



OpenService, Inc. White Paper

NerveCenter™ 3.8: An Overview



PublicationDate

Contents

- Introduction 1
- Overview of NerveCenter 2
 - What NerveCenter Does 2
 - How NerveCenter Correlates Events 3
- How Behavior Models Work. 4
 - Behavior Model Objects 4
 - Behavior Model Operation. 5
 - Alarms 7
- NerveCenter Components 9
 - The NerveCenter Client Console. 10
 - NerveCenter Discovery 11
 - NerveCenter Database Management 12
 - NerveCenter's Compiled MIB 14
 - Additional NerveCenter Utilities. 15
- Web-based Monitoring. 17
- NerveCenter Integrated with a Network Management Platform. 18
 - OVPA Integration. 19
 - IT/Operations Integration. 24
 - Universal Platform Adapter Integration 25
- Multiple NerveCenters 29
- Starting With NerveCenter 30
- Index 43

About Open Service, Inc.

Open (OpenService, Inc.) is the premier provider of network security management solutions that enable enterprises and service providers to continuously protect and manage mission-critical business information. More than 450 customers are using Open's SystemWatch and/or NerveCenter™ for network security management. Open's products are available globally through a network of VARs and direct sales. A privately held company based in Westborough, MA, Open is backed by venture capital and an equity stake taken by Veritas. For more information, please call 800-892-3646, or visit the Open web site at <http://www.open.com>.



Introduction

NerveCenter™ is an application that helps you monitor and manage a complex heterogeneous network. This document describes the components of NerveCenter and explains how these components interact with each other. You will also learn how NerveCenter can be configured to interact with external software, such as a network management platform.

This document contains the following sections.

Title	Description
<i>Overview of NerveCenter</i> on page 2	Explains what NerveCenter does and how it uses behavior models to detect important network events and track the status of devices.
<i>How Behavior Models Work</i> on page 4	Introduces the objects that make up behavior models and describes how they interact.
<i>NerveCenter Components</i> on page 8	Describes the components of NerveCenter in a simple standalone configuration.
<i>Web-based Monitoring</i> on page 17	Explains how NerveCenter can be integrated with a Web server to enable browser-based monitoring.
<i>NerveCenter Integrated with a Network Management Platform</i> on page 18	Describes how NerveCenter can be integrated with a network management platform.
<i>Multiple NerveCenters</i> on page 29	Specifies how multiple NerveCenters can be configured to work together and with a network management platform.

Overview of NerveCenter

NerveCenter is a management tool that automates the management of important network events. Through a process called event correlation, NerveCenter helps you track significant events, quickly identify the root cause of critical network problems, and initiate corrective actions. When a critical event occurs, NerveCenter can notify you or someone else using a variety of methods.

NerveCenter uses the Simple Network Management Protocol (SNMP) to acquire data about managed devices. NerveCenter additionally relies on Internet Control Message Protocol (ICMP) messages from your network to provide basic information about unresponsive devices.

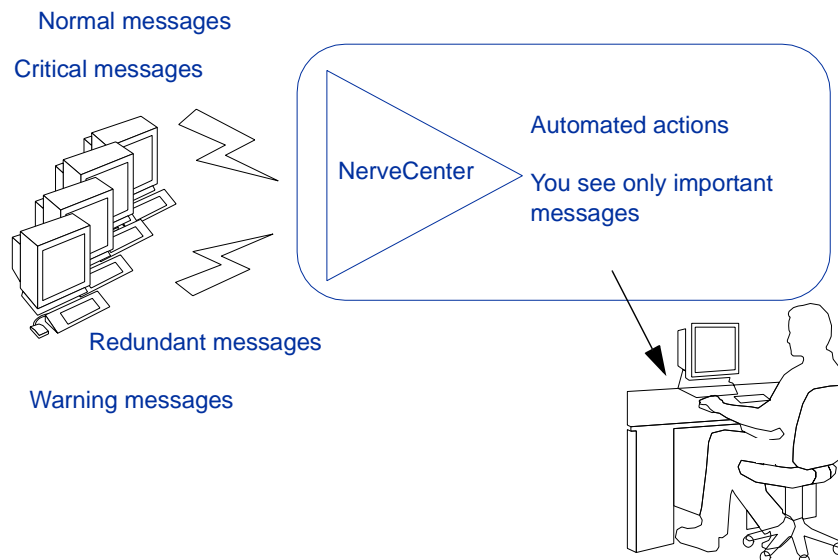
The following sections describe:

- ♦ *What NerveCenter Does* on page 2
- ♦ *How NerveCenter Correlates Events* on page 3

What NerveCenter Does

There are a number of management tools designed to identify network faults and send you some type of alert. However, these tools often flood the event console with large quantities of events and network data. For every critical or warning message indicating a possible problem, such as a router that's gone down, you also receive many events as a side effect of the router down condition. You further receive large numbers of messages that are defined as normal, such as simple power-up operations. As a result, you or other operators must sift through the plethora of data before you can identify the real problem and take corrective action.

Figure 1. NerveCenter Correlates and Filters Network Messages



NerveCenter greatly reduces the data on your screen and helps you quickly identify network problems. NerveCenter intelligently interprets and correlates the raw network data to determine whether a problem is serious enough to warrant human intervention. NerveCenter can then take automatic corrective action to solve the problem, inform the management platform, or notify the appropriate person. You see only the real problems and don't waste time on false or redundant alarms.

As a network management tool, NerveCenter provides intelligent, proactive management of local and wide area networks.

How NerveCenter Correlates Events

NerveCenter obtains data from SNMP agents running on managed nodes by processing incoming SNMP traps and polling the nodes for specific MIB values using SNMP Get requests. When a predefined network condition is detected, NerveCenter stores the event information in a finite state machine called an alarm. The alarm continues to track the status of the interface, node, or enterprise being monitored. The alarm waits for subsequent events or issues polls to determine if the condition warrants further action.

For example, by tracking events based on their persistence, NerveCenter can detect when a router experiences a persistent link-down condition. Upon detecting a downed communication link, NerveCenter waits to see if the link is restored within a given amount of time. If the link comes back up within the specified time, NerveCenter acknowledges the situation and returns the alarm to its normal (ground) state. However, if the link remains down, NerveCenter can perform a variety of actions based on your management strategy.

In another example, NerveCenter can detect when a group of nodes appears to be down or unreachable and then poll their parent router. If the router is down, further polling of those nodes is disabled until the router is back up.

To correlate and filter this data, NerveCenter relies on configurable models of network and system behavior, or behavior models, for each type of managed resource.

Behavior model

A behavior model is a group of NerveCenter objects that detect and handle a particular network or system behavior. A typical behavior model consists of an alarm with all its supporting polls and masks, though behavior models can have multiple alarms. Any managed device can be associated with one or more behavior models.

For a description of behavior models, see *How Behavior Models Work* on page 4.

Once a NerveCenter behavior model has identified a problem, it can take automatic actions, including notifying an administrator or a network management platform, executing a program or script, modifying the node's properties, changing SNMP values, and logging the critical data.

How Behavior Models Work

NerveCenter detects events and compares these events to conditions of interest defined in behavior models. To understand how behavior models work, you must understand the objects that make up behavior models and how they interact.

Behavior Model Objects

NerveCenter uses the objects listed in Table 1 to define behavior models:

Table 1. NerveCenter Objects Used with Behavior Models

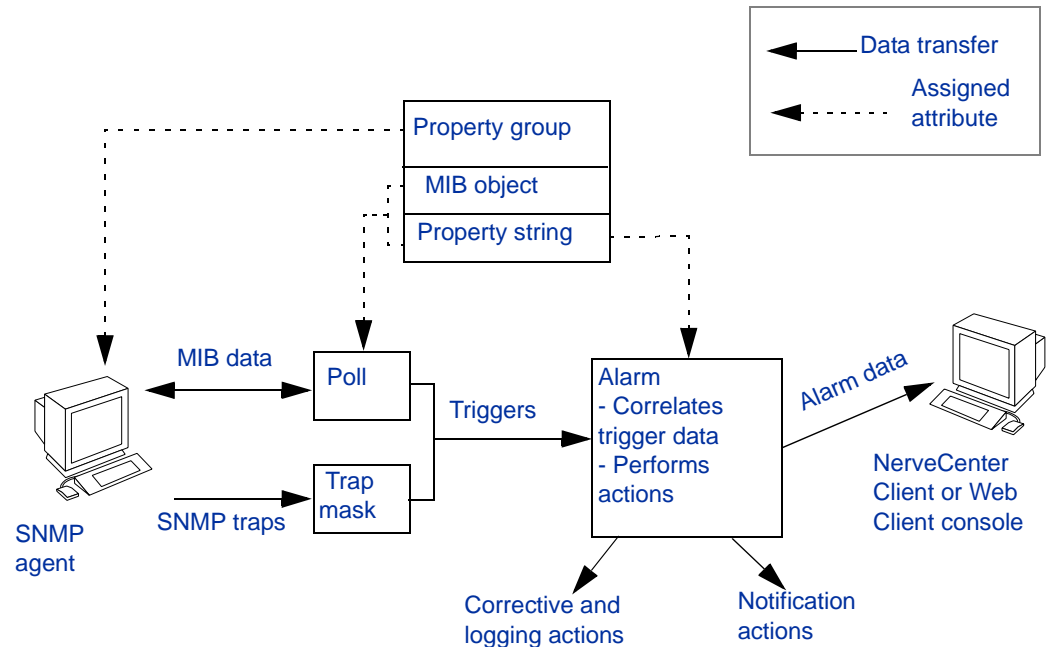
Object	Description
Node	Represents either a workstation, server, or a network device, such as a router, hub, or bridge.
Property	A text string that describes the type of node or one of the node's MIB objects. Polls and alarms use properties to target specific nodes.
Property group	A group of properties assigned to a node. The properties in the group allow behavior models to target the node.
Trigger	An internal event that is generated when a defined network event has been detected or a condition satisfied. Triggers are generated by polls and trap masks, as well as by other NerveCenter objects such as Perl subroutines and NerveCenter alarms.
Poll	Periodically solicits defined SNMP values from the agents running on targeted nodes and fires a trigger when specified conditions are met. Polls can fire multiple triggers, which can be detected by one or more alarms.
Trap mask	Detects a predefined type of SNMP trap. Trap masks can specify generic and specific trap numbers, enterprise OIDs, and variable binding data to match against incoming traps. A trap mask fires a trigger when the specified type of trap is detected.
Alarm	Detects triggers that cause the alarm to transition from one state to the next. Each transition is triggered by its own set of network data, which is defined in the poll or mask that generates the associated trigger. The alarm performs any actions assigned to a transition.

Behavior Model Operation

A behavior model is not a NerveCenter object. Rather, it is a collection of NerveCenter objects designed to monitor specific network devices, events, or behaviors.

Figure 2 shows the interaction of objects in a simple behavior model.

Figure 2. A Simple Behavior Model



When a trigger-generating object detects an event of interest on a node, it raises a trigger. This trigger generator might be a NerveCenter poll that looks for a specified network condition or a trap mask that detects a certain type of SNMP trap. NerveCenter alarms internally subscribe for triggers based on the state of the alarm. When an alarm detects its first trigger, the alarm transitions to a predefined state. Based on the state, the alarm ignores triggers that can no longer cause transitions and listens instead for those triggers that can. The alarm waits in that state until another trigger is received—either from the same or another trigger generator. This allows an alarm to react only to relevant events and poll only for relevant data. For more information about alarms, see *Alarms* on page 7.

How properties affect behavior models

Properties allow you to limit excessive polling and control which nodes are targeted by which alarms. A property is text string that describes the type of node or one of the node's MIB objects. A property group is a group of properties assigned to a node. Each node is targeted based on its assigned property group, which can contain many properties. Before an alarm can be applied to a node, the node's group of properties must include any property specified in the alarm or its associated polls. Additionally, before a node can be polled, the node's property group must contain the MIB base object that the poll is designed to evaluate. Assigning a node to a property group with multiple properties allows the node to be targeted by multiple alarms.

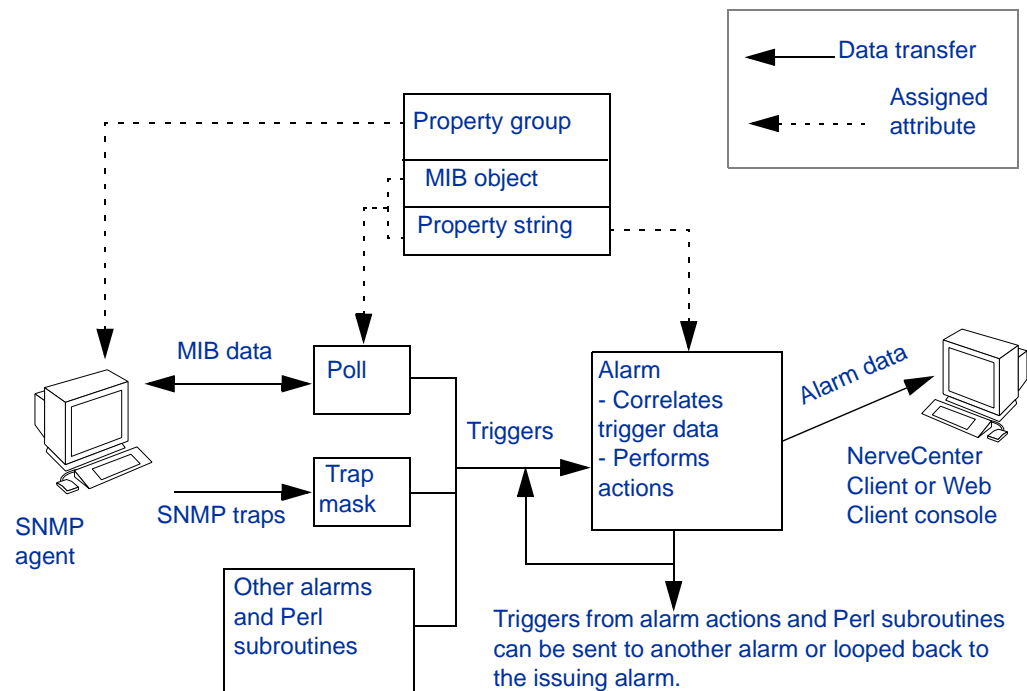
Other trigger generators

You've already seen how polls and trap masks detect pre-defined network events and fire triggers. NerveCenter has other generators as well.

