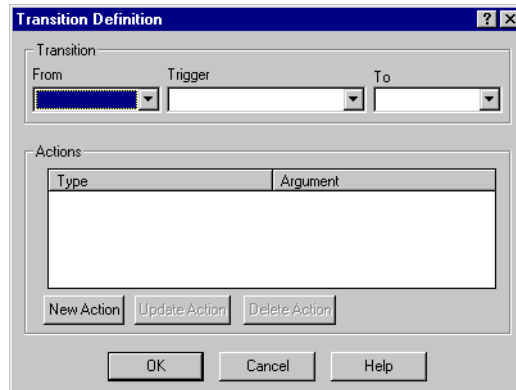
The State Definition window closes. An icon appears in the top left corner with the name of your new state, Busy, and a bright green color, indicating a low severity.

5. Position the Busy state icon in the middle of the state diagram.



6. From the toolbar at the top of the Alarm Definition window, 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 Chapter 4, *How to Use Polls* 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.



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.

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



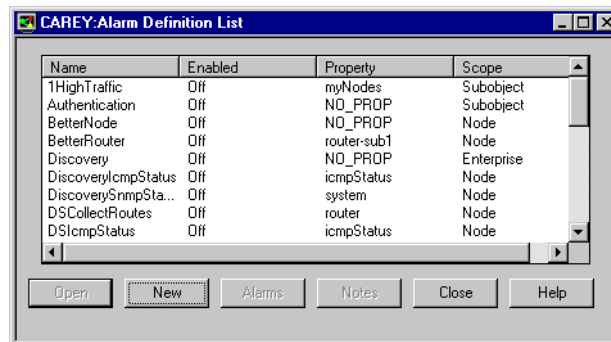
1. From the **Admin** menu, choose **Poll List**.
NerveCenter displays the Poll List window.

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.



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**.

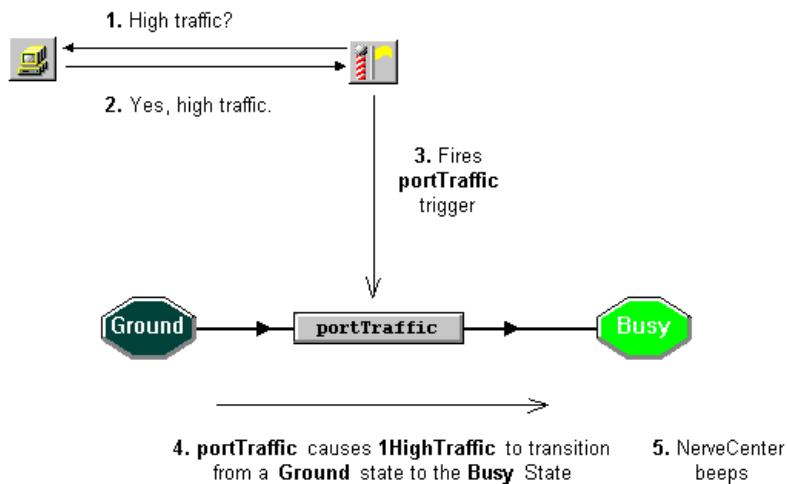
This enables the 1HighTraffic alarm.

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.

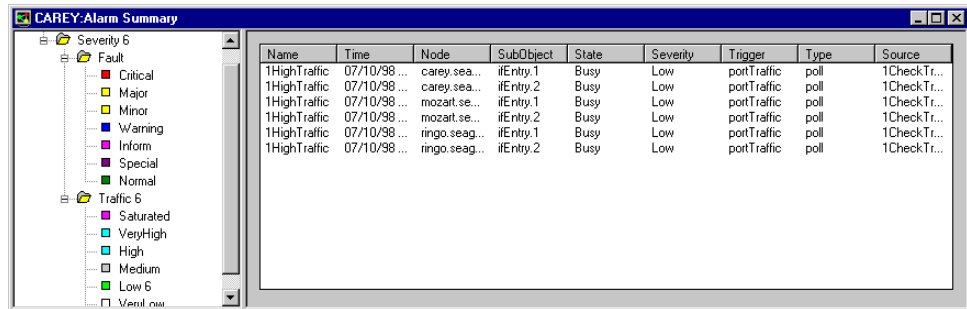


NOTE

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.



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 an Alarm's State Diagram

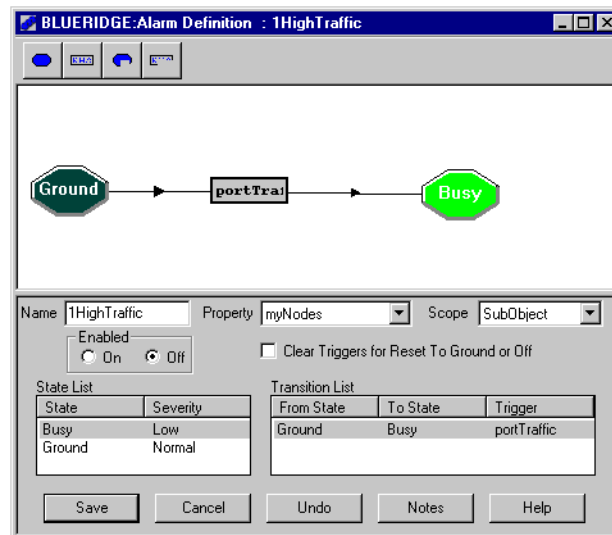
There may be times you will want to adjust the icons of the state diagram to make them easier to read.

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



- From the **Admin** menu, select **Alarm Definition List**.
NerveCenter displays the Alarm Definition List window.
- From the Alarm Definition list, select **1HighTraffic**. Then select **Open**.
The Alarm Definition window for 1HighTraffic appears.

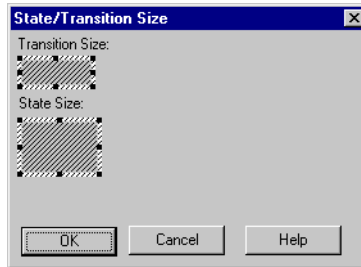


- If the alarm is on, select **Off**, then select **Save**.

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.

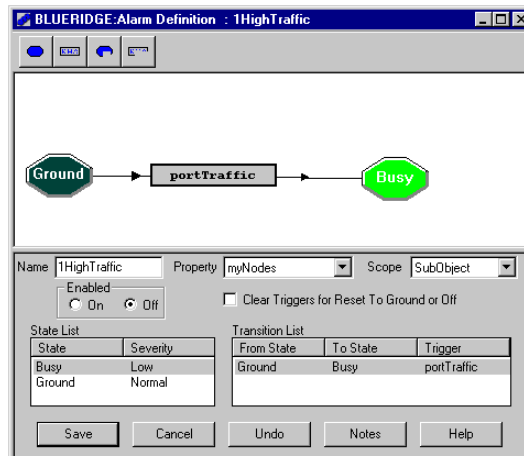
- Right-click anywhere within the state diagram. Select **Size**.
The State/Transition Size window appears.



The State Size and Transition Size rectangles indicate the current size of the State and Transition icons.

- Drag the handles of the State Size rectangle to change the height and width of the rectangle. Repeat for the Transition Size rectangle.
- Select **OK**.

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



- 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. From the toolbar at the top of the Alarm Definition window, select **Add State**.



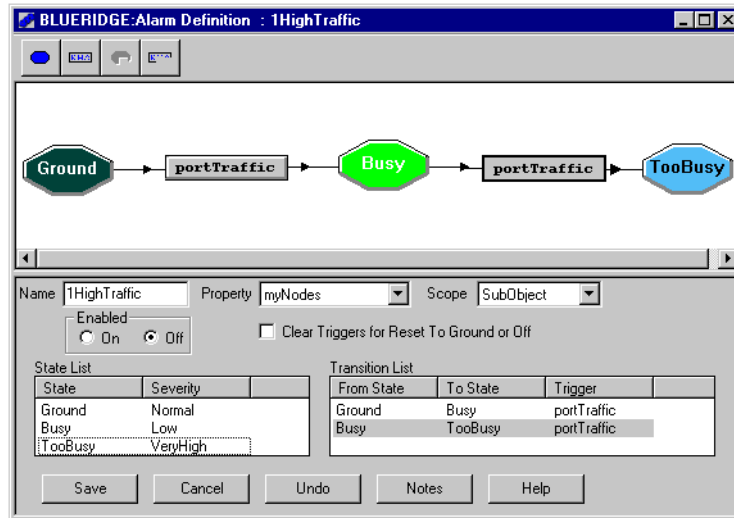
NerveCenter displays the State Definition window.

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. From the toolbar at the top of the Alarm Definition window, 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.



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

- In the state diagram, double-click on 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.

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

The **Delete Action** button is enabled.

