HP Network Node Manager

A Guide to Scalability and Distribution

Windows, HP-UX, Solaris, Linux operating systems

Manufacturing Part Number : n/a
July, 2004

Legal Notices

Warranty.

Hewlett-Packard 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. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

A copy of the specific warranty terms applicable to your Hewlett-Packard product can be obtained from your local Sales and Service Office.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause in DFARS 252.227-7013.

Hewlett-Packard Company
United States of America

Rights for non-DOD U.S. Government Departments and Agencies are as set forth in FAR 52.227-19(c)(1,2).

Copyright Notices.

Trademark Notices.

Linux is a U.S. registered trademark of Linus Torvalds.

Microsoft® is a U.S. registered trademark of Microsoft Corporation.

Oracle® is a registered U.S. trademark of Oracle Corporation, Redwood City, California.

Support

1. Overview of Scalability and Distribution in NNM

2. The Scalable Architecture of Network Node Manager

Support

Please visit the HP OpenView web site at:

`http://openview.hp.com/`

There you will find contact information and details about the products, services, and support that HP OpenView offers.

You can go directly to the HP OpenView support web site at:

`http://support.openview.hp.com/`

The support site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

1 Overview of Scalability and Distribution in NNM

This chapter contains an initial look at the features in HP OpenView Network Node Manager (NNM) that make it a scalable solution for network management.

This document is not an introduction to NNM. Readers are assumed to be conversant with NNM's key features for network monitoring, data collection, event handling, and so on. Readers who are looking for that information should turn to *Managing Your Network with HP OpenView Network Node Manager*.

This chapter introduces the ideas and features for scalability and distribution in NNM; many details are omitted in the interest of making the broad concepts clear.

Later chapters contain many more details about the architecture of the NNM software, guidelines on how to deploy NNM effectively in your organization, and specific configuration procedures for using the scaling and distribution features of NNM.

Scalability and Distribution in NNM

You can configure NNM to perform well for a wide range of network sizes and degrees of complexity. This attribute is commonly referred to as the **scalability** of NNM. Scalability gives NNM the potential to handle almost any size network, in both local and wide-area environments.

One approach to making a product like NNM scalable, and an important feature of NNM, is to distribute the network management workload to multiple, usually remote, systems. NNM makes **distribution** possible, so you can monitor and manage much larger networks while minimizing the management resources required.

In summary, the scalability and distribution of NNM optimizes the use of two key resources:

- First, NNM includes features that help you use the *system resources* at the management station very efficiently.
- Second, other features of NNM, especially distribution, reduce the amount of management traffic over your *network*, most importantly, over congested or expensive links.

Benefits of Scalability

Having a scalable network management platform, implemented in part through distribution, provides you with many benefits in your application of NNM as a network management solution.

First, scalability gives your network management solution the ability to evolve with your organization. Networks are dynamic entities that reflect the changing structures and needs of the organizations they serve. As your network changes and grows, the scalable features of NNM make it easier to adapt your network management to handle new networking landscapes.

Second, a distributed network management solution can dramatically reduce the traffic overhead that network management can impose on congested or expensive network links. Most large (enterprise) networks include serial or wide-area (WAN) links. Such links are both relatively slow and expensive. Without a distributed network management solution, it is simply impractical to monitor tens of thousands of nodes from a single workstation; the networking load emanating from the

management station becomes a bottle neck. Distribution minimizes the amount of network management traffic on these links, freeing bandwidth for business purposes.

Third, a distributed solution makes it possible to share network management responsibility and resource requirements among multiple sites. By spreading the resources and responsibilities out, you reduce the risk of losing all network management in the event of an unexpected shutdown of your management station. Distribution also makes it easier to perform maintenance shutdowns without losing your overall network management.

Fourth, a scalable network management solution, sensibly deployed and properly configured, results in faster response times and better performance. You spend less time waiting for data and more time acting on it. Furthermore, it improves network reliability by allowing local management actions even when contact is lost with a remote central management site.

Finally, deploying a scalable network management platform means everyone in your organization can share a single customizable tool. This naturally leads to a common vocabulary, common experiences, sharable expertise, and the ability to more easily articulate and implement standard processes and procedures in your organization.

All together, NNM's scalability and distribution gives you the power to manage the large and geographically dispersed enterprise networks that are critical to key business or organizational processes. You can use common applications at every level to create a more integrated, consistent solution to the complex problems in these environments.