Alarm transitions can trigger actions that, themselves, generate triggers. These trigger-generating actions can cause a transition in another alarm or transition yet another state in the same alarm. An alarm can also execute a Perl subroutine, which can fire a trigger when certain conditions are met. In both cases, the alarm serves as a trigger generator that provides an important function in a behavior model. These types of trigger generators provide an additional level of control over your alarms.

The following diagram shows the interaction of various trigger generators in a behavior model.

Figure 3. A Behavior Model with Triggers Generated by Alarms



Alarm actions

NerveCenter can implement a broad range of actions when an alarm is instantiated. A transition might cause NerveCenter to send a trap, email, or page. NerveCenter has its own type of message—an inform—that contains the variable bindings associated with the event that caused the alarm to transition. Inform messages can be sent to one or more destinations, including network management platforms and other NerveCenters. For each destination, you can specify a minimum severity level that your network conditions must meet before the informs are issued.

NerveCenter can call a Perl subroutine as an alarm action. The Perl subroutine might fire a trigger when certain conditions are met or execute other commands that effect changes to your network. Perl subroutines have access to associated alarm and trigger data for nodes that cause alarms to transition. Subroutines can then evaluate this information and conditionally take further action.

NerveCenter can also log alarm data and perform corrective actions such as setting SNMP attribute values. Finally, all actions can be performed conditionally using the NerveCenter Action Router. The Action Router performs actions based on user-defined criteria such as time of day, severity of the alarm, type of node, and so on.

NerveCenter allows you to monitor alarm activity in its Client or Web Client consoles. For a description of the NerveCenter Client, see *The NerveCenter Client Console* on page 10. For information about the Web Client, see *Web-based Monitoring* on page 17.

For more information

To learn more about NerveCenter's behavior models, refer to the following sources of information:

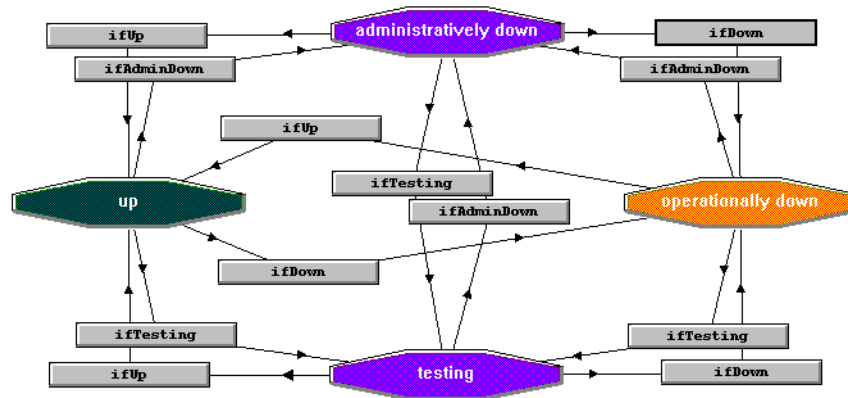
- ♦ The *Learning How to Create Behavior Models* and the *Designing and Managing Behavior Models* online guides shipped with NerveCenter contain complete descriptions and procedures for behavior models.
- ♦ Each model ships with its own HTML file that describes the model and includes related procedures. The HTML files are stored in the NerveCenter *installation/Model* directory.
- ♦ NerveCenter objects, such as alarms, include notes that describe the object and provide specific information related to that object. For example, the notes for an alarm include a list of triggers that transition the alarm, the objects that fire those triggers, and any actions associated with the alarm's transitions.

Alarms

Alarms are key to the operation of behavior models. An alarm is a finite state machine that defines the operational states it wants to detect and transitions from one state to the next when the proper trigger is received. Each transition is triggered by its own set of network data, which is defined in the NerveCenter object that generates the associated trigger. Transitions can also be driven by any alarm action that causes a trigger to be fired. If actions are associated with a transition, the server performs these actions each time the transition takes place.

Figure 4 shows the state diagram for a NerveCenter alarm called IfUpDownStatus. IfUpDownStatus monitors the operational status of interfaces on managed nodes. If an interface is down, an inform action notifies the network management platform.

Figure 4. IfUpDownStatus Alarm



Each state is depicted as an icon in the diagram. The state icons are color coded to represent a particular level of severity.

The alarm contains four states related to the status of an interface:

- ♦ **Up**—The interface is up. The current (operational) and desired (administrative) statuses are set to 1 (defined in RFC1213 as up). This state has a severity level of Normal.
- ♦ **Administratively down**—Both the current (operational) and the desired (administrative) statuses are set to 2, defined in RFC1213 as down.
- ♦ **Testing**—Either the current (operational) or desired (administrative) status is set to 3, defined in RFC1213 as test mode.
- ♦ **Operationally down**—The current (operational) status is down but the desired (administrative) status is up. This state has a high severity level of Major. All transitions leading to this state include an alarm action that sends an inform to NerveCenter or to the platform.

The ifStatus poll fires all the triggers that transition the IfUpDownStatus alarm. The poll is designed to evaluate the MIB base object attributes ifAdminStatus and ifOperStatus, which belong to the ifEntry base object in the interfaces MIB-II group. The poll fires four triggers—one for each of the four states previously described—based on the values returned for the two attributes. The alarm correlates the data it receives from the poll and transitions based on the combinations shown in Table 2.

Table 2. Correlation of IfEntry Attributes in the IfUpDownStatus Alarm

IfUpDownStatus Status	ifAdminStatus	ifOperStatus
Up	1	1
Administratively down	2	2
Testing	3	3
Operationally down	1	2

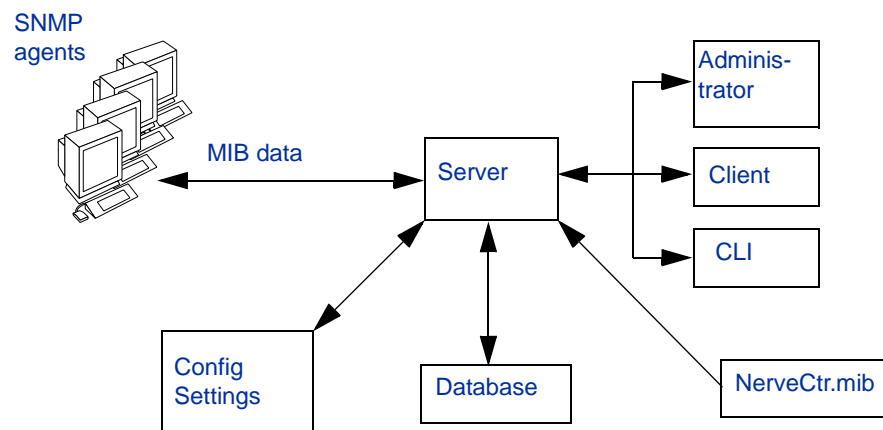
In this alarm, a trigger representing a condition other than up for either attribute—ifAdminStatus or ifOperStatus—transitions the alarm from the Up state. A subsequent trigger representing a condition other than up for the remaining attribute then transitions the alarm to a different state. The state to which it transitions depends on the combination of the values for the two attributes.

NerveCenter Components

This section introduces the main components that make up NerveCenter. For this discussion, we will analyze NerveCenter as a standalone network management application in its simplest configuration.

Figure 5 illustrates the main NerveCenter components and shows the direction in which data flows.

Figure 5. NerveCenter Components and Communication



The following table describes the components in the diagram:

Table 3. NerveCenter Components

Component	Description
NerveCenter Server	The NerveCenter Server carries out all of the major tasks that NerveCenter performs. It manages communication among all components, processes event data from managed nodes, performs automated actions, saves all behavior model data to the database, and loads the compiled MIB file. The server can run as a daemon on UNIX systems and as a service on Windows systems.
Database	A database or flat file is the repository for information about NerveCenter's behavior models and the nodes NerveCenter monitors. Database utilities allow you to convert, update, import, or export the NerveCenter database. Most operations can be performed without shutting down the NerveCenter Server.
MIB definitions	NerveCenter ships with support for standard RFC MIB definitions and specific vendor devices. Users can add and remove MIBs using a supplied MIB compiler.

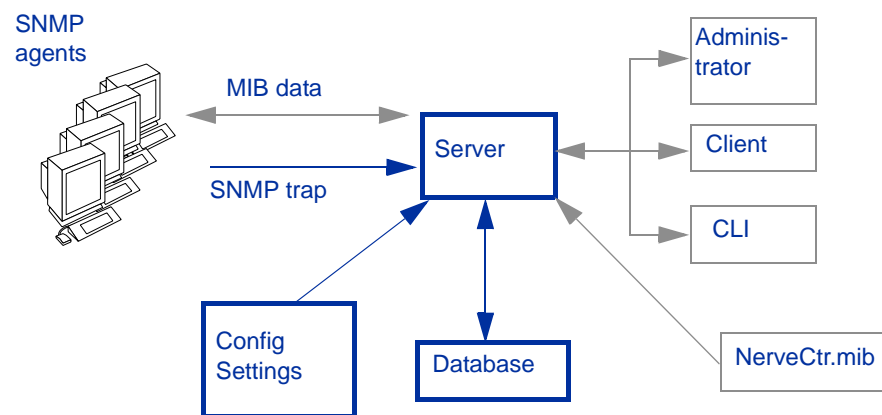
Table 3. NerveCenter Components (continued)

NerveCenter also provides a Web-based console that lets you monitor network activity from any machine that has a Microsoft Internet Explorer or Netscape Navigator browser installed. For a description of the NerveCenter Web Client, see *Web-based Monitoring* on page 17.

NerveCenter Discovery

Though you can manually add nodes to NerveCenter, it is easy to configure NerveCenter to discover nodes. When NerveCenter discovery is enabled, if the database does not already contain a node that sends it an SNMP trap or NerveCenter inform, NerveCenter adds that node to the database.

Figure 7. NerveCenter Configured To Discover Nodes



After installing NerveCenter, you can use the NerveCenter Administrator to define subnet IP filters that limit the sets of nodes NerveCenter can monitor. These filter values are stored with the NerveCenter configuration settings.

When a trap is received from a node, NerveCenter compares the node against those that are already in its database and confirms whether the node falls within the subnet IP filters you defined. Nodes that fall within the subnet range but are not in the database are added to the database.

Note NerveCenter can be configured to process SNMP traps from nodes that reside outside the defined subnet filters. These nodes, however, are not added to the database. For information about these traps or about configuring subnet filters in Administrator, refer to the *Managing NerveCenter* online guide that is shipped with NerveCenter.

You can also use NerveCenter's ping sweep utility, IPSweep, to populate your database. See *IPSweep* on page 16 for more information about IPSweep.

NerveCenter Database Management

NerveCenter includes two utilities that facilitate data management.

Table 4. Database Utilities

Utility	Description
SerializeDB	A conversion utility that transfers NerveCenter data to and from a serialized file.
DBWizard	An installation and upgrade utility available only on Windows. DBWizard works with SerializeDB and InstallDB to install, upgrade, or connect to a Microsoft Access or SQL Server database.

NerveCenter uses different methods of storing information on UNIX and Windows systems.

The following sections describe data management for each platform:

- ♦ *UNIX* on page 12
- ♦ *Windows* on page 12

UNIX

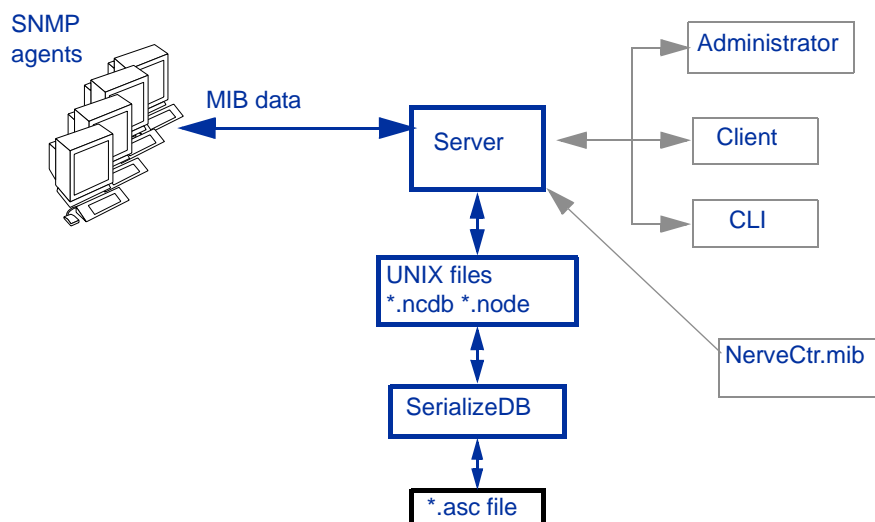
NerveCenter uses two files for data storage on UNIX:

- ♦ **nervecenter.ncdb**—Contains information about NerveCenter objects, such as polls, masks, and alarms.
- ♦ **nervecenter.node**—Contains information about the nodes that NerveCenter manages.

Both files are loaded into NerveCenter during installation. After installation, you can back up, restore, and transfer this data by using SerializeDB.

Figure 8 shows the UNIX database files.