- Select **Delete Action**.

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



- 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 Chapter 6, *How to Use Trap Masks*.

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 4 Review Questions](#) on page 142.

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 checklist 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

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

This chapter includes the following sections:

Section	Description
What is a Trap Mask? on page 66	Explains that a trap mask fires a trigger when it detects a relevant SNMP trap from a managed node or other source.
How to Create a Trap Mask on page 67	Steps you through the process of creating a trap mask and an associated alarm.
Review and Summary on page 79	Gives you the opportunity to review what you learned.

What is a Trap Mask?

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

- ◆ A poll
- ◆ A trap mask
- ◆ Another alarm
- ◆ A Perl subroutine

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.

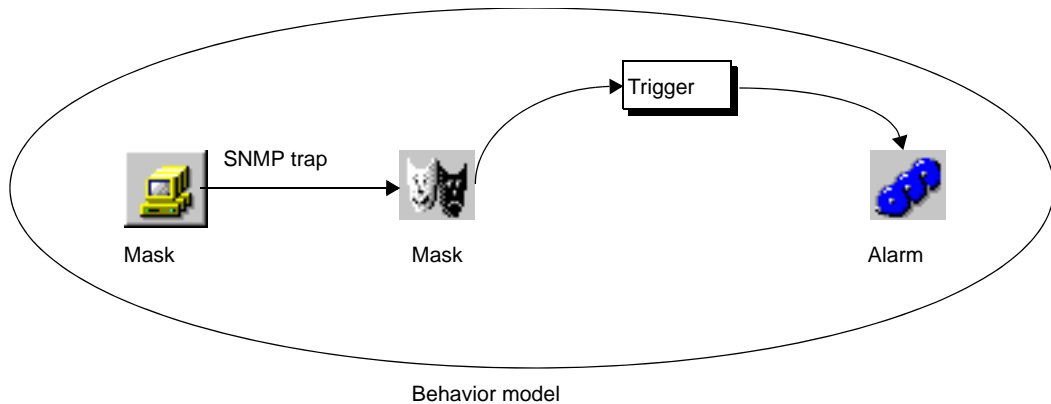


FIGURE 6-1. The Role of a Trap Mask within a Behavior Model

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:

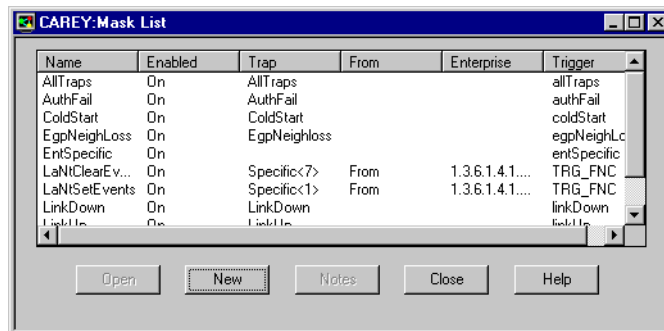
1. *Creating a New Trap Mask* on page 67
2. *Creating an Alarm to be Triggered by a Mask* on page 70
3. *Using Traps to Generate a Trap* on page 73
4. *Defining a Trigger Function* on page 76

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

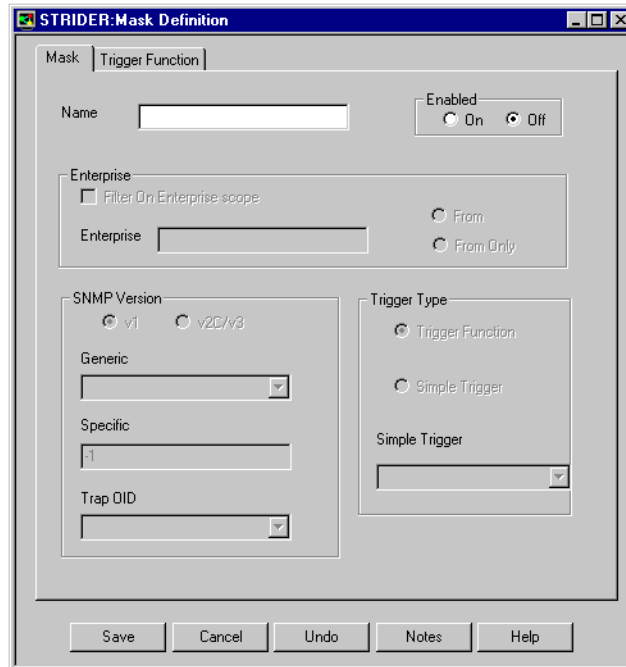
1. Open a NerveCenter Client and connect to an appropriate NerveCenter Server.
2. From the **Admin** menu, choose **Mask List**.
NerveCenter displays the Mask List window.



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

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

The Mask Definition window appears.



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

- In the **Name** field, type **1CheckAuth**.
- 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.

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

These fields will be explained later.

- 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**.
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 usually include the following information:

- ◆ A generic number (0-6)
- ◆ An enterprise-specific number
- ◆ An enterprise name
- ◆ Extra data in structured trailing bytes called variable bindings

Creating an Alarm to be Triggered by a Mask

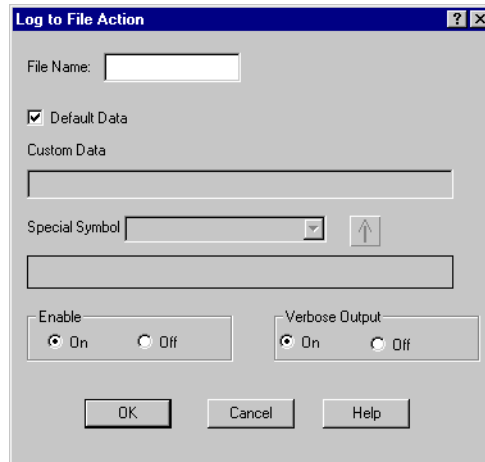
In the last activity, you created a mask called 1CheckAuth to detect for SNMP AuthFail traps. You now need an alarm that will be triggered when the trap is detected.

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.



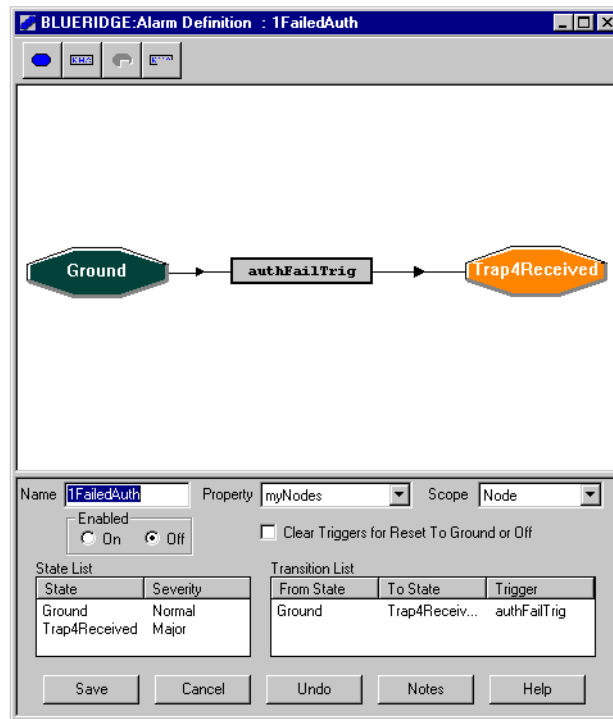
10. In the **File Name** field, type **myLog**.
11. Leave **Default Data** checked.
12. Select **On** in the **Enable** and **Verbose Output** fields.
13. Select **OK**.

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 the directory *installation_directory/Log* (Windows) or *installation_directory/userfiles/logs* (UNIX). If you enter a full pathname, the log file is written to the directory you specified.

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

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



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 utility *trapgen*.

TO USE TRAPGEN TO GENERATE A TRAP

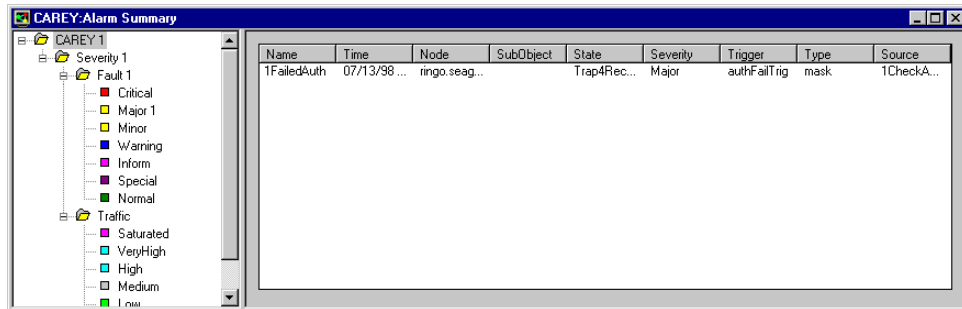
1. Check to make sure the 1CheckAuth mask and the 1FailedAuth alarm are On.
2. From the command line (MS-DOS command prompt or UNIX shell) type:

```
trapgen server_name "" a_node_name 4 0 ""
```

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

- ♦ *server_name* should be the name of the current active server for NerveCenter.
 - ♦ "" for enterprise sends the trap with a default value.
 - ♦ *a_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.