What Makes NNM a Scalable Solution for Network Management?

The idea of "scalability" is abstract, but HP OpenView Network Node Manager has several broad categories of features that make scalability a concrete reality.

NOTE There are important differences between the scalability features delivered with the NNM Starter Edition and Advanced Edition. The NNM Advanced Edition can function as a management station, collection station, or both. The NNM Starter Edition can only function as a collection station.

Distributed Internet Discovery and Monitoring

Distributed Internet Discovery and Monitoring lets remote collection stations monitor their part of the network and inform the central management station about it. This reduces network traffic, and also reduces the load on the central station.

Management Consoles

Management consoles off-load display processing from the management station to a display station. With management consoles, you can have more operators monitoring and managing your network without loss of performance; these operators can handle larger environments.

Web Browser Interface

Accessing network management functions from your web browser lets you monitor your network from any location with a web browser installed, even if it is not a management console. See *Managing Your Network with HP OpenView Network Node Manager* for more information.

Large-Map Viewing Support

NNM gives you two essential features for examining large maps:

- The panner, which makes it easy to navigate submaps with hundreds of symbols.
- The quick navigator, which gives you instant access to your most critical or frequently visited submaps.

On-Demand Submaps

The on-demand submap feature provides a very significant boost in the performance of NNM, allowing it to manage many more objects (up to ten times) than with persistent submaps. Enabling the on-demand submap feature lets your management station use less memory to perform the same tasks.

Filtering

Filters let you reduce the amount of extraneous data being handled by the system, or presented to the operator. This provides improved performance, and reduces visual clutter for the operator.

Event Forwarding

You can configure your remote stations so that events from managed nodes, or from non-NNM applications on the collection station, get forwarded to interested management stations. This can be set up to occur whether or not the event triggers local actions, or has already been correlated.

Important events generated internally to NNM, such as topology or status change events, are automatically forwarded from a collection station to the interested management stations, without additional configuration steps.

Distributed Threshold Monitoring

Distributed threshold monitoring reduces management traffic on the network by allowing remote stations, instead of the management station, to collect SNMP trend data and trigger threshold events based on that data.

Interface Managed-state Automation

NNM lets you automatically manage or unmanage interfaces based on filters. Interface managed-state automation makes it easier to manage large environments, where frequently many discovered interfaces do not require active status polling.

High Availability and Fault Tolerance

NNM stations can be run on HP Service Guard clusters to provide continuous availability. Also, if a collection station fails, you can configure its status polling duties to transfer over to a management station. You can also perform backup of your network management data without losing incoming information during that time.

Your practical limits in using all the above depend on the size of your network environment, your management strategy, and the resources available on the management and collection stations. For the most current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*, available at `http://openview.hp.com/`.

NOTE The upper limits of operators and managed devices depends very heavily on the level of monitoring and data collection you require. In general, increasing the polling intervals on noncritical nodes increases the number of nodes that can be handled.

Network, System, and Operator Efficiency via Filtering

For NNM to manage very large environments, all resources (network, system, and human) must be used as efficiently as possible. To maximize the efficient use of these resources, NNM employs a strategy of data reduction via filters.

A **filter** is essentially just a way to remove unneeded data from consideration for further processing. The result of less data in the system is higher performance for NNM, and improved usability for the operators.

For example, you may have an operator whose only concern is the routers and hubs in your environment. For that operator, any other objects on his map are merely clutter. With NNM, you can apply a filter to the map, so that only routers and hubs appear.