Figure 8. NerveCenter Data Storage Files on UNIX



SerializeDB performs two operations:

- ♦ Converts UNIX *.ncdb and *.node files to a single database file, whose name you provide at the time of conversion. The database file has a binary format with an .asc extension.
- ♦ Converts the same or another *.asc file to two *.ncdb and *.node files.

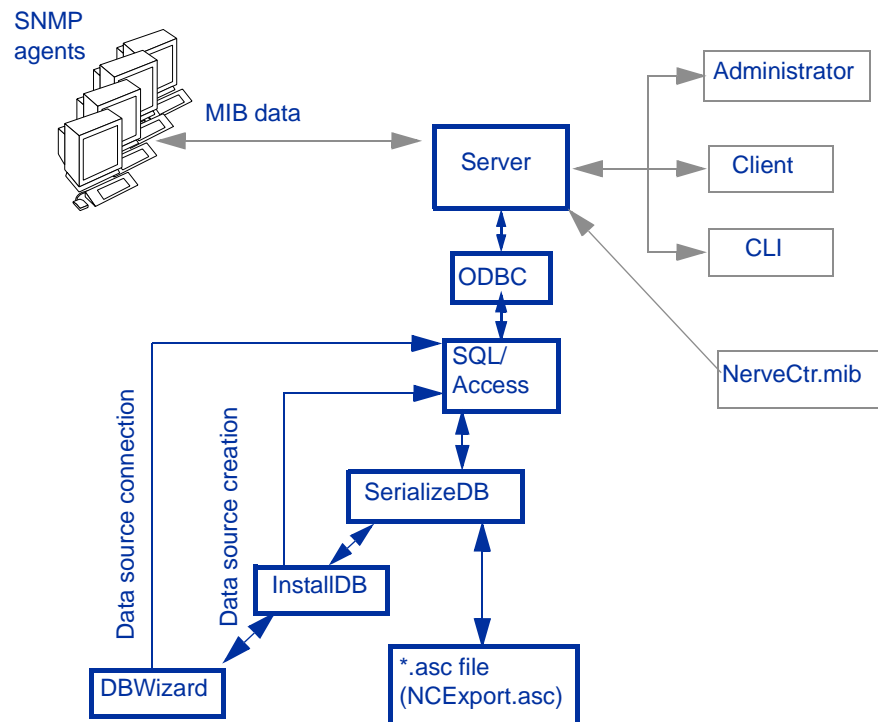
You can use the serialized file as a backup. Should you need to restore the backup, convert the file back to your original *.ncdb and *.node files. You can also transfer the serialized file to another NerveCenter machine and convert the file to *.ncdb and *.node files for that NerveCenter. Finally, you can transfer your original database files to a Windows NerveCenter and import the files to its database. For more information about multiple NerveCenters, see *Multiple NerveCenters* on page 29.

Windows

On Windows, NerveCenter stores its behavior model objects in a database that is accessed using ODBC.

Figure 9 shows the Windows database files along with the other NerveCenter components.

Figure 9. NerveCenter Database Components on Windows



NerveCenter ships with a serialized file that is imported into the database during installation. The file, NCEExport.asc, contains the base set of NerveCenter objects, such as polls, masks, and alarms. During installation, DBWizard allows you to enter settings, for example, an ODBC source and a SQL Server host. DBWizard invokes an installation tool called InstallDB to configure the SQL Server or Access database. InstallDB, in turn, invokes SerializeDB to convert NCEExport.asc to the proper database format. InstallDB can also be run standalone with SerializeDB to perform unattended database installations.

Afterward, if you upgrade to a new version of NerveCenter, DBWizard can be used to run an upgrade script against your database. Upgrade requirements for each new release are included in the *Release Notes* that accompany the release.

As with the UNIX environment, you can use SerializedDB to transfer data between your NerveCenter database and a serialized file. This lets you back up, restore, and transfer a database. On Windows, SerializedDB performs the following operations:

- ♦ Converts your database to a single database file. The database file has a binary format with an .asc extension.
- ♦ Converts an *.asc file to a database.
- ♦ Converts your database to NerveCenter flat files (*.ncdb and *.node), which can then be imported to another NerveCenter on a UNIX machine.
- ♦ Converts UNIX *.ncdb and *.node files to a SQL Server or Access database.

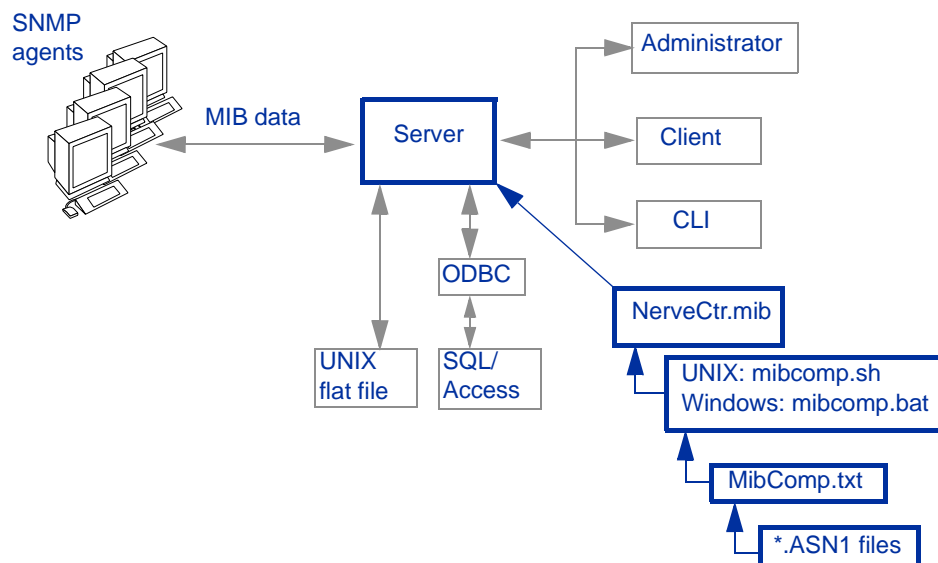
For information about working with multiple NerveCenters, see *Multiple NerveCenters* on page 29.

You can also use DBWizard as an interface that runs SerializedDB and InstallDB to import a serialized file that you specify in DBWizard.

NerveCenter's Compiled MIB

NerveCenter enables you to control which types of devices are managed based on their SNMP management information base (MIB) definitions. Figure 10 shows the NerveCenter MIB files along with the other NerveCenter components.

Figure 10. NerveCenter MIB Files



The compiled MIB file shipped with NerveCenter, *NerveCtr.mib*, contains definitions for Internet Standard RFC SNMP Versions 1, 2 and 3 agents as well as many vendor specific MIBs. *NerveCtr.mib* defines the management information available from an SNMP agent based on the MIBs the agent supports. This support information is contained in the .ASN1 file for each managed device. NerveCenter's MIB file enables you to construct SNMP requests for particular devices and to decode data received in SNMP traps.

`NerveCtr.mib` is compiled from definitions referenced in a text file, which by default is named `MibComp.txt`. `MibComp.txt`, in turn, points to the `*.ASN1` files you are using. To add support for new devices, modify `MibComp.txt` to include the new `*.ASN1` files, located within the *installation/Mib* directory. You must have the proper `.ASN1` file for each device that is defined.

After modifying the text file, you recompile the MIB file. NerveCenter's MIB compiler, `MibComp`, verifies each respective `.ASN1` file, compiles the definitions in the text file, and creates a new version of `NerveCtr.mib` that can then be loaded into the NerveCenter Server.

Additional NerveCenter Utilities

NerveCenter includes the following utilities that extend its capability and enhance its performance:

- ♦ **Trapgen**—Generates a trap or inform on a specified node.
- ♦ **Traprcv**—Receives traps and shows the traps in standard output.
- ♦ **ImportUtil**—Imports behavior models and/or changes the NerveCenter Server configuration.
- ♦ **IPSweep**—Pings nodes and sends SNMP traps to the NerveCenter Server, which then adds those nodes to its database.

The following sections briefly describe each utility.

TrapGen

TrapGen is a standalone utility that allows you to send an SNMP trap or inform to a particular device. From a command line or shell prompt, you can set the generic and specific trap numbers, provide the sender and recipient IP addresses, include an enterprise identifier, and specify the variable bindings.

Generating a trap is useful for testing trap masks and behavior models. When you send an SNMP trap to a NerveCenter Server, any trap mask configured to detect the specified trap values fires its trigger. If your trap mask and alarm are working properly, you see the alarm listed in your console.

Traprcv

The `traprcv` command displays the SNMP Trap messages received by the NerveCenter Trap service. This utility can be useful when debugging behavior models. When you run the `traprcv` command from a command line or shell prompt, any trap you receive is shown on the screen.

Traprcv can receive the following traps and informs:

- ♦ SNMPv1 Traps
- ♦ SNMPv2c Traps
- ♦ SNMPv2c Informs
- ♦ SNMPv3 Traps
- ♦ SNMPv3 Informs

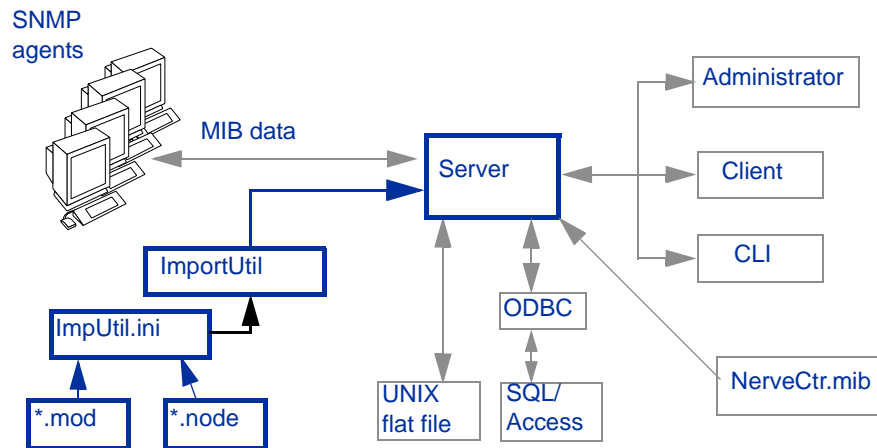
ImportUtil

ImportUtil is a utility that does the following from the command line:

- ♦ Imports behavior models and nodes to a NerveCenter Server.
- ♦ Imports configuration settings to a NerveCenter Server.

Figure 12 shows ImportUtil along with the other NerveCenter components.

Figure 11. NerveCenter ImportUtil Utility



ImportUtil imports the data referenced in the editable file `ImpUtil.ini`, which is located in the `installation\Sms` directory (Windows) or the `installation/userfiles` directory (UNIX).

You can edit NerveCenter configuration settings in `ImpUtil.ini` as well as specify the behavior models and nodes you want to import. You must have the proper `*.mod` or `*.node` file for the behavior models or nodes you want to import. When you have made the changes you want to `ImpUtil.ini`, run `ImportUtil` to import the specified information or files.

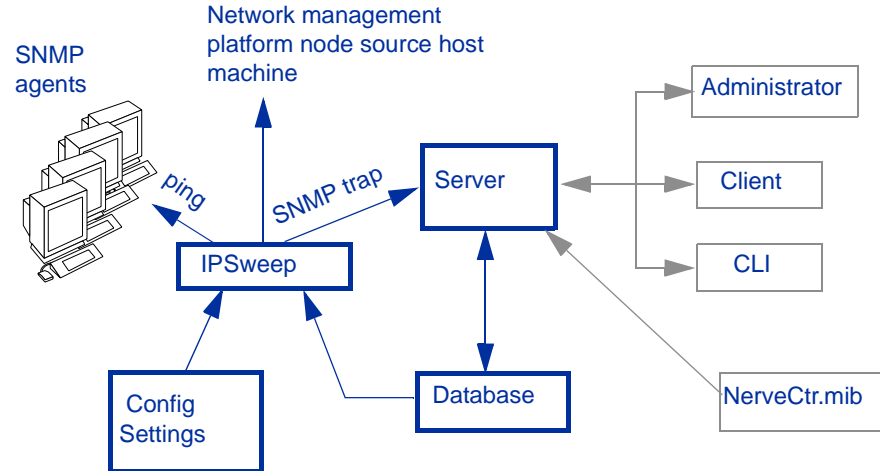
Note NerveCenter includes other import and export operations as well. These are built directly into the NerveCenter interface.

IPSweep

You can use NerveCenter's ping sweep utility, `IPSweep`, to populate your database. Figure 12 shows how `IPSweep` fits in with the other NerveCenter components.

`IPSweep` itself does not populate the NerveCenter server database. All it does is send a trap to NerveCenter if it gets a ping response from an IP address. The server can add the "unknown" node to its database once it gets a trap.

Figure 12. NerveCenter Configured to Discover Nodes



IPsweep extracts information from the NerveCenter configuration settings about the subnet IP filters defined for NerveCenter. IPSweep also obtains node information from the NerveCenter database, identifies nodes that fall within the subnet range but are not in the database, and sends those nodes a ping (ICMP echo request). If the ping returns a response, IPSweep issues an SNMP trap. The trap can be sent either to a host specified as the node source or, if no node source is specified, to the local NerveCenter. In a standalone configuration, the trap is sent to NerveCenter. Once NerveCenter receives the trap, the node is added to the NerveCenter database.