5. Locate in the directory *installation_directory/Log* (Windows) or *installation_directory/userfiles/logs* (UNIX) the file **myLog**. Open this file with an ASCII text editor such as Notepad (Windows) or vi (UNIX).

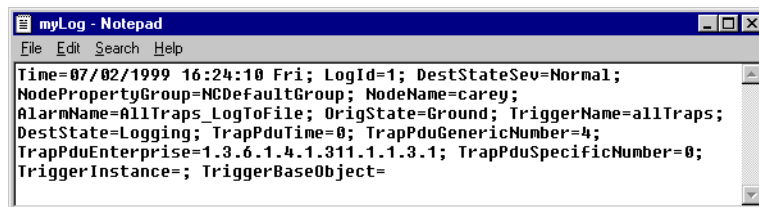


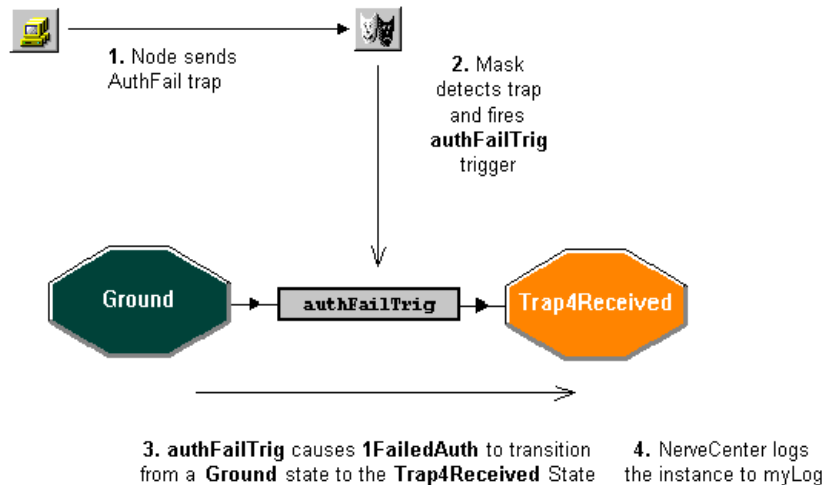
FIGURE 6-2. myLog as it Appears in Notepad

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?

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

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.



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 would detect SNMP generic 4 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 a trap with the generic trap number 4 and the enterprise-specific trap 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 generic 4 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**.

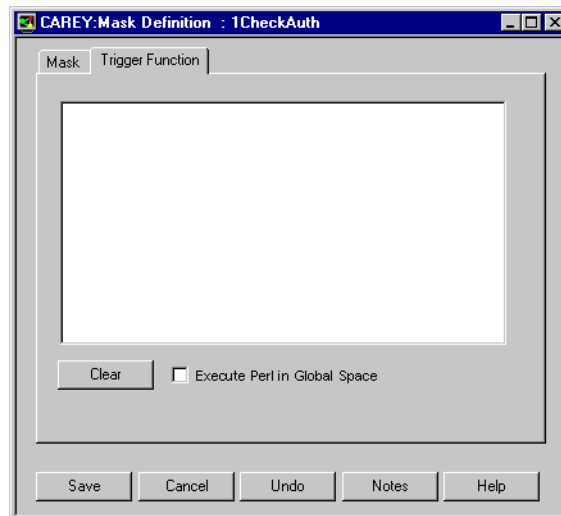
The Mask Definition window for 1CheckAuth appears. If 1CheckAuth is still enabled, all the fields will be grayed out.

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.



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.

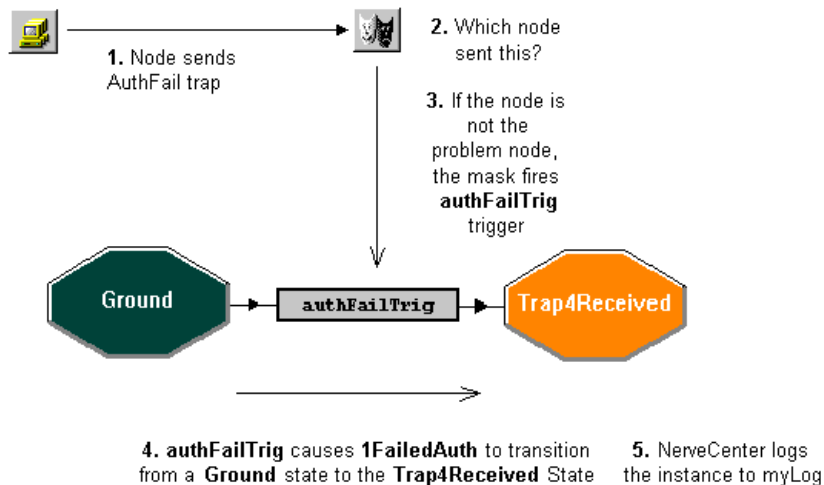
7. Select the **Mask** tab.
8. In the Enable frame, select **On**.
9. Select **Save**.
10. Open the Alarm Summary window.
11. Follow the steps in the last activity, *Using Trapgen to Generate a Trap* on page 73 to test the modified mask:

- 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 trap 4 and determines if the 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.



You now know how to use alarms, polls, and masks, the main elements in NerveCenter behavior models. In Chapter 7, *How to Use Behavior Models* 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 `!LinkDown`, that fires the simple trigger `downLink` when it detects an SNMP generic 2 trap
 - b. Creating a mask named `!LinkUp`, that fires the simple trigger `upLink` when it detects an SNMP generic 3 trap
 - c. Creating an alarm named `!LinkDownUp`, 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. Modify the trigger function of the two masks so that they only fire a trigger when one specific node in the `CriticalDevices` property group sends a trap.

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 143.

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 checklist 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

6

How to Use Trap Masks

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

This chapter includes the following sections:

Section	Description
What is a Behavior Model? on page 84	Explains that a behavior model is a set of specifications that tell NerveCenter how to detect and respond to certain conditions.
How to Create Useful Behavior Models on page 85	Steps you through the process of creating different behavior models for notifying you only of persistent problems.
Review and Summary on page 99	Gives you the opportunity to review what you learned.

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* on page 85
2. *Notifying of a Persistent Condition* on page 88
3. *Using a Timer to Detect Persistence* on page 91
4. *Detecting Unresponsive Nodes* on page 94
5. *Testing for an Unresponsive Node* on page 96

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.

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.



NOTE

The notBusy trigger was created in [Review and Summary](#) on page 44. 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, such as OpenView. 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 IPersistentTraffic'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.

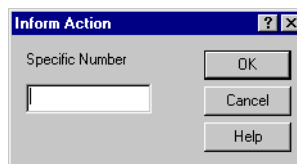


2. Add a transition, which moves from the **Busy** state to the **TooBusy** state when the tooBusy trigger is fired.

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 85.

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 like HP OpenView 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 **11111** 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.

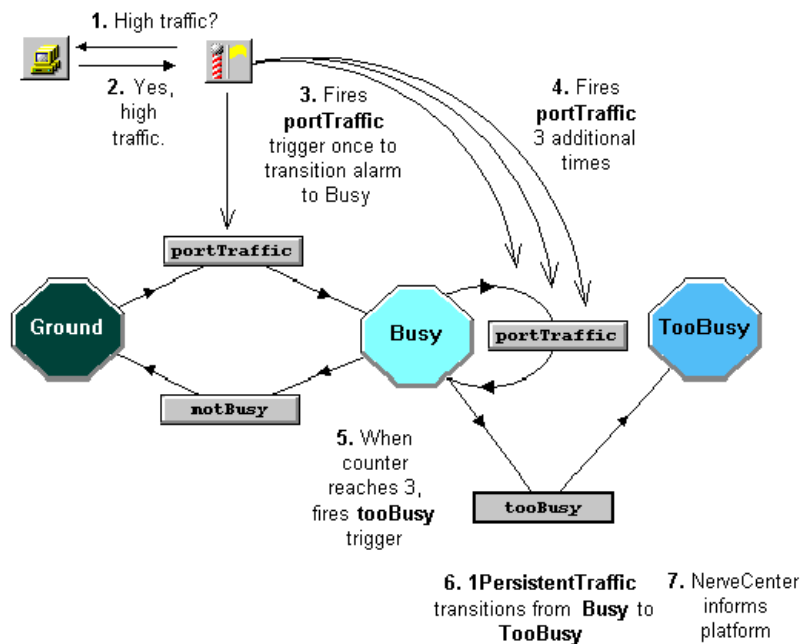
The new event should be configured for the specific trap number 11111 or the number you entered in step 4. See your network management platform’s documentation for further instructions.