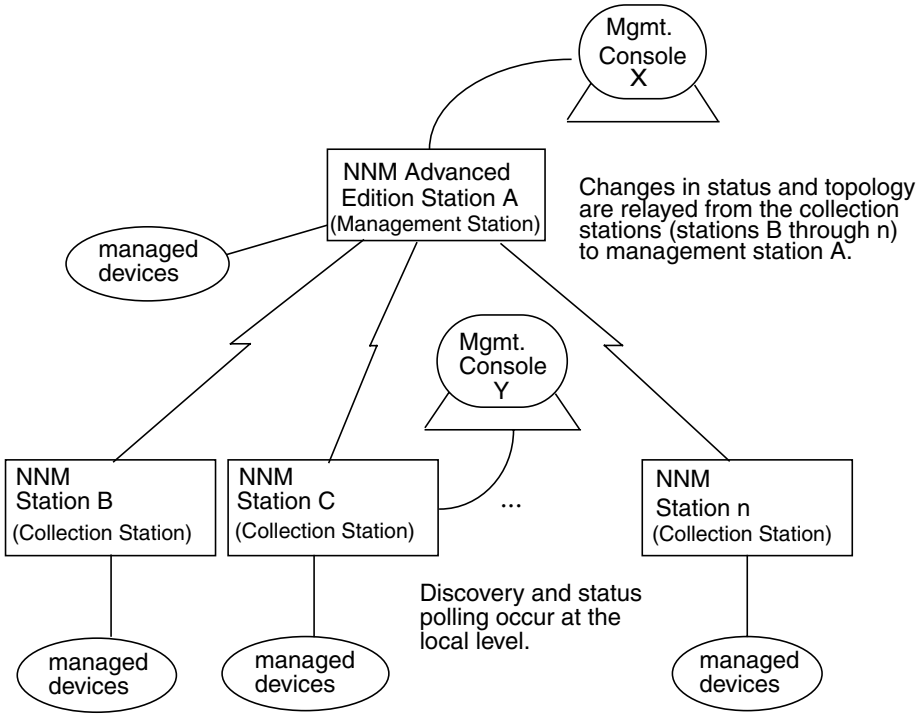
See “Filters” on page 37 for more details about filtering in NNM.

Distributed Internet Discovery and Monitoring

A network management system can consume significant resources while performing discovery, topology monitoring, and status polling. One feature of HP OpenView Network Node Manager is the ability to move much of this load from the management station to one or more remote machines.

This capability is called **Distributed Internet Discovery and Monitoring**, and is illustrated in Figure 1-1. The device count for this model assumes one interface per device, on average. Note that the management console at Station C indicates that it also serves as a management station for the objects it monitors; this is the usual configuration of NNM.

Figure 1-1 Simplified Distribution Model for NNM



To use distributed internet discovery and monitoring, you designate one or more remote machines as **collection station**s. Each collection station takes over the task of monitoring some portion of the network, while informing all interested management stations of any changes in the status or topology of the network it monitors.

Capacity Planning

Because distributed management partitions the work of discovery and status polling, it also dramatically reduces traffic on the network, most importantly on expensive or congested links.

When considering performance and configuration issues, you must distinguish between nodes and objects. There is a one to many relationship between the number of nodes and the number of objects in the object database. Even though a network model is typically described in terms of the number of nodes the application manages, it is the number and type of managed objects that is the most important parameter when predicting how the application will scale and perform.

Determining the exact number of managed objects is especially critical when predicting, for example, the size of a database, the load on `netmon`'s polling cycle, or the load imposed on a management station when synchronizing with a collection station. You'll want to determine the appropriate node to object multiplier for your environment.

Not all network environments conform to the typical environment described above. In some environments, the ratio of objects to nodes is higher because enterprise routers tend to have more than two physical interfaces. For example a network center responsible for monitoring only backbone routers would have a higher ratio. In some cases nodes may have additional virtual interfaces. Secondary addresses, for example, increase the number of objects associated with that particular managed node.

There is no concrete limit to the number of objects or devices you can manage, or to the management stations you can deploy. Your practical limits depend on the size of your network environment, your management strategy, and the resources available on the management and collection stations. For the most current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*, available at `http://openview.hp.com/`.

About Management Stations and Collection Stations

In the previous section (including Figure 1-1), two terms were introduced:

- Management station
- Collection station

It is important to recognize that these labels state the *roles* of the machines involved, and do not refer to different software on them. In this vein, you should note that a single NNM Advanced Edition station can perform either role, or both at once. Regardless of the role it is playing (as a collection or management station, or both), every NNM Advanced Edition station has the full capabilities of NNM[1]. HP Service Guard clusters for UNIX® operating systems may be used for any type of management station to ensure constant availability.

The role of a management station is to make the network management functionality available to users, either directly or via one or more management consoles.[2]

The role of a collection station is to be a collection point in the overall network management system. A collection station typically performs topology and IP status monitoring, threshold data collection, local event correlation, and event forwarding or handling on behalf of one or more management stations.

When changes occur in topology or status, or when a configured threshold is crossed, the necessary information is forwarded from the collection station to any interested management stations, so that the end-user at a management console is informed. The set of objects for which topology and status information is forwarded can be controlled via filtering.[3]

1. Only the NNM Advanced Edition can receive information from collection stations.
2. See “Management Consoles” on page 24.
3. See “Use of Scalability Features in NNM Deployment” on page 90 for introductory information, and “Filters” on page 37 for more details about filtering.