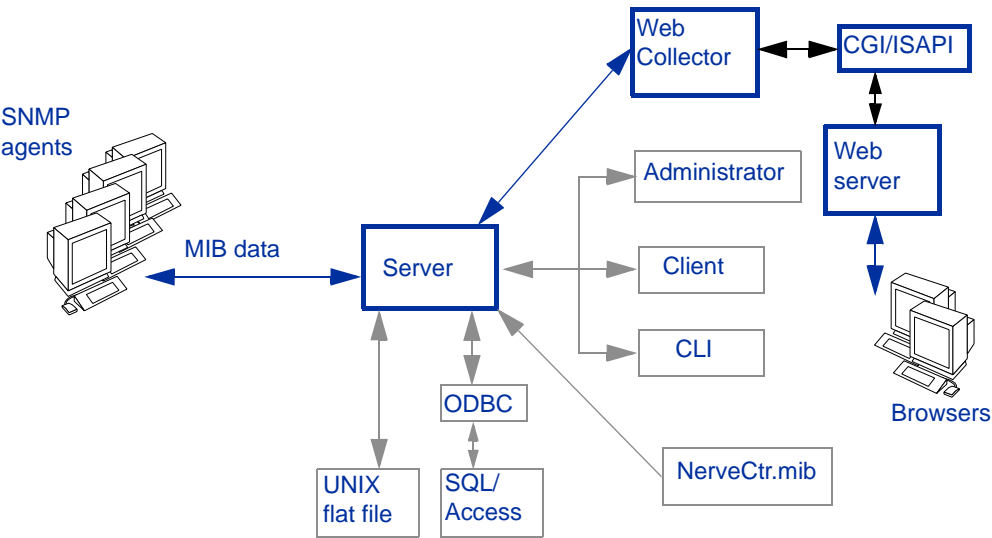
You run IPSweep by enabling the IPSweep behavior model. For more information about NerveCenter's IPSweep behavior model, refer to the *Learning How to Create Behavior Models* online guide shipped with NerveCenter.

Web-based Monitoring

NerveCenter provides a Web-based console, called the NerveCenter Web Client, that lets you monitor network activity from any machine that has a Microsoft Internet Explorer or Netscape Navigator browser installed. In order to use a Web-based console, you must install NerveCenter's Web support component, the Web Collector. The Web Collector can be installed on the NerveCenter Server host or on your Web server.

Figure 13 shows how the Web Collector works with NerveCenter components.

Figure 13. Web Server Integration



From a Web browser, you can monitor alarms for a NerveCenter Server by entering `http://webserver/NerveCenter` in the browser, where `webserver` is the name of the machine on which your Web server is running. After entering this URL, your Web server invokes a NerveCenter CGI/ISAPI application that connects you to the Web Collector. The Web Collector is responsible for transmitting information between the NerveCenter Server and the Web Client browser. Through the Web Collector, the NerveCenter Server prompts you to enter your NerveCenter user name and password. The NerveCenter Server verifies your logon and, in turn, provides alarm data. This alarm information is relayed through the Web Collector, CGI application, and Web server to your browser.

NerveCenter Integrated with a Network Management Platform

Using one of NerveCenter’s three platform adapters—OVPA, SEMSOPCA, and the universal platform adapter—you can integrate NerveCenter with a network management platform. The type and extent of integration varies with the platform you’re using. Table 5 lists the platforms supported along with the adapter and level of integration for each.

Table 5. NerveCenter Platforms, Adapters, and Integration Level

Platform	Adapter and level of integration
The following have full integration with NerveCenter: <ul style="list-style-type: none">◆ Hewlett Packard OpenView Network Node Manager	NerveCenter uses the OVPA adapter to do the following: <ul style="list-style-type: none">◆ Extract node and topology information◆ Insert events into event console◆ Change map symbol colors
The following exchanges information with NerveCenter: <ul style="list-style-type: none">◆ Hewlett Packard OpenView IT/O OperationsCenter	NerveCenter uses the SEMSOPCA adapter to do the following: <ul style="list-style-type: none">◆ Insert events into the event console◆ Receive and filter IT/O events

Table 5. NerveCenter Platforms, Adapters, and Integration Level (continued)

Platform	Adapter and level of integration
<p>The following receive events from NerveCenter:</p> <ul style="list-style-type: none"> ♦ Tivoli System TME Enterprise Console ♦ Computer Associates Unicenter TNG ♦ Micromuse Netcool/OMNIbus 	<p>NerveCenter uses the Universal Platform adapter to insert events into the event console.</p>

NerveCenter can operate with the network management platforms in various combinations, for example, by integrating with both OpenView Network Node Manager and Micromuse Netcool/OMNIbus. You can also integrate multiple NerveCenters with one or more platforms. Each NerveCenter might monitor one or more subnets and provide event information to the platform.

The following sections describe the integration configurations for each of the three NerveCenter adapters.

OVPA Integration

NerveCenter's OVPA platform adapter enables integration with Hewlett Packard's OpenView Network Node Manager. In this configuration, the platform typically provides node information to NerveCenter.

Note NerveCenter can monitor nodes that are outside the platform's domain or that are not managed by the platform. Such nodes can be added manually to the NerveCenter database, discovered from traps, or loaded using ImportUtil.

OVPA retrieves node and topology information from the platform and forwards this information to the NerveCenter Server. The Server can be configured to monitor all platform nodes, a subset of nodes, or nodes that have certain capabilities or system object IDs (OID). NerveCenter in turn forwards noteworthy events back to the platform by sending informs to OVPA, which relays them to the platform's event console.

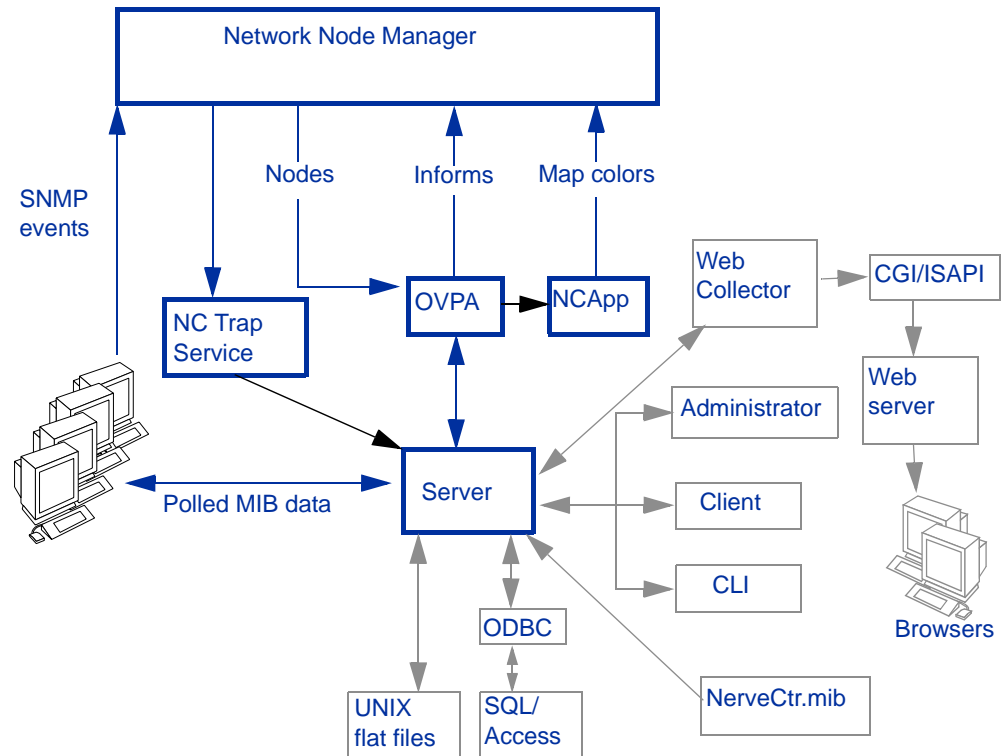
When NerveCenter sends informs to the platform, if color changes are required, OVPA sends a message to NerveCenter's NCAApp process, which then forwards instructions for color changes to the platform map.

Co-resident Installation

NerveCenter and the platform can have a co-resident installation, that is, they can be installed on the same system. Co-resident installations are supported both on Windows and UNIX.

Table 14 shows a typical co-resident platform integration.

Figure 14. NerveCenter Co-resident Platform Integration

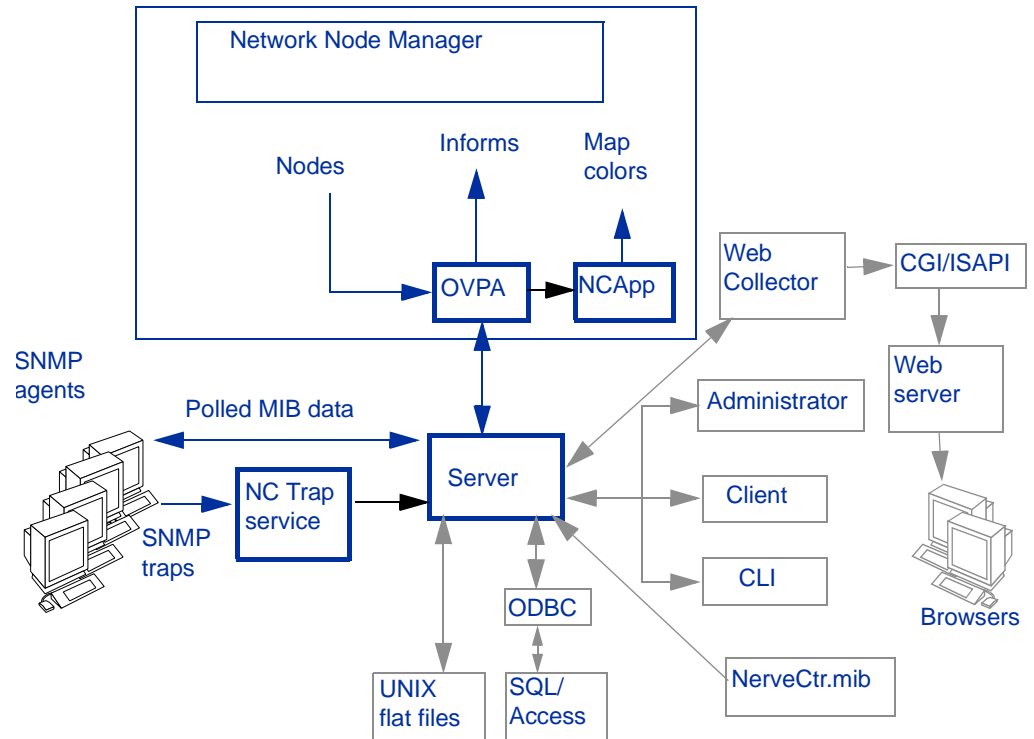


In a co-resident configuration, SNMP traps are sent to the platform. NerveCenter Trap service detects this information from the platform and forwards the traps to NerveCenter.

UNIX Installation

On UNIX, NerveCenter Trap service is always used whether or not the integration is co-resident. When NerveCenter and the platform are on separate machines, NerveCenter Trap service receives traps directly from the network and forwards the information to NerveCenter. Figure 15 shows NerveCenter and the platform installed on separate machines. In the figure, NerveCenter's OVPA and NCAApp components are located on the platform host machine.

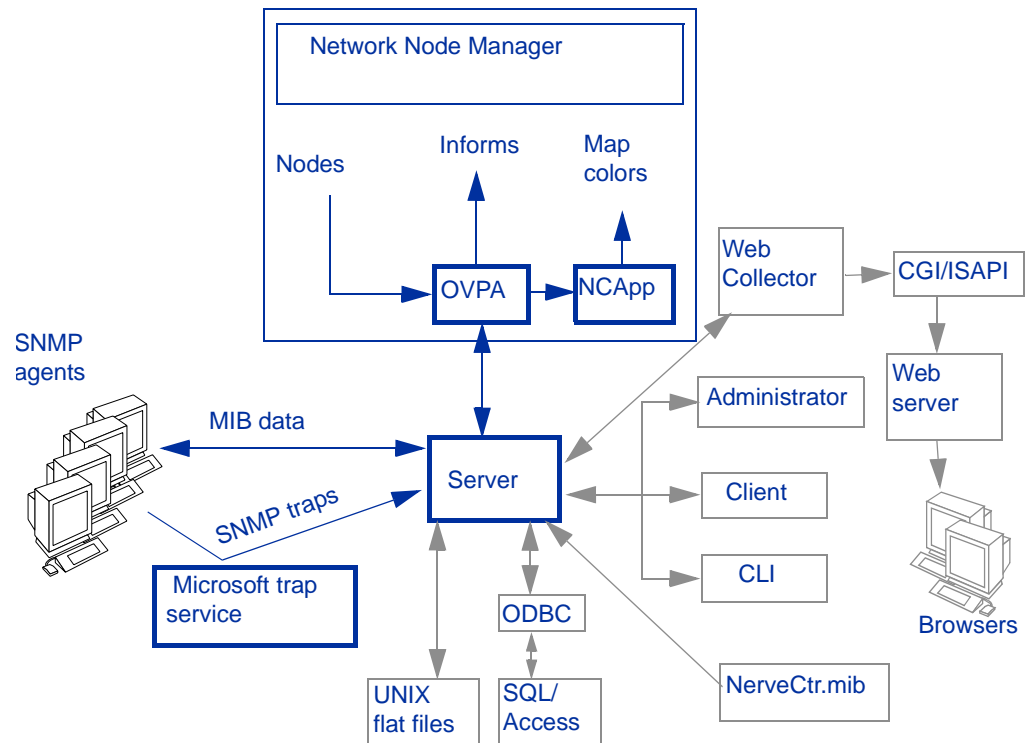
Figure 15. OVPA Integration on UNIX, not Co-resident



Windows installation on separate machines

Figure 16 shows OVPA integration with NerveCenter on Windows. NerveCenter and the platform are installed on separate machines. In the diagram, NerveCenter's OVPA and NCAApp components are located on the platform host machine.

Figure 16. NerveCenter OVPA Integration on Windows, not Co-resident



Note that NerveCenter Trap service is not required in the Windows configuration shown above. Traps are detected by Microsoft's trap service and forwarded to NerveCenter.