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

TABLE 7-1. The Process that Occurs when 1PersistentTraffic is Enabled**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.



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 a generic 4 trap to be sent out. You want your behavior model to notify you only of repeated attempts to breach security.

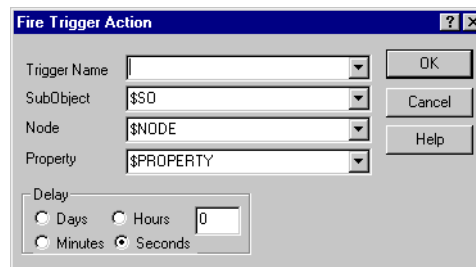
Earlier in this chapter you used a counter to detect a persistent condition. In this activity you will use a timer.

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.



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.

- 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.

- 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.

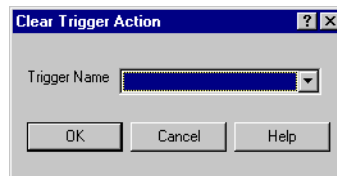
- Add a transition that moves from the Trap4Received state to the Ground state when the trigger authTimeout is fired.

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.

- 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.
- In the Transition Definition window, select **New Action**. From the alarm action list, select **Clear Trigger**.

The Clear Trigger Action window appears.



- 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.

- In the Clear Trigger Action window, select **OK**. In the Transition Definition window, select **OK**. In the Alarm Definition window, select **Save**.
- Turn the mask **1CheckAuth** and the alarm **1PersistentAuthFail** on.



13. Use **trappgen** to simulate two generic 4 traps from the same node within two minutes of each other.

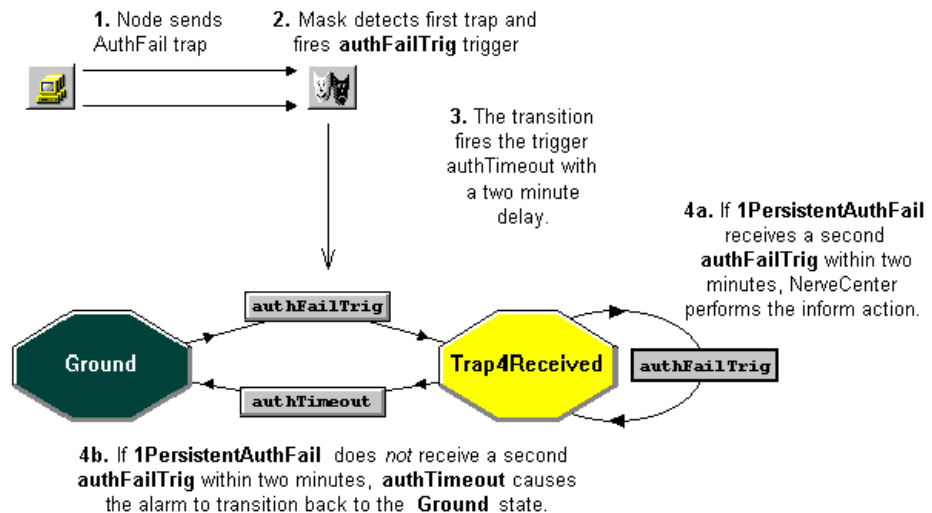
The Alarm Summary window will display the progress of the 1PersistentAuthFail alarm.

TABLE 7-2. The Process that Occurs when 1PersistentAuthFail is Enabled

What is NerveCenter doing?

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

1. The mask 1CheckAuth detects the trap 4 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.



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.

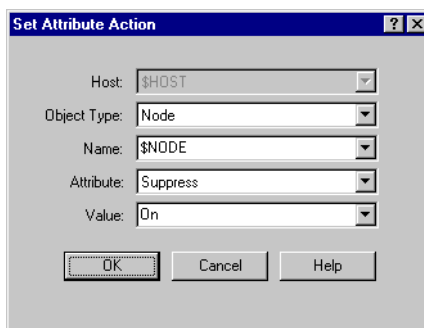


3. In the state diagram, add a transition that will move from the Ground state to the NoResponse state.
4. In the **trigger** field, select **ERROR**.

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.



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 30. 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.

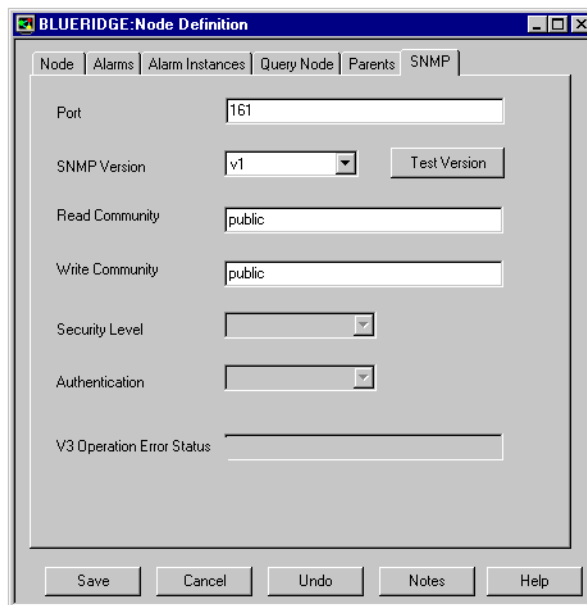
TO TEST FOR AN UNRESPONSIVE NODE



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.

The screenshot shows the 'BLUERIDGE: Node Definition' window with the 'SNMP' tab selected. The 'Name' field contains 'Carey'. The 'Property Group' dropdown is set to 'NCDDefaultGroup'. The 'IP Address' field contains '10.52.174.24'. There are three buttons: 'Add', 'Update', and 'Delete'. Below these is an empty 'IP Address List' box. On the right side, there are four checkboxes: 'Managed' (checked), 'Autodelete' (checked), 'Suppressed' (unchecked), and 'Platform Node' (unchecked). At the bottom of the window are buttons for 'Save', 'Cancel', 'Undo', 'Notes', and 'Help'.

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



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.

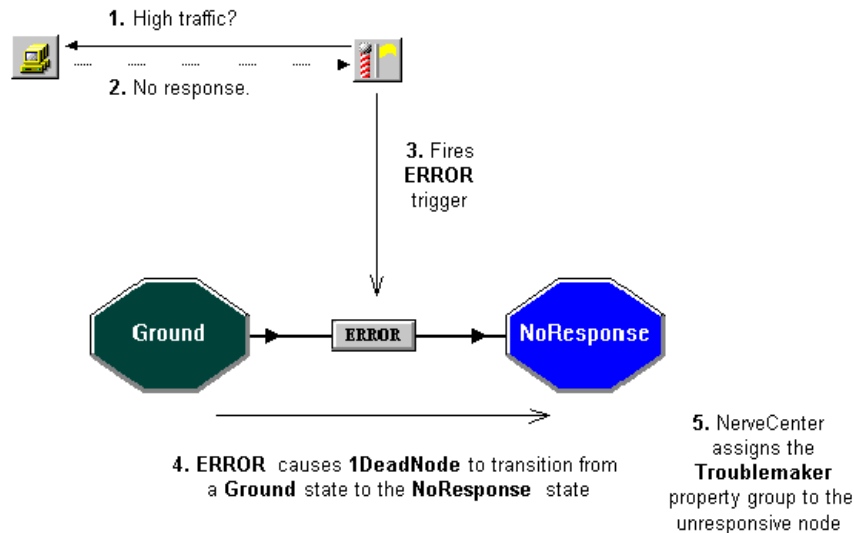
The **ERROR** trigger will not be fired unless an unresponsive node is polled. This is why you must first turn on **1CheckTraffic**. Because of smart polling, the **1CheckTraffic** poll will not work unless an associated alarm, such as **1HighTraffic**, is enabled.

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

TABLE 7-3. The Process that Occurs when 1DeadNode is Enabled**What is NerveCenter doing?**

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

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



5. Follow the instructions from step 1 through step 4, to return the proper value to the node's **Read Community** field.
6. 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.

Chapter 8, *How to Use Alarm Scope in Behavior Models* 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 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 6 Review Questions](#) on page 143.