NNM Advanced Edition management and collection stations alike have all components of the HP OpenView NNM software running on them. The only difference is that a collection station has been designated and configured to serve one role, and a management station has been designated and configured to serve another.

Stated another way, every NNM Advanced Edition station (regardless of operating system or license limit) is inherently a management station, but it must be configured to use the management resources of a remote collection station. Likewise, every NNM Advanced Edition station is *also* inherently a collection station, but it must be configured to provide its data to a remote management station.

In NNM Advanced Edition, a computer acting as a collection station is generally also a management station in its own right, with local users performing network monitoring and management, either independently or in cooperation with the users at a remote management station.

This is illustrated in Figure 1-1. In that illustration, the operator at Management Console Y can see only the nodes directly managed by NNM Station C. On the other hand, the operator at Management Console X can (depending on how the collection stations are configured) see all the objects in the entire scope of management for NNM Station A.

NNM Station View

The NNM Station view is useful in a distributed management environment. It shows you a graphical representation of the collection stations and management stations in your topology. You can use this view to troubleshoot communications between collection stations and management stations. See “Identifying Collection Station Communication Problems” on page 146 for detailed information.

Discovering and Distributing Extended Topology Information

Numerous dynamic views are available from NNM Advanced Edition. These views present a graphical (or tabular) representation of your network infrastructure.

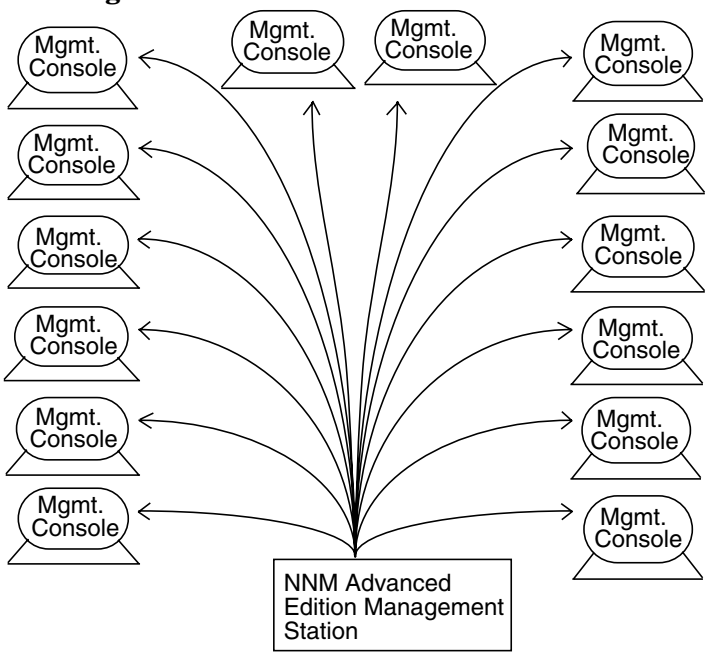
NNM Advanced Edition’s Extended Topology functionality only discovers information from locally managed nodes and does not pass Extended Topology information from the collection station to the management station. To open Dynamic Views that include information from the extended topology, point your web browsers to an object’s primary collection station.

Management Consoles

One of the main problems associated with managing large networks is providing access to enough operators to handle all the objects in the map.

With HP OpenView NNM Advanced Edition, you can have many operators simultaneously monitoring your network, as shown in Figure 1-2, operating from the same topology database and sharing common discovery and monitoring processes at the management station. Operators can share one map, or have individual maps. The limit on how many again depends on the size of your environment and the resources available to the management solution. For the most current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*, available at `http://openview.hp.com/`.

Figure 1-2 Multiple Management Consoles



Consoles run on UNIX[4] or Windows[5] operating systems work with management stations on a UNIX system. These management consoles communicate with the UNIX system management station via NFS and socket connections on a LAN link. Management stations on the Windows operating system only support consoles run on Windows, not consoles run on UNIX. This is because a management station on a Windows operating system uses Windows file sharing, not NFS.

Web browsers using Network Presenter may reside on any system, including the same system as a management console. However, the web server must reside on a full management station, not a console.

4. In this manual, the term UNIX, when used without other qualifications, refers to the supported UNIX operating systems, as described in the Release Notes.
5. in this manual, the term Windows, when used without other qualifications, refers to the supported Windows operating systems, as described in the Release Notes.