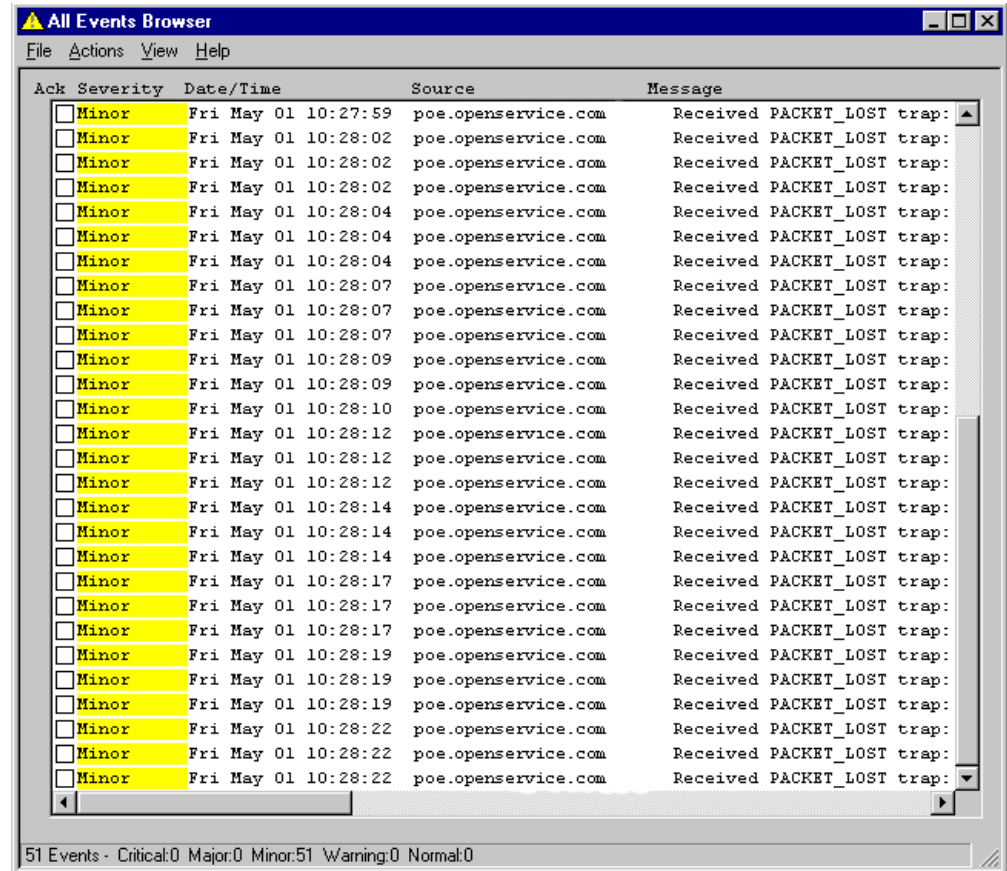
How NerveCenter integration helps the platform

For the platform, integration with NerveCenter means increased efficiency. NerveCenter can be configured to take over all event processing and minimize the number of events that appear in the platform's event console. NerveCenter does this by:

- ♦ Filtering out unimportant events.
- ♦ Correlating related events and notifying the platform only of the underlying problem.
- ♦ Handling an array of events through automated actions so that no notification is necessary.

Figure 17 shows an OpenView event browser that contains a number of events all caused by the same problem.

Figure 17. Redundant and Unfiltered Events

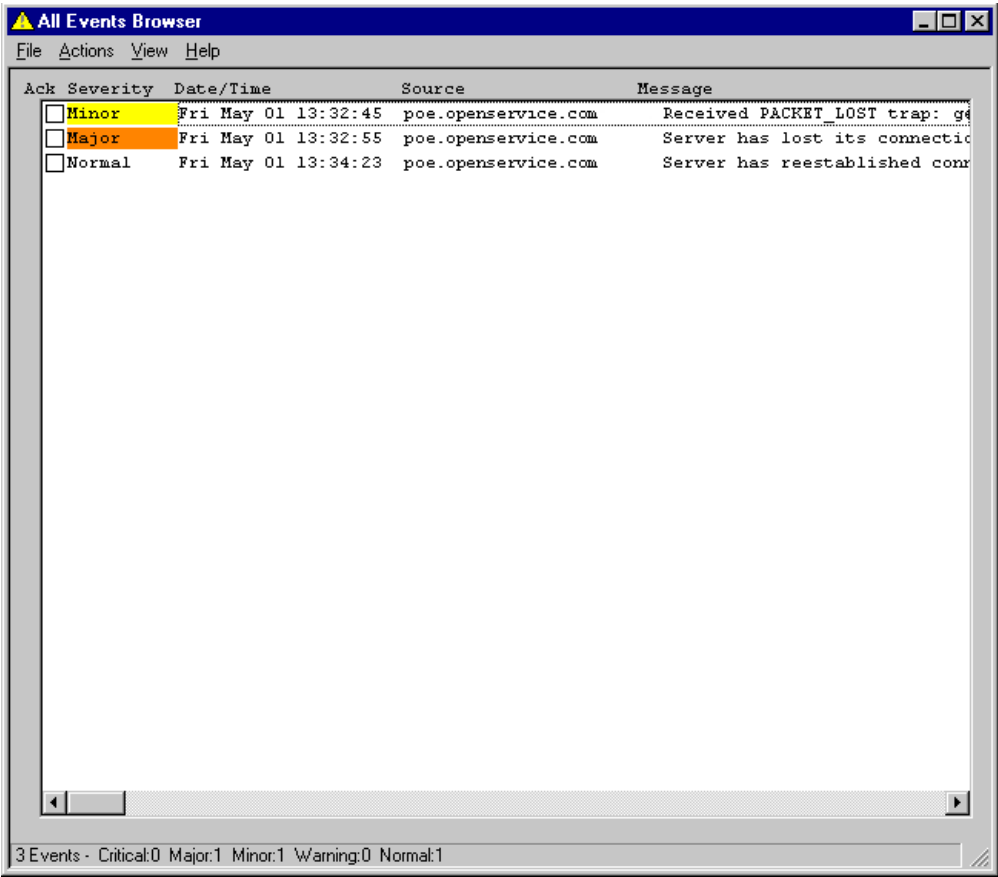


Ack	Severity	Date/Time	Source	Message
<input type="checkbox"/>	Minor	Fri May 01 10:27:59	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:02	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:02	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:02	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:04	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:04	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:04	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:07	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:07	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:07	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:09	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:09	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:10	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:12	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:12	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:12	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:14	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:14	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:14	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:17	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:17	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:17	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:19	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:19	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:19	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:19	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:22	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:22	poe.openservice.com	Received PACKET_LOST trap:
<input type="checkbox"/>	Minor	Fri May 01 10:28:22	poe.openservice.com	Received PACKET_LOST trap:

51 Events - Critical:0 Major:0 Minor:51 Warning:0 Normal:0

Figure 18 shows what might appear in the browser if NerveCenter were used to screen and correlate the conditions and pass on only important information to the platform event browser.

Figure 18. Filtered, Important Events

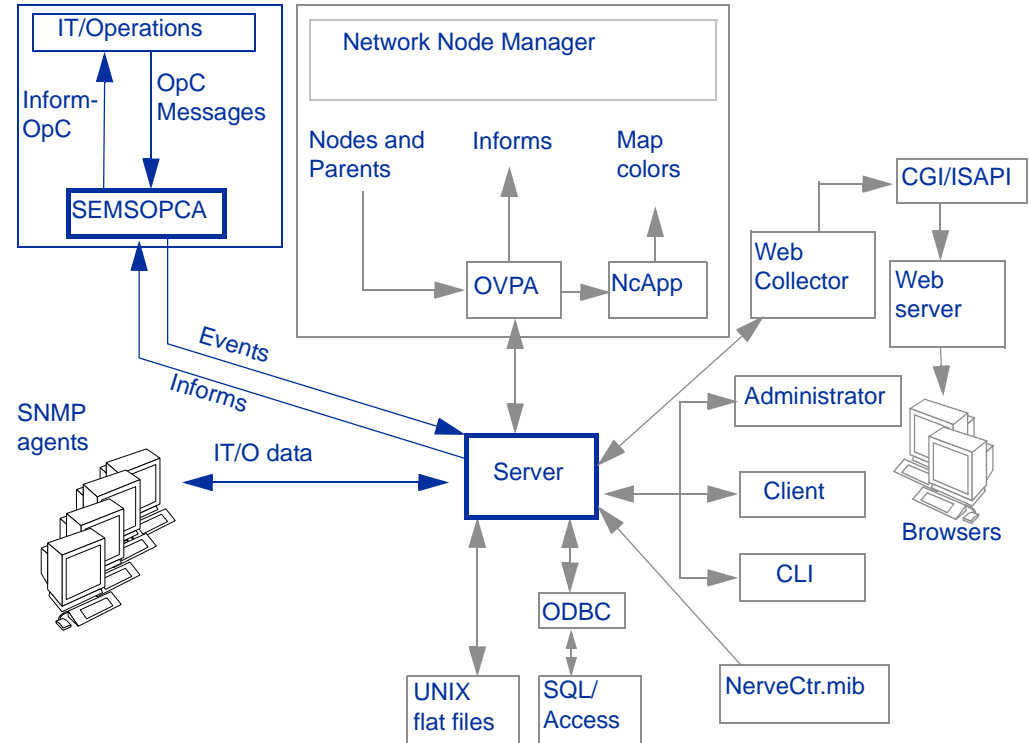


IT/Operations Integration

When you install OpenView integration on a system that contains Hewlett Packard OpenView IT/Operations, the installation script gives you the option of configuring integration with IT/Operations. If you choose this setup, NerveCenter installs the SEMSOPCA adapter as well as the OVPA adapter. The SEMSOPCA adapter enables NerveCenter to exchange information with IT/Operations.

Figure 19 shows IT/Operations integration with NerveCenter. In the diagram, NerveCenter’s SEMSOPCA component is located on the IT/Operations host machine.

Figure 19. NerveCenter IT/Operations Integration



The IT/Operations agent software, including SNMP agents, must be loaded onto each managed node. IT/Operations sends events to SEMSOPCA, which forwards them to NerveCenter. To accomplish this, IT/Operations must be configured to divert messages to NerveCenter.

NerveCenter detects the IT/Operations events by using special masks called OpC masks. In return, NerveCenter notifies IT/Operations of significant events by sending InformOpC messages, which are inform messages that have been customized to include IT/Operations information. InformOpC messages are sent to SEMSOPCA and then relayed to the IT/Operations message console.

Universal Platform Adapter Integration

NerveCenter's universal platform adapter, paserver, enables NerveCenter to send informs to the following platforms:

- ◆ Tivoli TME Enterprise Console on UNIX (Sun Solaris)
- ◆ Computer Associates Unicenter TNG on Windows
- ◆ Micromuse Netcool/OMNibus on UNIX and Windows

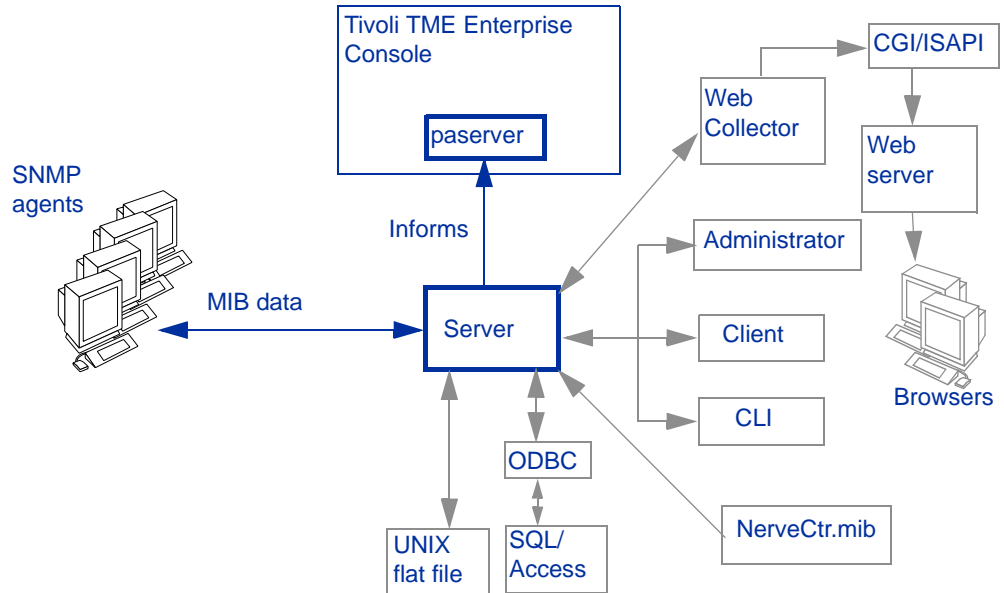
The adapter is typically installed on the network management platform host machine, though there are other possible configurations. Other NerveCenter components, including the Server, can be installed on the same machine or on a different machine.

The following sections describe a typical setup for each platform.

Tivoli TME Enterprise Console

NerveCenter can forward important events to the Tivoli TME Enterprise Console on Sun Solaris. Figure 20 shows TME Enterprise Console integration along with the other NerveCenter components.