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 checklist 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

7

How to Use Behavior Models

How to Use Alarm Scope in Behavior Models

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

This chapter includes the following sections:

Section	Description
<i>What is Alarm Scope?</i> on page 104	Explains that alarm scope is a setting that determines how specifically an alarm will monitor an instance.
<i>How to Use Alarm Scope in a Multi-alarm Behavior Model</i> on page 107	Steps you through the process of using alarm scope to create a multi-alarm behavior model.
<i>Review and Summary</i> on page 115	Gives you the opportunity to review what you learned.

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 8-1. The scopes are listed on the left and the properties required to match the scope are listed across the top.

TABLE 8-1. 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?

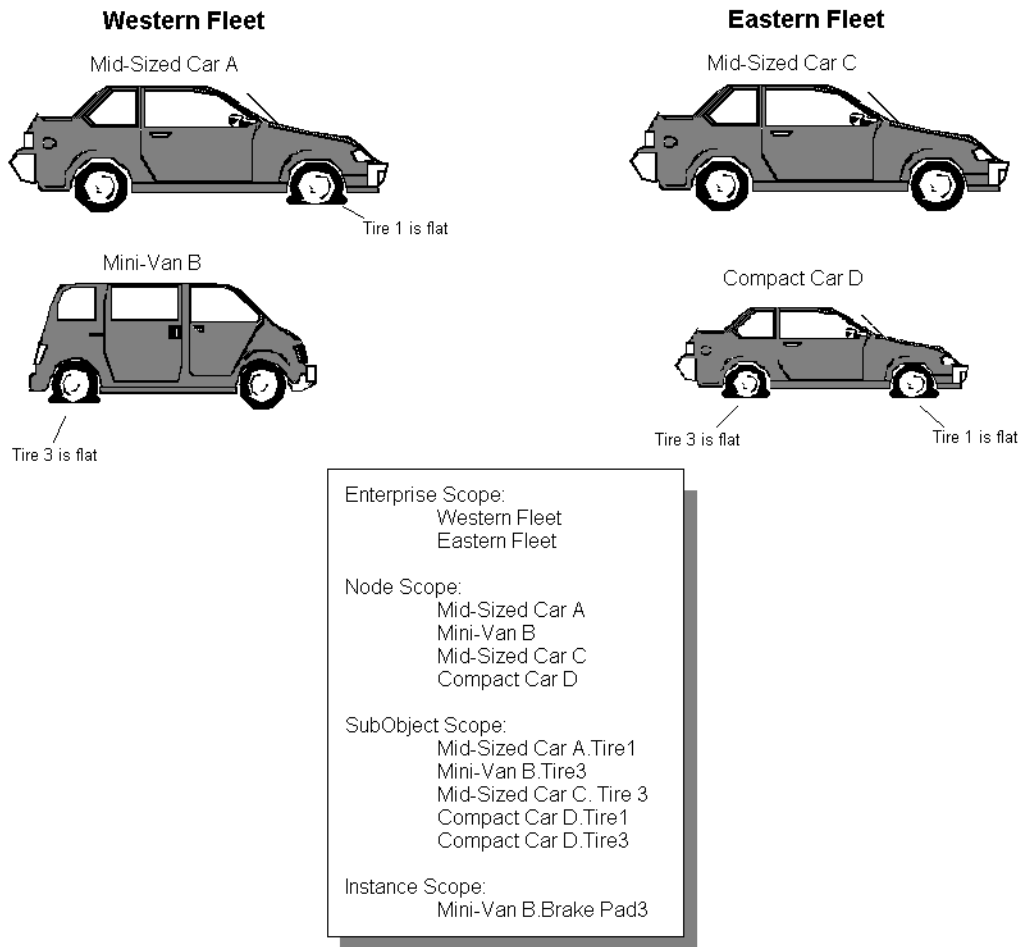


FIGURE 8-1. A Car Rental Company Uses Alarm Scope to Keep Track of Flats

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](#) on page 108
2. [Creating the Second Alarm of a Multi-alarm Behavior Model](#) on page 110
3. [Adding Reset Capabilities to a Multi-alarm Behavior Model](#) on page 112

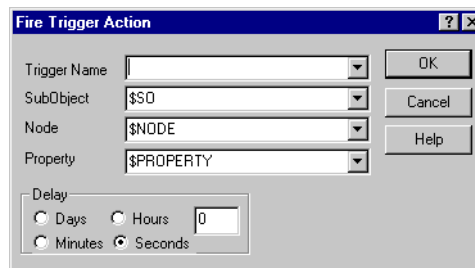
Creating the First Alarm of a Multi-alarm Behavior Model

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.



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.

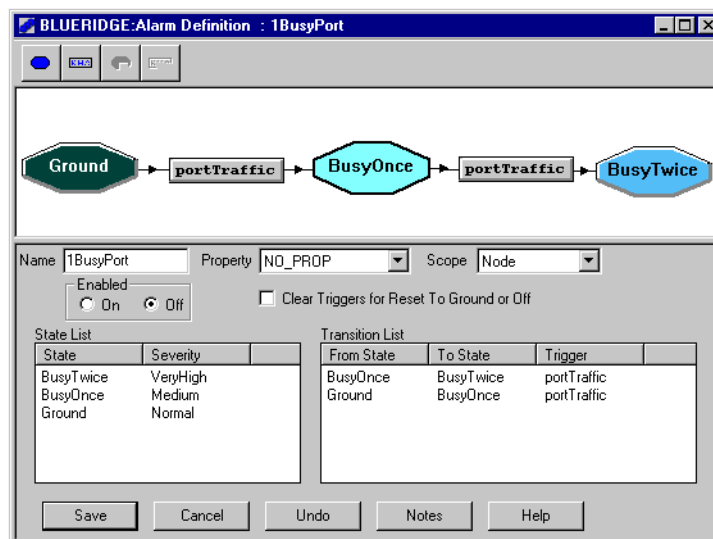


FIGURE 8-2. The Completed State Diagram for 1BusyPort

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.

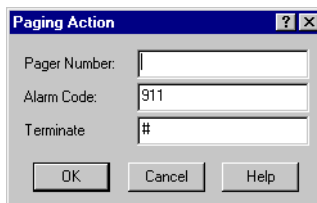
- a. `1PortBusy` with a traffic severity of Low
- b. `2PortsBusy` with a traffic severity of Medium
- c. `3PortsBusy` with a traffic severity of High
- d. `4PortsBusy` with a traffic severity of Very High

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.

- In the Transition Definition window, select **New Action**. From the alarm action list, select **Paging**.

The Paging Action window appears.



NOTE

Before you can use the Paging action, someone must have configured NerveCenter to handle paging actions correctly. If this has not been done, or you do not use a pager, substitute an alarm action of your choice.

- In the **Pager Number** field, type your pager's phone number.
The pager number is the sequence of digits and special Hayes AT commands needed by the Paging action to reach the pager. For a list of valid commands, see your modem manual.
- In the **Alarm Code** field, either type a number that identifies the network situation being reported or leave it set to the default.
The alarm code is a sequence of digits that is displayed on the pager. The maximum number of digits that a pager can display varies from pager to pager. If you don't supply an alarm code, a default value of 911 is used.
- In the **Terminate** field, type the character that terminates the paging connection on your pager.
This character is a key the paging system uses to terminate the connection and send the page. It differs from system to system, but is usually # (pound sign) or * (asterisk). Consult your paging system manual to determine the correct key for your system. If you don't specify a key, the Paging action uses the default value #.
- In the Paging Action window, select **OK**.
The Paging action is added to the Transition Definition window's action list.
- In the Transition Definition window, select **OK**.
- In the Alarm Definition window, select **Save**.

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

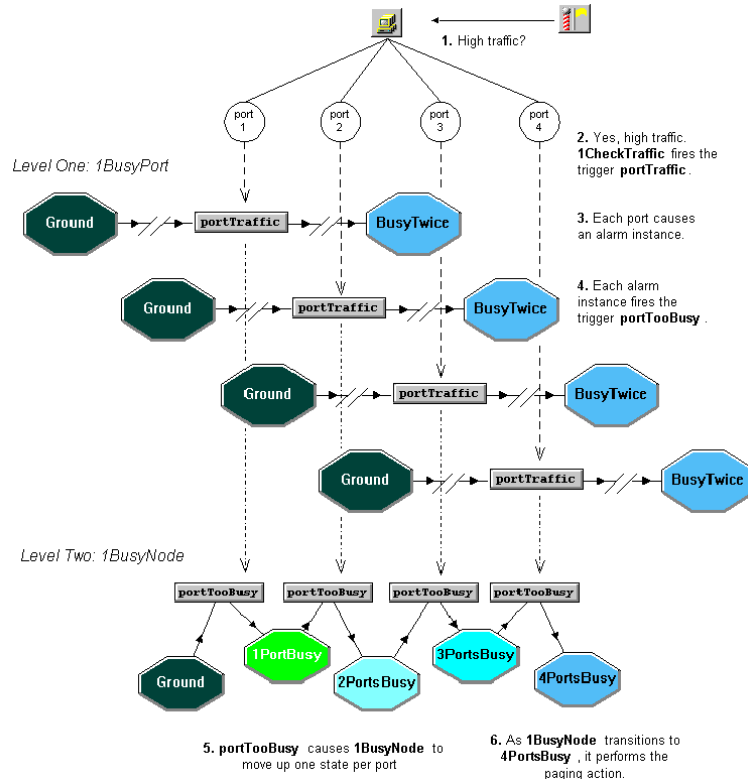
Unless one of the nodes in the CriticalDevices property group has four interfaces, the alarm IBusyNode will not reach the final state.