Figure 20. Tivoli TME Enterprise Console Integration

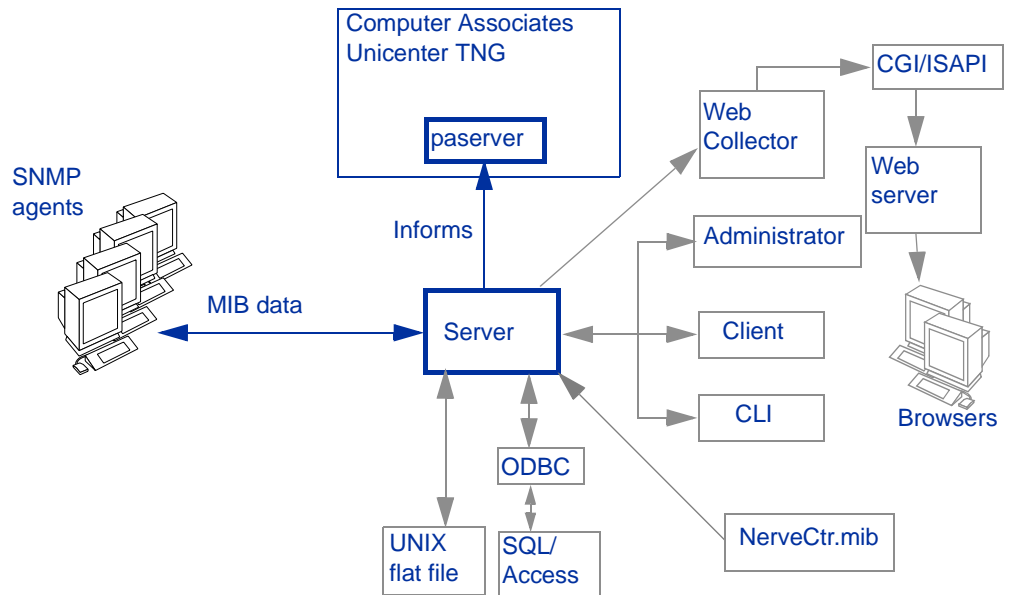


The NerveCenter Server forwards noteworthy events to the platform by sending inform messages to paserver, which relays the informs to the TME Enterprise Console. NerveCenter includes information about the associated node, the MIB values that were evaluated, the alarm that generated the inform action, and the severity of the states to and from, which the alarm transitioned when the inform was sent.

Computer Associates Unicenter TNG

NerveCenter can forward important events to Computer Associates Unicenter TNG on Windows. Figure 21 shows Computer Associates Unicenter TNG integration along with the other NerveCenter components.

Figure 21. Computer Associates Unicenter TNG Integration

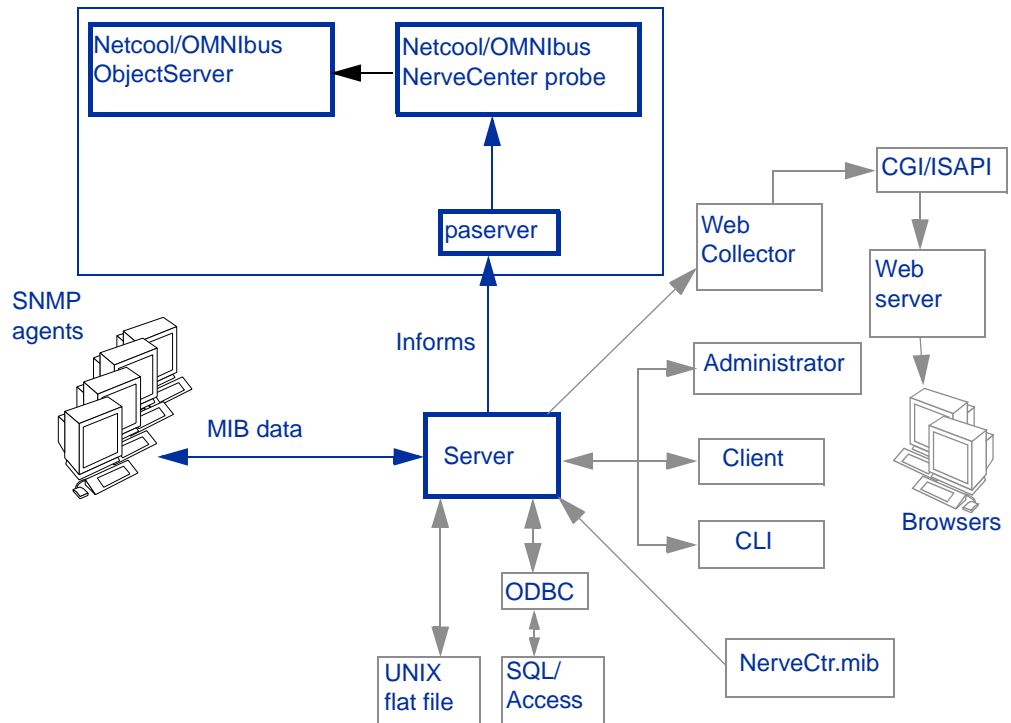


The NerveCenter Server forwards noteworthy events to the platform by sending inform messages to paserver, which relays the informs directly to the Unicenter TNG console. NerveCenter includes information about the associated node, the MIB values that were evaluated, the alarm that generated the inform action, and the severity of the states to and from, which the alarm transitioned when the inform was sent.

Micromuse Netcool/OMNIBus

NerveCenter can forward important events to Micromuse Netcool/OMNIBUS on Windows and on UNIX. Figure 22 shows Netcool/OMNIBUS integration along with the other NerveCenter components.

Figure 22. Micromuse Netcool/OMNIBUS Integration



To integrate the universal platform adapter with Micromuse Netcool/OMNIBus, you must obtain a NerveCenter probe from Micromuse. NerveCenter's inform messages travel from NerveCenter to the platform adapter, then to the probe, and finally to the Netcool/OMNIBus ObjectServer. You can customize the inform messages by editing the `nervecenter.rules` file located in `$OMNIHOME/probes/platform`.

Micromuse recommends that the probe be installed on the same machine as paserver, and this configuration is the default setting for paserver. See your Micromuse documentation for details.

The NerveCenter Server forwards noteworthy events to the platform by sending inform messages to paserver, which relays the informs directly to Micromuse Netcool/OMNIBus. NerveCenter includes information about the associated node, the MIB values that were evaluated, the alarm that generated the inform action, and the severity of the states to and from, which the alarm transitioned when the inform was sent.

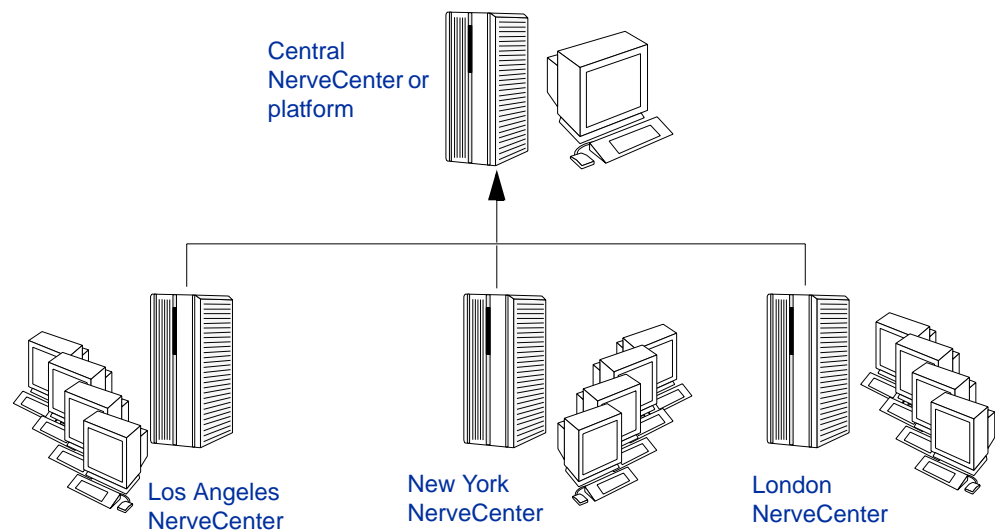
The NerveCenter Server and paserver are not required to be installed on the same system as Micromuse Netcool/OMNIBus and the probe. NerveCenter Server can be installed on any supported platform and still relay messages through paserver to Netcool/OMNIBus on Windows, Solaris, or HP-UX.

Multiple NerveCenters

If your network is large or distributed over a wide area, you may need to have more than one NerveCenter to manage all your devices. You may want to manage certain subnets independent of one another, or you might configure NerveCenter so that all polling is accomplished on local area networks rather than across a wide area network. Using this capability, you can minimize bandwidth by limiting the type of information to be monitored for each subnet and the number of nodes to be polled. Another aspect of distributed management is fault tolerance—your important resources are not all located in one place.

A distributed NerveCenter configuration can reduce network overhead while providing some form of centralized management. For example, say that your company consists of a central office and several small branch offices. Local NerveCenters could manage the branch offices and forward important information to a central NerveCenter and/or network management platform. Because the NerveCenter Servers can run as a daemon on UNIX systems or as a service on Windows, the branch NerveCenters could be managed remotely using the NerveCenter Client, the command line interface, or utilities like ImportUtil. Figure 23 illustrates centralized management of multiple NerveCenters.

Figure 23. Multiple NerveCenters



As events for the local NerveCenters are detected and filtered, each NerveCenter sends inform messages to the central NerveCenter or network management platform.

Starting With NerveCenter

These are just some of the organizations trust their network management to NerveCenter, along with over four hundred others, so you can be confident that it works.



In order to manage your network successfully with minimal resources, an automated real-time, scalable fault correlation, suppression and root cause analysis time solution such as NerveCenter is essential. More information can be found on the Internet at <http://www.open.com> or by calling Open at (508) 599 2000. An easy way to review the product's capabilities is to arrange an Internet demo, which can quickly show you the solution's benefits without taking up significant time or expense.

Glossary

Action Router	<p>Performs standard NerveCenter alarm actions in response to detected network conditions and based on additional conditions that you specify.</p> <p>See also <i>alarm action</i> on page 31.</p>
Administrator rights	<p>Users with administrator logon rights can customize NerveCenter and save changes to the NerveCenter database. In the NerveCenter Client, administrator rights are required to create or modify the objects used in behavior models. Those with administrator rights belong to the NerveCenter Admins group on Windows or to ncadmins on UNIX. They can also log in as administrator on Windows or root on UNIX.</p> <p>See also <i>user rights</i> on page 42.</p>
alarm	<p>A NerveCenter object that detects a trigger generated by a poll, trap mask, OpC mask, Perl subroutine, or another alarm. The alarm is a finite state machine that transitions from one state to the next and performs any actions assigned to a transition. Each transition is triggered by its own set of network data, which is defined in the associated trigger generator. When an alarm detects its first trigger, the alarm transitions to the next state, where it remains until another trigger is received—either from the same or another trigger generator. The sequence of transitions enables NerveCenter to monitor persistent, simultaneous, or sequential events that, taken together, indicate a critical or important condition.</p>
alarm action	<p>A NerveCenter automated response that helps you manage network activity and stay informed about network conditions. You can assign one or more actions to any transition in an alarm state diagram. Actions fall into four main categories: notification, logging, triggering other alarms, and correcting network conditions. Examples include sending e-mail, issuing a page, logging data, sending an SNMP trap, executing a command, and sending an Inform message to a network management platform. In addition, NerveCenter actions can be performed conditionally based on criteria that you specify using the Action Router.</p>
alarm instance	<p>A single instance of a detected network event. Each instance is one active occurrence of an alarm definition that tracks a current network or system condition through its own copy of the alarm's state diagram. For example, one alarm might have five distinct alarm instances, each tracking the same condition on a different node.</p> <p>See also <i>alarm scope</i> on page 31.</p>
alarm scope	<p>A setting that determines whether an alarm instance monitors a subobject (for example, a port), several MIB objects on a subobject, a node, or an entire enterprise. Scope is assigned in an alarm's definition window.</p> <p>If an alarm is using node scope, each alarm instance tracks the alarm's states for a single, distinct node. If an alarm is using subobject scope, each instance tracks the alarm's states for a MIB base object on a node, for example, an interface on a router. Instance scope alarms track instances for every interface or port that fits the polled condition regardless of the base object. Enterprise scope alarms track an event for the enterprise as a whole.</p>
alarm severity	<p>An indication of the urgency of a detected event. When creating an alarm, you assign each state a severity level, which is defined by a name and unique color. NerveCenter ships with severity options ranging from normal to critical for a fault condition, and</p>

from very low to saturated for a traffic condition. If your network management platform has event severities, you can map NerveCenter's severities to match those on your platform.

alarm state

Corresponds to a network condition that an alarm is monitoring. To monitor certain network conditions, or states, you must specify these states in the alarm's state diagram. Each state listens for certain triggers. Once the correct trigger is fired, the alarm transitions to the corresponding state.

Alarm state diagram

Specifies which detected conditions are correlated, in what order, and which responses (if any) are assigned to each stage. For each alarm that you create, you design a state diagram to specify the states you want to monitor. The state diagram can detect persistent, simultaneous, or sequential events that, taken together, indicate a critical or important network condition. Incoming triggers transition the alarm from one state to another.

alarm transition

A change from one alarm state to another, prompted by a trigger. When an alarm transitions from its ground state, the transition creates an alarm instance. Further triggers affect the current alarm instance, each generating a transition in the alarm. During a transition, NerveCenter carries out any alarm actions that are assigned to the transition.

ASN.1 file

MIB base objects are defined using Abstract Syntax Notation One (ASN.1), a language that's understood by network management protocols. The ASN.1 language is described in the ISO documents ISO.8824 and ISO.8825.

See also *Management information base (MIB)* on page 35.

attribute

NerveCenter terminology includes two meanings for the word attribute:

- ◆ A MIB object that contains an actual value. For example, sysContact, ifInOctets, and ipInDiscards are all attributes in the industry-standard MIB-II because they contain strings, numbers, or other values. The value associated with an attribute can be static (for example, the speed of the interface) or dynamic (for example, an entry in a routing table).
- ◆ A value that you assign to a NerveCenter object. For example, if you assign a property to a poll, that property is an attribute of the poll. NerveCenter's Set Attribute alarm action allows you to change certain node, poll, trap mask, or alarm attributes when a specified network condition is detected.

authentication

In SNMP v3 communication, the process of confirming the parties (i.e. entities) that communicate with each other as well as the timeliness of the messages received at an SNMP v3 entity.

authentication key

The seed used to generate a message digest, as specified in the authentication protocols, to authenticate SNMP v3 messages. Both sender and receiver generate the digest, and the receiver matches the generated digest against the digest accompanying the message received to authenticate the message.

authentication protocol

Protocol used with SNMP v3 communication that allows you to verify the sender and timestamp of a message. Two authentication protocols are currently defined: HMAC-MD5-96, which is based on MD5, and HMAC-SHA-96, which is based on SHA-1 cryptographic algorithms. NerveCenter supports both protocols.

AuthNoPriv	Security level that requires message authentication services to be used while communicating with an SNMP v3 entity.
AuthPriv	Security level that requires both the message authentication and message encryption services to be used while communicating with an SNMP v3 entity.
base object	See <i>MIB base object</i> on page 36.
behavior model	<p>The group of all alarm, trigger generator, and property group definitions required to detect and handle a particular network or system behavior. A typical behavior model consists of an alarm with all its supporting trigger generators, though behavior models can have multiple alarms. Any NerveCenter object can be associated with one or more behavior models.</p> <p>You can customize the behavior models that ship with NerveCenter as well as create new behavior models.</p>
boots	See <i>engine boots</i> on page 34.
CBC-DES	A privacy protocol as specified in SNMP v3 specifications to be used for message encryption in communication between two SNMP v3 entities with AuthPriv security level. This protocol is supported by NerveCenter.
Computer Associates Unicenter TNG	A network management platform that can be configured to receive NerveCenter Inform messages. The Unicenter platform host requires NerveCenter's universal platform adapter to communicate with NerveCenter.
context	Every communication between two SNMP v3 entities takes place on behalf of a <i>user</i> (a uniquely identified entity in the SNMP v3 management domain) in some <i>context</i> (a uniquely identified entity for access control generally configured on conventional SNMP v3 agents/nodes) with any one of three security levels. These three parameters (user, context, and security level) are used together by VACM (View based Access Control Mechanism defined in SNMP v3 specifications) to grant access to any MIB data at the agents/nodes. Context can be thought of as the MIB information available to a particular user who seeks information from an agent using a certain security level. By default, the NerveCenter user is NCUser and the NerveCenter context is NCContext.
DBWizard	<p>The utility that enables you to create or convert a NerveCenter database on Windows NT. Run the Database Wizard, DBWizard.exe, from your Start menu.</p> <p>See also <i>SerializeDB</i> on page 40.</p>
DES	A privacy protocol as specified in SNMP v3 specifications to be used for message encryption in communication between two SNMP v3 entities with AuthPriv security level. This protocol is supported by NerveCenter. See also CBC-DES.
digest	The hashing code generated for message authentication using authentication key. This digest is appended to the message being sent out by a sending SNMP v3 entity. The receiving SNMP v3 entity separate out this digest from the message, generate the digest again from the message using the locally available authentication key and then compares the two digests for message authentication.

discovery	The process of discovering nodes and adding them to the NerveCenter database. When you enable discovery, if the NerveCenter database does not already contain a node matching the source of an SNMP trap or NerveCenter Inform, it adds that node to the database. You can also customize NerveCenter's Discovery behavior model, which uses a TCI/IP sweep program (ipsweep.exe) to populate your node list. Alternatively, the NerveCenter node list can be populated from your network management platform.
Downstream alarm suppression	A NerveCenter behavior model that uses parent-child topology information to determine the status of nodes and suppress polling on nodes that are down.
engine boots	Number of times an SNMP v3 engine has been started or re-initialized (i.e., booted) since its engine ID was last configured.
engine ID	An SNMPEngineID value with a length of 12 octets that uniquely identifies an SNMP v3 engine within the SNMP v3 management domain. There is one SNMP engine ID for every instance of a NerveCenter Server.
engine information	An SNMP v3 agent's engine ID, boots, and time ticks.
engine time ticks	As defined in SNMP v3 specifications, number of seconds elapsed since the last engine boot. When this counter reaches its maximum limit, it is reset to 0 and the engine boots value is incremented by 1.
enterprise scope	A setting for an alarm definition that determines that there will be at most one alarm instance monitoring the entire enterprise. See also <i>alarm scope</i> on page 31.
event	A detected network or system condition. An event can be forwarded to one or more NerveCenters and network management platforms.
generic trap number	Indicates the nature of an SNMP trap detected by a NerveCenter trap mask. There are seven different types of traps, numbered 0 through 6. The first six (0-5) are industry-standard traps and are described in any standard SNMP text. NerveCenter also provides an option for selecting all six traps. Trap number 6 is reserved for enterprise-specific traps and NerveCenter Inform actions. See also <i>specific trap number</i> on page 40.
Hewlett Packard OpenView IT/Operations	A part of the OpenView network management platform that provides network and system management. NerveCenter integrates with IT/Operations by sending OpC Informs and receiving IT/Operations messages, which are detected by OpC masks. IT/Operations requires the SEMSOPCA adapter on its host machine. SEMSOPCA is installed when OVPA platform integration is selected during installation.
Hewlett Packard OpenView Network Node Manager	A part of the OpenView network management platform that provides network discovery, a map, threshold detection, a MIB browser, an event browser, and other functions. NerveCenter integrates with Network Node Manager by retrieving nodes from its OpenView database, sending color changes to the node symbols on the map, and sending events to the event browser. The Network Node Manager host requires NerveCenter's OVPA adapter to communicate with NerveCenter.
HMAC-MD5-96	See <i>MD5</i> on page 35.

HMAC-SHA-96

See *SHA* on page 40.

IBM Tivoli Systems TME 10 Enterprise Console

A network management platform that can be configured to receive NerveCenter Inform messages. The Enterprise Console platform host requires NerveCenter's universal platform adapter to communicate with NerveCenter.

inform message

A message indicating a specified network condition. You can configure alarms to send a NerveCenter Inform when a particular network event is detected. NerveCenter can send the Inform data to one or more NerveCenters and network management platforms. In NerveCenter Administrator, you specify Inform recipients and the types of events to be forwarded (for example, those above a specified severity level). This allows you to manage which platforms handle particular events.

instance

The identity of a particular MIB base object when there are multiples of that base object on a node. For example, if a managed node has four ports, it has four instances of the ifEntry base object, numbered one through four. In NerveCenter, a base object and its instance is called a subobject.

instance scope

A setting for an alarm definition that lets you monitor one or more base objects in an alarm instance. Instance scope is similar to Subobject scope but has the following difference: Instance scope lets you monitor any instance for different base objects.

See also *alarm scope* on page 31.

IP address

The 32-bit host address defined by the Internet Protocol in STD 5, RFC 791. It is usually represented in dotted decimal notation, for example, 192.164.10.0.

managed node

A node that can be targeted by NerveCenter polls. Only managed nodes are polled. SNMP traps, however, are received from nodes regardless of their managed state.

See also *node* on page 37.

Management information base (MIB)

A defined collection of device information that's governed by SNMP. An agent's MIB contains the configuration and status values for the particular type of device. A specific type or class of management information is called a MIB base object.

MIB-II defines the common set of objects for network management of TCP/IP-based intranets. Other enterprise-specific MIBs can be defined to support specific pieces of hardware.

See also *Simple Network Management Protocol (SNMP)* on page 40.

map

The graphic representation of your network and its devices that a network management platform provides.

mask

See *trap mask* on page 41.

MD5

Message authentication/ hashing algorithm as defined in HMAC-MD5-96 specifications, intended for secure message exchanges between any two entities communicating with each other. The algorithm takes a message of arbitrary length and produces a 128-bit message digest. This is one of the two algorithms suggested in SNMP v3 specifications for message authentication. This protocol is supported by NerveCenter.

MIB

See *Management information base (MIB)* on page 35.

MIB base object	<p>An object that is managed by a network manager. Base objects contain attributes that have values. In MIB-II, for example, system, ifEntry, icmp, and ipAddrEntry are all base objects. System contains attributes such as sysDescr and sysUpTime that have associated values.</p> <p>Some base objects, such as system, are simply the MIB group name and contain a single set of attributes. These are zero-instance base objects because they have no separate instances. Other base objects represent one instance out of two or more. IfEntry, for example, is one interface on a device that may contain several interfaces. Connecting the base object name and instance number with a period, as in ifEntry.2, fully qualifies the base object instance that is being referenced.</p>
Micromuse Netcool/OMNibus	A network management platform that can be configured to receive NerveCenter Inform messages. The Netcool platform host requires NerveCenter's universal platform adapter to communicate with NerveCenter.
NCContext	Every communication between two SNMP v3 entities takes place on behalf of a <i>user</i> (a uniquely identified entity in the SNMP v3 management domain) in some <i>context</i> (a uniquely identified entity for access control generally configured on conventional SNMP v3 agents/nodes). Before NerveCenter can poll SNMP v3 agents, the agents must be configured to support a NerveCenter user and NerveCenter context. By default, the user is NCUser and the context is NCContext, though you can change both names in NerveCenter Administrator.
NCUser	Every communication between two SNMP v3 entities takes place on behalf of a <i>user</i> (a uniquely identified entity in the SNMP v3 management domain) in some <i>context</i> (a uniquely identified entity for access control generally configured on conventional SNMP v3 agents/nodes). Before NerveCenter can poll SNMP v3 agents, the agents must be configured to support a NerveCenter user and NerveCenter context. By default, the user is NCUser and the context is NCContext, though you can change both names in NerveCenter Administrator.
NerveCenter Administrator	The NerveCenter software module that enables you to customize and manage NerveCenter operations. Run the NerveCenter Administrator program, ncadmin.exe, from your Start menu program group (Windows) or from the NerveCenter <i>installation/bin</i> directory.
NerveCenter administrator	A system or network administrator who is responsible for configuring and maintaining NerveCenter operations and, optionally, installing NerveCenter.
NerveCenter Client	The NerveCenter software module that enables you to create and manage behavior models as well as monitor and report on network activity. Run the NerveCenter Client program, client.exe, from your Start menu program group (Windows) or from the NerveCenter <i>installation/bin</i> directory.
NerveCenter enterprise trap number	The enterprise-specific MIB object identifier (OID) for NerveCenter is 1.3.6.1.4.1.78. You use this number when creating trap masks to detect Inform messages.
NerveCenter object	A NerveCenter entity that can be part of a behavior model used to manage your network. The following are NerveCenter objects: nodes, property groups and their properties, polls, trap masks, OpC masks, and alarms.

NerveCenter Server

The background engine that handles event correlation and manages the NerveCenter database. Each time you use the NerveCenter Client or Administrator, you must connect to a server. The server can be installed on the same machine as the NerveCenter Client or Administrator applications.

NerveCenter Web Client

The NerveCenter software module that enables you to monitor network activity using Microsoft Internet Explorer 4.0 or Netscape Navigator 4.0. Run the Web Client from your browser by entering the following address:

`http://servername/NerveCenter`

See also *NerveCenter Web Collector* on page 37.

NerveCenter Web Collector

If Web server integration was included with your NerveCenter installation, the NerveCenter Web Collector was installed on your machine. The Web Collector communicates with both the NerveCenter Server and your Web server in order to provide current information about alarms associated with the NerveCenter Server.

network management platform

NerveCenter 3.5 supports several network management platforms, and communicates with each using a platform adapter. The platform adapter must be installed and running on the platform host before a connection can be established with NerveCenter.

Following are the platforms that can both provide node information and receive NerveCenter messages, using NerveCenter's OVPA adapter.

- ◆ Hewlett Packard OpenView Network Node Manager

The following platforms can only receive NerveCenter messages and display those messages in their event or message browser. These use the NerveCenter universal platform adapter:

- ◆ IBM Tivoli Systems TME Enterprise Console
- ◆ Computer Associates Unicenter TNG
- ◆ MicroMuse Netcool/OMNIBus

Finally, Hewlett Packard's OpenView IT/Operations (ITO) uses the SEMSOPCA adapter to send and receive IT/O messages. SEMSOPCA is installed when OVPA platform integration is selected during installation.

Network Node Manager

See *Hewlett Packard OpenView Network Node Manager* on page 34.

NoAuthNoPriv

Security level that does not require message authentication or encryption services for communication between any two SNMP v3 entities. That means communication between the two SNMP v3 entities communicating with this (i.e. NoAuthNoPriv) security level is not secure.

Communication between two SNMP v3 entities takes place on behalf of a *user* (a uniquely identified entity in the SNMP v3 management domain) in some *context* (a uniquely identified entity for access control generally configured on conventional SNMP v3 agents/nodes). As such, NerveCenter still requires the user name and context for polling.

node

Any device on the network that can be managed. Examples of nodes are servers, workstations, printers, hubs, routers, bridges, and gateways. When NerveCenter is integrated with a network management platform, you can configure NerveCenter so its

node list is populated by the platform. In addition, NerveCenter can be configured to add unknown nodes that send it a trap. You can also populate the list by using NerveCenter's Discovery behavior model, or you can add nodes manually.

See also *managed node* on page 35.

node scope

Setting for an alarm definition that determines that each alarm instance monitors a node.

See also *alarm scope* on page 31.

object identifier, object ID, or OID

A unique SNMP name given to each MIB base object, identified by the value of the sysObjectID object in the system group of MIB-II. The name is written as a sequence of integers separated by periods. For example, the sequence 1.3.6.1.2.1.1.1.0 specifies a system description. Each object associated with a vendor or type of equipment also has an object identifier.