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

TABLE 8-2. 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 paging action.



Chapter 9, *How to Define Conditional Actions with Action Router* 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 7 Review Questions](#) on page 144.

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 checklist 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

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

This chapter includes the following sections:

Section	Description
What is Action Router? on page 118	Explains that Action Router is a feature that allows you to dictate which actions should be performed under particular circumstances.
How to use Action Router on page 119	Steps you through the process of using Action Router to set particular conditions for when you should be notified.
Review and Summary on page 127	Gives you the opportunity to review what you learned.

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*.

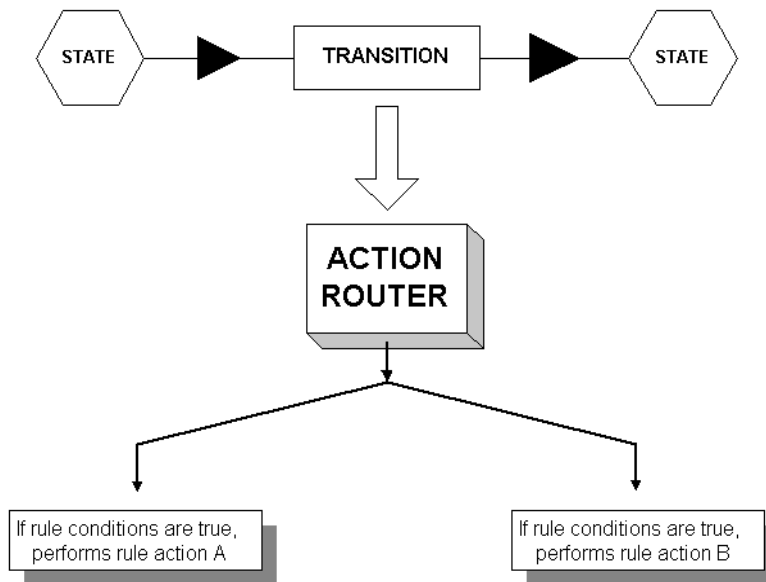


FIGURE 9-1. The Role of Action Router within a Behavior Model

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 getting paged for non-critical nodes. You want to reserve paging 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 page 119
2. *Defining a Rule Action in Action Router* on page 123
3. *Using Action Router in an Alarm* on page 125

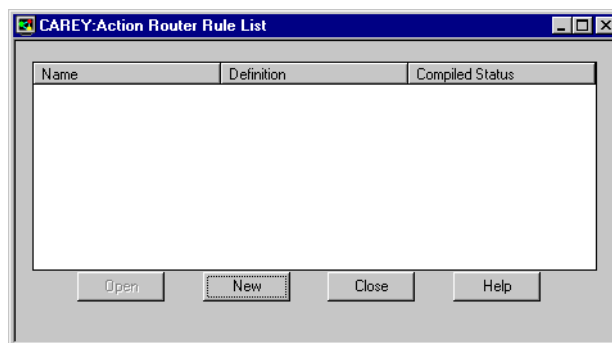
Defining a Rule Condition in Action Router

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

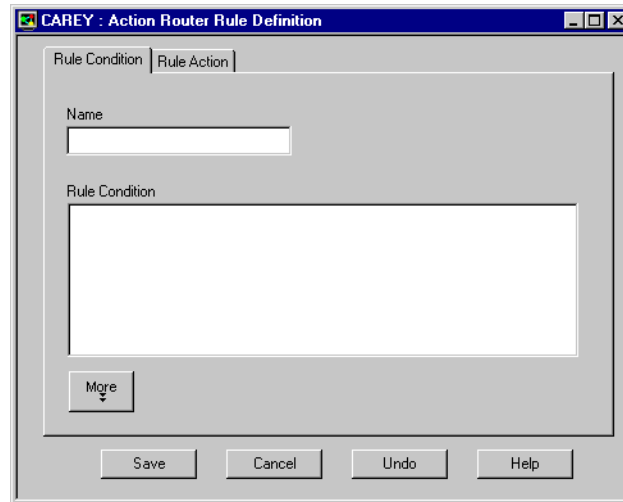
TO DEFINE A RULE CONDITION IN ACTION ROUTER



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 **PagingTest**.

This rule will determine by the property group of the node if NerveCenter should page an administrator 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.

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.

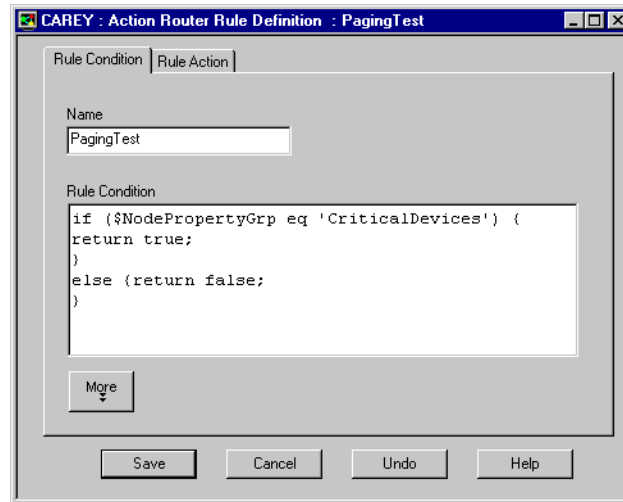


FIGURE 9-2. The Completed Rule Condition of PagingTest

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.

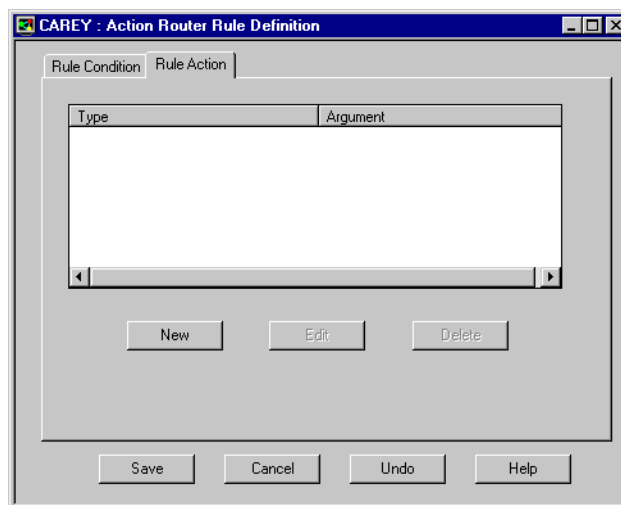
Defining a Rule Action in Action Router

In the last activity you defined the conditions under which the Action Router would perform the actions of the PagingTest 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.

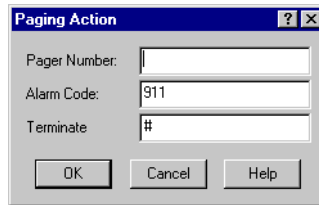
TO DEFINE A RULE ACTION IN ACTION ROUTER

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 **Paging**.

The Paging Action window appears.

**NOTE**

Before you can use the Paging action, someone must have configured NerveCenter to handle paging actions correctly. If this has not been done, or you do not use a pager, substitute an alarm action of your choice.

3. In the **Pager Number** field, type your pager's phone number.

The pager number is the sequence of digits and special Hayes AT commands the Paging action needs to reach the pager. For a list of valid commands, see your modem manual.

4. In the **Alarm Code** field, either type a number that identifies the network situation being reported or leave it set to the default.

The alarm code is a sequence of digits displayed on the pager. The maximum number of digits varies from pager to pager. If you don't supply an alarm code, a default value of 911 is used.

5. In the **Terminate** field, type the character that terminates the paging connection on your pager.

This character is a key used by the paging system to terminate the connection and send the page. It differs from system to system, but is usually # (pound sign) or * (asterisk). Consult your paging system manual to determine the correct key for your system. If you don't specify a key, the Paging action uses the default value #.

6. In the Paging Action window, select **OK**.

The Paging action appears in the **Rule Action** action list.

7. In the Action Router Rule Definition window, select **Save**. Select **Cancel** to close.

The PagingTest 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

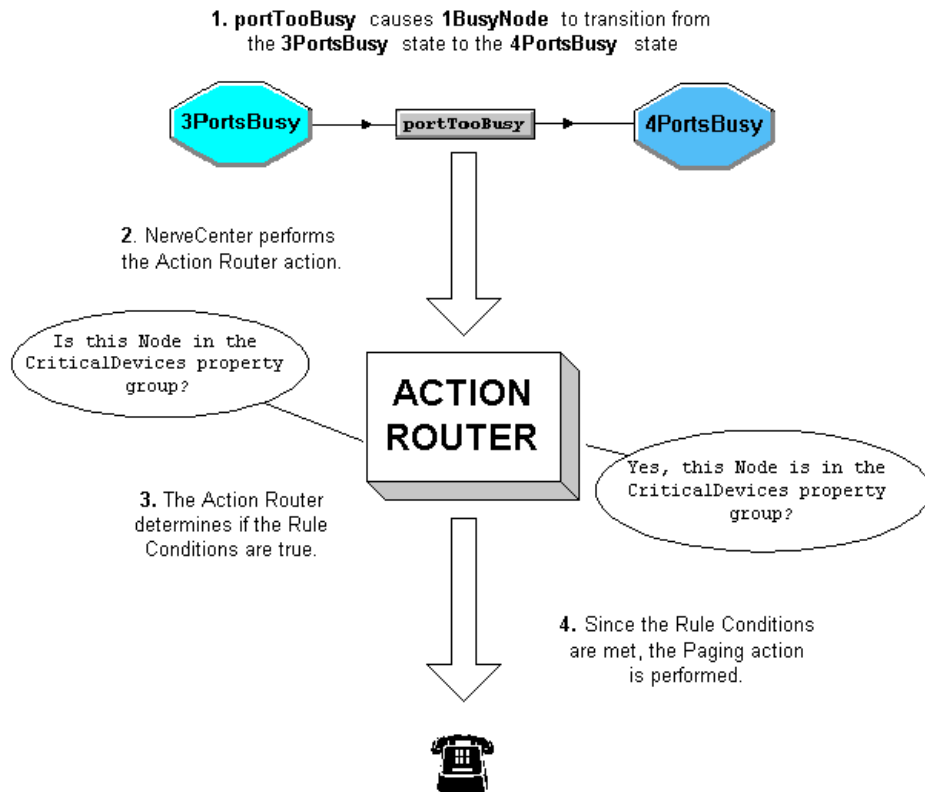


1. From the **Admin** menu, choose **Alarm Definition List**.
NerveCenter displays the Alarm Definition List window.
2. Open the **1BusyNode** alarm.
The Alarm Definition window for 1BusyNode appears.
3. In the state diagram, double-click on the portTooBusy transition between the 3PortsBusy state and 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.

TABLE 9-1. The Process that Occurs when the Action Router Action is Performed.**What is NerveCenter doing?**

High-traffic condition occurs on the fourth port on one of your nodes. The `portTooBusy` trigger causes the `IBusyNode` 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 paged.



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

Chapter 10, *How to Reset Your Environment* 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 pages and when your coworker is on call. You've suggested a schedule in which you'll receive pages Sunday through Wednesday, and your coworker will be on call the rest of the week. Modify the rule condition of the PagingTest rule to work within this paging schedule. Some of the actions you will take include:
 - a. Add to the Rule Condition for PagingTest an expression telling Action Router that it is only to perform the paging action Sunday through Wednesday.



TIP

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 PagingTest except for the days of the week and the pager's phone number.

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 144.

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 checklist 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

This last chapter explains:

- ◆ How to delete alarms, polls, and masks
- ◆ How to delete property groups

This chapter includes the following sections:

Section	Description
<i>How to Reset Your Environment</i> on page 131	Steps you through the process of resetting your environment.
<i>Summary of What You Learned</i> on page 136	Gives you the opportunity to see how much you have learned.

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. See *Deleting NerveCenter Objects* on page 132.
2. *Deleting Property Groups* on page 134

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 through this tutorial.

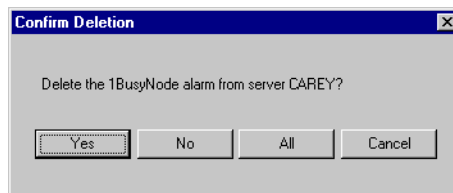
Each alarm should be highlighted as you select it.

TIP

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.



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 through 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 through 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



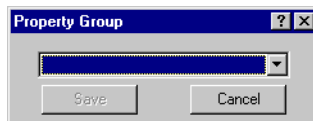
1. From the **Admin** menu, select **Node List**.

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.



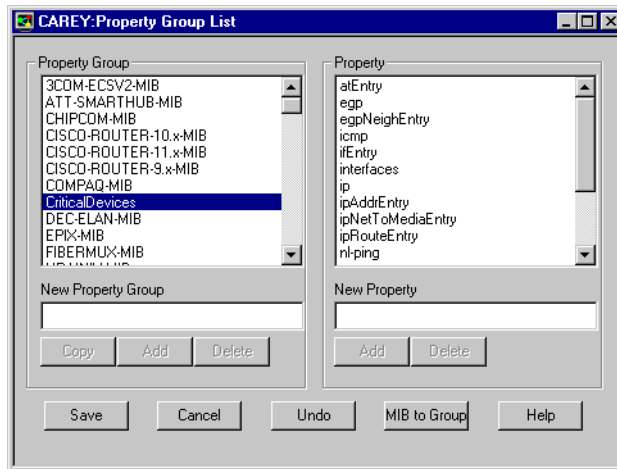
5. In the Property Group window, select an appropriate property group.

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.

The answers are listed by chapters:

- ♦ *Answers to the Chapter 1 Review Questions* on page 140.
- ♦ *Answers to the Chapter 2 Review Questions* on page 140.
- ♦ *Answers to the Chapter 3 Review Questions* on page 141.
- ♦ *Answers to the Chapter 4 Review Questions* on page 142.
- ♦ *Answers to the Chapter 5 Review Questions* on page 143.
- ♦ *Answers to the Chapter 6 Review Questions* on page 143.
- ♦ *Answers to the Chapter 7 Review Questions* on page 144.
- ♦ *Answers to the Chapter 8 Review Questions* on page 144.

Answers to the Chapter 1 Review Questions

Following are the answers to the *Review and Summary* on page 18:

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.

3. What time-saving feature is available for those logging on to a Windows NerveCenter Client?

The unified logon feature allows a Windows user to connect to a NerveCenter Server using the same name and password as that used to log on to Windows.

Answers to the Chapter 2 Review Questions

Following are the answers to the *Review Questions* on page 30:

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 3 Review Questions

Following are the answers to the *Review Questions* on page 44:

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. How many different MIB base objects can you use to form a complex poll condition?

Although you may include as many attributes as you like in a poll condition, each poll may use only one MIB base object.

4. 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 4 Review Questions

Following are the answers to the *Review Questions* on page 63:

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 5 Review Questions

Following are the answers to the *Review Questions* on page 80:

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 6 Review Questions

Following are the answers to the *Review Questions* on page 100:

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 7 Review Questions

Following are the answers to the *Review Questions* on page 116:

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 8 Review Questions

Following are the answers to the *Review Questions* on page 128:

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

I

A

Action Router
 defined 118
 defining a rule action 123
 defining a rule condition 120
 deleting rules 133
 role in a behavior model 84, 118
 rule condition 119
 use in an alarm 125
 using 117, 119
Action Router alarm action 118
Action Router Rule Definition
 window 120
Action Router Rule List window 119
alarm
 adding a state 60
 as a trigger generator 39, 66, 92
 creating 49
 defined 48
 deleting 132
 enabling 55
 masks 70
 role in a behavior model 48, 84
 use in a multi-tier behavior model
 108, 110
 using 47
alarm action
 Action Router 118
 adding 118
 Alarm Counter 86, 87
 Beep 53
 Clear Trigger 92, 94
 deleting 61
 Fire Trigger 38, 91, 94, 108
 Inform 89
 Log to File 70, 110
 Paging 111, 123
 role in a behavior model 48
 Set Attribute 94
Alarm Counter Action window 86