You can obtain the object identifier for a node in your network by opening its Node Definition window. Select the Query Node tab and then select the **Get** button.

OID

See *object identifier, object ID, or OID* on page 38.

OpC mask

Similar to a NerveCenter trap mask, an OpC mask is used to detect and filter IT/Operations messages. It detects an incoming message, does a preliminary screening for values of interest, and issues a trigger. The OpC mask also gives you the option of creating a trigger function, a Perl subroutine that NerveCenter uses to do more complex filtering on the message.

OVPA (OpenView Platform Adapter)

A NerveCenter platform adapter process that resides on a Hewlett Packard OpenView Network Node Manager host. The adapter enables communication between the network management platform and NerveCenter.

passwords (SNMP v3)

Before NerveCenter can poll SNMP v3 nodes, you must supply the proper passwords for the NerveCenter user. These passwords must be configured on your SNMP v3 agents as well as in NerveCenter Administrator. You need to provide the following passwords:

- ◆ Authentication: Required for providing authentication services when NerveCenter communicates with the agent using either an AuthNoPriv or AuthPriv security level.
- ◆ Privacy: Required for providing encryption services when NerveCenter communicates with the agent using an AuthPriv security level.

Depending on the security level you provide in NerveCenter Client for the node corresponding to an SNMP v3 agent, NerveCenter will use an appropriate combination of passwords to provide message authentication and message encryption services.

ping

An ICMP echo request sent to a network device that returns an echo reply from the device. ICMP is an error-control protocol that works with the Internet Protocol.

platform adapter

A NerveCenter process that facilitates communication between NerveCenter and a network management platform.

See also *OVPA (OpenView Platform Adapter)* on page 38 and *Universal Platform Adapter (paserver)* on page 42.

poll	A NerveCenter object that monitors the network for conditions of interest by polling SNMP agents on managed nodes for specific MIB data. The poll compares these values to a user-defined poll condition and trigger expression. If the condition is satisfied, the poll generates a trigger that signals one or more alarms. At least one alarm must be enabled and include a transition that can be triggered by the poll's trigger.
poll condition	<p>Defines the condition to be detected on each node that is tracked. When a poll condition is satisfied, the poll generates a trigger that signals one or more alarms.</p> <p>You can create conditions, such as threshold crossings or rates of change for tracked values, using arithmetic and relational operators. You can string together combinations of conditions using logical (Boolean) operators.</p>
poll rate	The interval (in seconds, minutes, or hours) that a poll waits between requests for node MIB data and evaluation of the associated poll condition.
privacy key	The seed used by the privacy protocol to encrypt messages while communicating with an SNMP v3 entity using AuthPriv security level. The privacy key is described in SNMP v3 specifications.
property	A text string that is a member of one or more property groups. Properties fall into two categories: MIB base objects and user-defined strings. Poll and alarm definitions use properties when determining whether a device should be monitored.
property group	<p>A collection of one or more text strings called properties. Property groups allow you to categorize nodes into logical or managerial units. The groups can be based on device type, location, priority, supported MIBs, business function, or any other useful characteristic.</p> <p>Each managed node belongs to a property group and, therefore, is associated with all of the group's properties. These properties are used to restrict which polls and alarms monitor the node. Assigning a node to a property group that contains multiple properties allows the node to be targeted by multiple behavior models.</p>
resync	<p>When NerveCenter is integrated with a network management platform, NerveCenter issues a Resync command upon start-up. This command updates NerveCenter's entire node list with information from the platform. The update contains all nodes within NerveCenter's subnet filters, if applicable, and includes details about each node's parents.</p> <p>You can manually refresh the NerveCenter node list by selecting the Resync command from the client's Server menu. If this command is not available, the connection to the node source is down.</p> <p>See also <i>synchronization</i> on page 41.</p>
scope	See <i>alarm scope</i> on page 31.
security level	<p>The use of message authentication and message encryption services for communication between any two SNMP v3 entities. NerveCenter supports the following security levels while communicating with SNMP v3 agents:</p> <ul style="list-style-type: none"> ◆ NoAuthNoPriv: Neither message authentication nor message encryption services are used for communication between two SNMP v3 entities. As such, the communication is not secure. NerveCenter requires the user name and context for polling.

- ◆ **AuthNoPriv:** Message authentication services are used without message encryption for communication between two SNMP v3 entities. This ensures only authenticity of the messages being exchanged between two SNMP v3 entities. The messages are not encrypted.
- ◆ **AuthPriv:** Both message authentication and message encryption services are used for communication between two SNMP v3 entities. This is the most secure way of communication defined in SNMP v3 specifications at the present time.

SerializeDB

A utility that transfers data between the database being used by NerveCenter and a serialized file. This tool is useful for backing up and restoring database information. Run the SerializeDB program, `serializedb.exe`, from your Start menu (Windows) or from the NerveCenter installation/Bin directory.

See also *DBWizard* on page 33.

severity

See *alarm severity* on page 31.

SHA

A secure hash algorithm specified in the Secure Hash Standard (SHS, FIPS PUB 180). This is one of the two algorithms suggested in SNMP v3 specifications for message authentication. This protocol is supported by NerveCenter.

Simple Network Management Protocol (SNMP)

An industry-standard protocol that defines how network management systems exchange information with their managed nodes. Software processes called SNMP agents reside on each managed device and track defined sets of data. A database of network information, called a management information base (MIB), is associated with both the manager and agent. A manager and agent may exist on the same system.

SNMP messages allow a management application to set and retrieve agents' managed data. SNMP traps allow an agent to send events to a management application.

smart polling

A NerveCenter feature that reduces the network overhead associated with SNMP management. A poll doesn't solicit data from nodes unless an alarm definition uses the poll's trigger. Even then, the poll is issued to a specific node only when the alarm exists in a state that the poll can trigger. For example, if a behavior model correlates high traffic followed by high error rates, a node is not polled for error rates unless it fulfills the high traffic condition first.

SNMP

See *Simple Network Management Protocol (SNMP)* on page 40.

SNMP trap

An asynchronous notification that an SNMP agent can send to a management application. Traps are identified by a generic number (0–6), an enterprise-specific number, and an enterprise name. Related information data is typically bundled with a trap in structured formats called variable bindings. NerveCenter detects and filters SNMP traps with trap masks.

specific trap number

An SNMP trap number associated with an enterprise MIB. When defining trap masks, generic trap numbers (0-6) indicate the nature of the SNMP trap being reported. In addition, each enterprise, or vendor, can have any number of subtrap numbers, called specific trap numbers, within the category 6 generic trap. Each vendor defines its own number of enterprise traps, their associated specific numbers, their functions, and their variable bindings.

See also *generic trap number* on page 34.

state diagram	See <i>Alarm state diagram</i> on page 32.
subobject	A base object and an instance connected by a period. Examples from MIB-II include ifEntry.3, system.0, and ipForwardEntry.2.
subobject scope	<p>A setting for an alarm definition that determines that each alarm instance monitors a subobject. Subobject scope indicates that conditions for each subobject are tracked separately with separate alarm instances. For example, one alarm instance would track conditions on ifEntry.1, another would track conditions on ifEntry.2, and so on.</p> <p>See also <i>alarm scope</i> on page 31.</p>
suppression	<p>NerveCenter provides the following types of suppression as a way to manage unresponsive nodes.</p> <ul style="list-style-type: none"> ◆ Node suppression—When a node fails to respond, it can be suppressed manually or through an alarm action. Normal polling does not occur for the node as long as the node remains suppressed. ◆ Poll suppression—By default, NerveCenter polls are suppressible. This means they do not poll nodes that are suppressed. However, there may be specific polls that you want to occur on a node even when it is suppressed. In such a case, you can set the poll to insuppressible. <p>Suppression affects only polling; SNMP traps are still received from a managed node even though it is suppressed.</p>
synchronization	<p>When NerveCenter is integrated with a network management platform, the platform uses a process called synchronization to update NerveCenter whenever the platform adds, updates, or deletes nodes. This type of update does not include parent information.</p> <p>See also <i>resync</i> on page 39.</p>
time ticks	See <i>engine time ticks</i> on page 34.
transition	See <i>alarm transition</i> on page 32.
trap	See <i>SNMP trap</i> on page 40.
trap mask	A NerveCenter object that captures SNMP traps received from managed nodes and fires a trigger that transitions an alarm instance. A mask can detect a standard SNMP trap identified by one of six generic categories, an enterprise-specific trap, and, with the advanced Trigger Function feature, a trap whose contents match user-specified criteria. At least one alarm must be enabled and include a transition that can be triggered by the mask's trigger.
trigger	<p>A flag sent to one or more alarms to indicate that an event has been detected or a condition satisfied. The trigger transitions an alarm from one state to another. The alarm must be enabled and include a transition that can be triggered by this particular trigger.</p> <p>Triggers are generated by polls, trap masks, OpC masks, and the Fire Trigger or Perl Subroutine alarm actions. You specify a trigger using the FireTrigger() function in a poll condition, Perl subroutine, or trap mask trigger function. NerveCenter generates its own built-in triggers, designed to detect unresponsive nodes.</p>

trigger function

A Perl script that you define for a NerveCenter trap mask or OpC mask. The script is called whenever a detected SNMP trap or OpC message matches the mask's data fields.

The Perl script issues triggers based upon the contents of the trap's variable bindings. When a suitable trap is detected, NerveCenter executes the trigger function, supplying data from the trap's variable bindings. Your script evaluates the variable-binding contents and conditionally fires triggers or assigns property groups.

trigger generator

A NerveCenter object or action that fires a trigger when a certain condition is detected. The trigger is sent to one or more alarms, where it transitions one or more alarm states.

The following are NerveCenter trigger generators: polls, trap masks, OpC masks, Perl subroutines that include the FireTrigger() function, and alarm actions (Send Trap, Fire Trigger). NerveCenter generates its own built-in triggers when it detects unresponsive nodes.

Universal Platform Adapter (paserver)

A NerveCenter platform adapter process that resides on IBM Tivoli Systems TME Enterprise Console, Computer Associates Unicenter TNG, or Micromuse Netcool/OMNIbus. The universal platform adapter enables NerveCenter to send Inform messages to those platforms.

user rights

Those with user rights belong to the NerveCenter Users (Windows) or ncusers (UNIX) group. They perform such tasks as monitoring NerveCenter alarms, resetting alarms, and generating reports. They cannot modify the NerveCenter database, for example, by making changes to nodes, property groups, polls, masks, or alarms.

variable bindings

An array of related information that accompanies an SNMP message. The variable bindings are typically defined in standard or vendor-specific MIB definitions (.ASN1 files). The format of each variable binding is defined by SNMP. NerveCenter polls and trap masks obtain variable bindings from the nodes they monitor.

Zero-instance base object

A MIB base object that has only one instance and, therefore, contains a single set of attributes. For example, system is a zero-instance base object because it contains attributes (like sysLocation) that have only one associated value per node. In contrast, ifEntry is a base object with multiple instances. If a node had four ports, the node would have four instances of the ifEntry base object, numbered one through four, and each instance has associated attributes.

Index

A

Action Router	7
alarm	4, 7
finite state machine	7
IfUpDownStatus example	7
alarm actions	6
Alarm Summary window	10
ASN1 file	14

B

behavior model	3, 4
objects	4
operation	5

C

CLI	10
Command line interface	10
compiled MIB files	14
Computer Associates Unicenter TNG	27
Correlation	3

D

database	9
database management	12
DBWizard	12, 13
nervecenter.ncdb	12
nervecenter.node	12
SerializeDB	12, 13
UNIX	12
Windows	13
DBWizard	12, 13
discovery	11

E

events	3
--------	---

F

finite state machine	7
----------------------	---

I

ICMP	2
importing	16
ImportUtil	16
inform	6

Internet Control Message Protocol	2
IPSweep	
discovery	
IPSweep	16

M

management information base	14
MIB definitions	9
MIB files	14
ASN1 file	14
MibComp compiler	15
MibComp.txt	15
NerveCtr.mib	14
MibComp	15
MibComp.txt	15
Micromuse Netcool/OMNIbus	28
Multiple NerveCenters	29

N

Ncapp	19
NerveCenter	
overview	2
NerveCenter Administrator	10
NerveCenter Client	10
Alarm Summary window	10
NerveCenter components	9
NerveCenter integrated with a network management platform	18
NerveCenter Server	9
NerveCenter Trap service	20
NerveCenter utilities	15
ImportUtil	16
IPSweep	16
Trapgen	15
NerveCenter Web Client	17
nervecenter.ncdb	12
nervecenter.node	12
NerveCtr.mib	14
network management platform integration	18, 19, 25
Computer Associates Unicenter TNG	27
Micromuse Netcool/OMNIbus	28
OpenView IT/Operations	24
OpenView Network Node Manager integration	19
platform adapter	18
Tivoli TME Enterprise Console	26
node	4

O

OpenView	19
OpenView IT/Operations integration	24
OpenView Network Node Manager	19
Ncapp	19
NerveCenter Trap service	20
Overview of NerveCenter	2
OVPA	18, 19

P

paserver	25
Perl	6
Perl subroutine	6
platform adapter	18
poll	4
property	4, 5
property group	4, 5

S

SEMSOPCA	18, 24
SerializeDB	12, 13
severity	10
Simple Network Management Protocol	2
SNMP	2, 3

T

Tivoli TME Enterprise Console	26
trap mask	4
trap service, NerveCenter	20
Trapgen	15
trigger	4

U

universal platform adapter	18
Universal platform adapter integration	25
UNIX database	12
Unix database	
nervecenter.ncdb	12
nervecenter.node	12

W

Web Client	17
Web Collector	17
Web-based monitoring	17
What NerveCenter does	2
Windows database	13
DBWizard	12, 13