Alarm Counter alarm action 86, 87
Alarm Definition List window 50, 55
Alarm Definition window 50
alarm scope
 defined 104
 enterprise 104, 106
 instance 104, 107
 node 104, 106, 107, 110
 subobject 51, 57, 104, 106, 107,
 108
 use in a multi-tier behavior model
 107
 using 103
Alarm Summary window 57, 74
attribute
 in a poll condition 37, 38
B
base object
 as a property 22, 25
 in a poll condition 37, 38
Beep Action window 53
Beep alarm action 53
Before you begin 10
behavior model
 creating 85
 defined 84
 multi-tier 107, 108, 109, 112
 use of Action Router 84, 118
 use of alarm actions 48
 use of alarms 48, 84
 use of property group 22
 using 83
C
Clear Trigger Action window 92
Clear Trigger alarm action 92, 94
command line interface 12
Connect to Server window 16

counter 85, 87, 94

D

documentation
 conventions 5
 feedback 6

E

enterprise scope 104, 106
ERROR 94, 97, 98

F

Fire Trigger Action window 91, 108
Fire Trigger alarm action 38, 91, 94,
 108

G

generating a trap 73

I

Inform action 89
Inform Action window 88
instance scope 104, 107

L

log file 71, 74
Log to File Action window 71
Log to File alarm action 70, 110

M

mask
 as a trigger generator 39, 66, 70
 creating 67
 defined 66
 defining a trigger function 76

- deleting 133
 - using 65
 - Mask Definition window 68
 - Mask List window 67
 - multi-tier behavior model 107, 108, 109, 110, 112
- N**
- NerveCenter architecture 12
 - NerveCenter Client
 - defined 12
 - quitting 17
 - starting 14
 - using 9, 14
 - NerveCenter database 12
 - NerveCenter Server
 - connecting 16
 - defined 12
 - disconnecting 17
 - NerveCenter user interface 12
 - network traffic 10, 41
 - NO_PROP 51, 110
 - node
 - assigning a property group 22, 27, 134
 - Node Definition window 28
 - Node List window 27
 - node scope 104, 106, 107, 110
- O**
- online knowledgebase 8
- P**
- Paging Action window 111, 124
 - Paging alarm action 111, 123
 - Perl
 - use in a poll condition 37
 - use in a trigger function 77
 - use in this tutorial 11
 - Perl subroutine
 - as a trigger generator 39, 66
 - persistence, detecting 85, 88, 91
 - poll
 - as a trigger generator 34, 39, 66
 - creating 34, 35
 - defined 34
 - deleting 133
 - enabling 40
 - poll condition 37
 - using 33
 - poll condition
 - delta 38
 - modifying 41
 - writing 37
 - Poll Condition page 37
 - Poll Definition window 36
 - Poll List window 35, 42
 - Poll page 40
 - predefined trigger
 - ERROR 94, 97, 98
 - RESPONSE 99
 - property
 - creating 23, 26
 - defined 22, 25
 - use in alarms 51
 - using 21
 - property group
 - assigning to a node 27, 134
 - assinging to a node 22
 - creating 23, 24
 - defined 22
 - deleting 135
 - role in a behavior model 22
 - using 21
 - Property Group List window 135
 - Property Group window 29
- R**
- resetting conditions in a behavior model 112
 - resetting your environment 131
 - RESPONSE 99
 - rights of NerveCenter Users and Administrators 13
 - Rule Action page 123
 - rule action, defining 123
 - rule condition, defining 119
- S**
- Set Attribute Action window 95
 - Set Attribute alarm action 94
 - simple trigger 68
 - smart polling 41, 56, 97
 - state
 - adding 52, 60
 - role in a behavior model 48
 - State Definition window 52
 - state diagram
 - defined 51
 - designing 52
 - icons 58
 - modifying 58
 - State/Transition Size window 59
 - subobject scope 51, 57, 104, 106, 107, 108
- T**
- technical support 7
 - contacting 8
 - educational services 7
 - professional services 7
 - timer 87, 91, 94
 - transition
 - adding 52
 - circular 86
 - defined 54
 - role in a behavior model 48
 - Transition Definition window 53
 - trap 69
 - trap number
 - enterprise-specific 69, 76
 - generic 68, 69, 76
 - Inform actions 89
 - trapgen 73
 - trapgen, syntax 73
 - trigger
 - defined 39
 - defined by poll condition 37
 - trigger function 69, 76
 - Trigger Function page 77
 - trigger generator 39
- U**
- unresponsive node 94, 96
- V**
- variable bindings 69

Behavior Model Quick Reference Card

TO CREATE AN ALARM



1. In NerveCenter Client connected to a NerveCenter Server, choose **Admin > Alarm Definition List**, then click **New**.
2. Provide a name, property, and scope.

TO ADD A STATE TO THE ALARM



1. Click the **Add State** button.
2. Type a **Name** and select a severity.
3. Click **OK**. Resize and position the state icon.

TO ADD A TRANSITION



1. Select the **Add Transition** button.
2. Select alarm states from the From and To lists.
3. Select the trigger that will cause the alarm to transition.
4. Optional: select **New Action** and select an action. Add more actions as appropriated, then click **OK**. (See other side for a list of actions.)
5. Resize and position the transition icon.
6. When finished defining the alarm, select **Save**.

Trap	Name	Definition	SnmpTrapOID.0
0	ColdStart	The node is initializing. Its agent configuration or protocol may be altered.	1.3.6.1.6.3.1.1.5.1
1	WarmStart	The node is initializing. Neither its agent configuration nor its protocol will be altered.	1.3.6.1.6.3.1.1.5.2
2	LinkDown	A failure in one of the links in the agent's configuration.	1.3.6.1.6.3.1.1.5.3
3	LinkUp	One of the communications links in the agent's configuration is up.	1.3.6.1.6.3.1.1.5.4
4	AuthFail	The node received a protocol message that isn't authenticated.	1.3.6.1.6.3.1.1.5.5
5	EgpNeighLoss	An EGP peer of the node sending the trap is down. Peer relationships no longer exist.	1.3.6.1.6.3.1.1.5.6
6	EntSpecific	The occurrence of an event defined by the node vendor	n/a
-1	AllTraps	Monitors all SNMP-defined traps and vendor-defined traps	n/a

TO CREATE A TRAP MASK



1. In NerveCenter Client, choose **Admin > Mask List**. Then select **New**.
2. In the Name field, type a name.
3. Select the SNMP version appropriate to the type of trap you want to capture, either v1 or v2c/v3.
4. For a v1 trap, in the Generic field, select a generic trap number. If you select 6, enter in the Specific field an enterprise-specific trap number.
5. For a v2c/v3 trap, in the Trap OID field, enter the SNMPTrapOID for the mask.
6. To limit the mask to a particular vendor, select **Filter on Enterprise** and either **From** or **From**

Only. Then specify the enterprise OID in the Enterprise field (the enterprise's MIB must be incorporated in the NerveCenter MIB).

7. Select **Simple Trigger** or **Trigger Function**.
 - ♦ For a simple trigger function, type or select a trigger name in the **Simple Trigger** field.
 - ♦ For a trigger function, select the **Trigger Function** tab and type a function. Right-click in the function area to enter variable expressions and operators.
8. Select **Save**.

Alarm action	Description
Action Router	Performs actions based on criteria specified in the ActionRouter
Alarm Counter	Tracks recurring events and fires a trigger when a specified count is reached
Beep	Sounds a tone on all NerveCenter Clients connected to the NerveCenter Server managing the alarm instance
Clear Trigger	Clears a trigger
Command	Invokes a command or a script
Delete Node	Deletes the node that caused the alarm transition
Event Log	Stores data about a transition in the Event Log (Windows) or in the system log (UNIX).
Fire Trigger	Fires the specified trigger
Inform	Sends an Inform packet to OpenView or another NerveCenter Server
Inform Platform	Sends an Inform packet to CA Unicenter TNG, Tivoli TME, or Micromuse Netcool
Log to File	Stores data about a transition in a text file
Log to Database	Stores data about a transition in the NerveCenter database (Windows only)
Microsoft Mail	Sends mail using Microsoft Exchange (Windows only)
Notes	Displays notes about the alarm for viewing or editing
Paging	Sends a page to the specified phone number
Perl Subroutine	Executes an existing Perl subroutine
Send Trap	Generates an SNMP trap
Set Attribute	Changes select attributes of NerveCenter nodes, polls, masks, and alarms
SNMP Set	Issues an SNMP SetRequest command that changes the value of a MIB attribute
SMTP Mail	Sends mail using SMTP

TO CREATE A POLL

1. Open a NerveCenter Client and connect to a NerveCenter Server.
2. Select **Poll List** from the **Admin** menu.
3. In the Poll List window, select **New**.
4. In the Poll Definition window, select the **Poll** tab.
5. In the Name field, type a name.
6. Select a property from the Property list.



7. If the node doesn't use the default polling port, type the port number in the Port field.
8. Enter a Poll Rate by typing a number and by selecting Hours, Minutes, or Seconds.
9. Select the **Poll Condition** tab.
10. Select a base object for the Poll.
11. Type a condition in the Poll Condition field.
12. Select **Save**.

Note You may select attributes from the **Attribute** list by double-clicking. Right-clicking in the **Poll Condition** field will give you available expressions and operators.