On-Demand Submaps

One of the potential problems with having maps that contain thousands of nodes involves the use of available memory.

The `ipmap` application[6] typically creates a hierarchy of several levels of submaps (root, Internet, segment, node). An interesting and important point is that many of these submaps are rarely, if ever, visited by an operator.

Imagine if NNM kept every submap in the map in memory at all times. In a large map, with literally thousands of submaps, this approach would create a substantial requirement for memory.

However, NNM makes it possible to specify that only *some* of the submaps be kept persistently in memory; any *other* submap is created if and only if a user requests it. This means that for a management station with a given amount of memory, you can manage maps with many more submaps containing many more objects.

The main benefit of the on-demand submap feature is a dramatic improvement in performance, especially during synchronization. Meanwhile, the features and functionality of NNM are essentially identical whether or not on-demand submaps are enabled. Whenever you double-click on an explodable object in `ipmap`, the submap opens promptly (though submaps with many objects may take several seconds to open). Furthermore, all features of NNM, including “find” and IP status propagation, work as usual.

Some applications that were created before this feature was part of NNM may not be compatible with the on-demand submap feature. They may expect certain objects to be available in memory at all times. In this case, you can use **persistence filtering** to make those objects persistent.[7] Persistence filtering provides per-object exceptions to the on-demand feature, so that certain objects and their submaps are always present in memory even if on-demand submaps are enabled.

6. See *Managing Your Network with HP OpenView Network Node Manager* for information about this and other key NNM processes.
7. See “Persistence Filtering” on page 68 for details.

Distributed Threshold Monitoring and Event Forwarding

In an NNM distributed environment, it is necessary that operators at the management console be aware of critical events wherever they occur. This is commonly called “management by exception.” For this reason, NNM has the ability to forward events, typically from collection stations to management stations.

Using the graphical `Event Configuration` window, you can configure any event, on a case-by-case basis, to be forwarded to any or all of the following:

- All managers currently using the NNM station as a collection station.
- Particular hosts, identified by hostname or IP address.
- Hosts listed (by hostname or IP address) in a specific file.

All configurable events, including threshold events, can be configured for forwarding. Internal NNM events are automatically forwarded to management stations as needed.

When defining new events, we recommend that you use multiple varbinds with individual pieces of data rather than using a full string. This allows the receiving machine to specify the formatting. Refer to the reference page in NNM online help (or the UNIX system manpage) for *trapd.conf* more information.

General Events

You can forward any or all events from collection stations to management stations. This is most useful when a collection station receives little or no local attention, or when you have applications that issue events which you want to have forwarded to the management station.

For example, suppose you have a collection station that monitors an agent that generates an event called `FuseIsLit`. At the collection station, you can configure that event to be forwarded to the management station, and thereby alert the operator that the device is not functioning properly.

Threshold Events

Threshold events give you a way to be notified of traffic patterns that are outside normal expectations. For example, you could set a threshold to monitor disk utilization on a server. If the amount of free disk falls below the threshold you set, NNM can notify an operator, or take other predetermined actions (such as mailing a message). Threshold events help operators detect and isolate problems as they arise, before users experience difficulty.

In the context of a distributed network management solution, the ability to forward threshold events means that the management station does not itself have to perform data-collection on all the objects that need thresholds monitored. That effort, and the associated network traffic, is delegated to the various collection stations that the manager uses.

Trend Data Collection in a Distributed NNM Solution

Some users may be interested in having trend data collected at remote locations and forwarded back to a central point, such as an NNM Advanced Edition management station, for trend analysis.

Distributed data collection doesn't reduce the load on a management station if the data is sent back to the management station in real time. Therefore, in NNM the remote station stores any trend data in its own database. Data can then be transferred to a relational database if that is available.[8]

NNM does not provide a way to automatically synchronize data back to the management station. It does, however, offer some basic tools that you can use to create custom solutions. For example, you can transfer NNM's data files from the collection stations to the management station (via FTP or a similar utility) during off-peak periods. Then you can use NNM's data-merging tool (see the *ovcoltosql* reference page in NNM online help (or the UNIX system manpage)) to merge the data into an SQL database. Then you can use database tools to create reports on the data.

8. See *Reporting and Data Analysis with HP OpenView Network Node Manager* for information on supported database.

Large-Map Viewing Support

When a submap contains hundreds of nodes, the symbols and connections get smaller, and it becomes difficult to make sense of the graphical picture. Also, changes in status are not as noticeable.

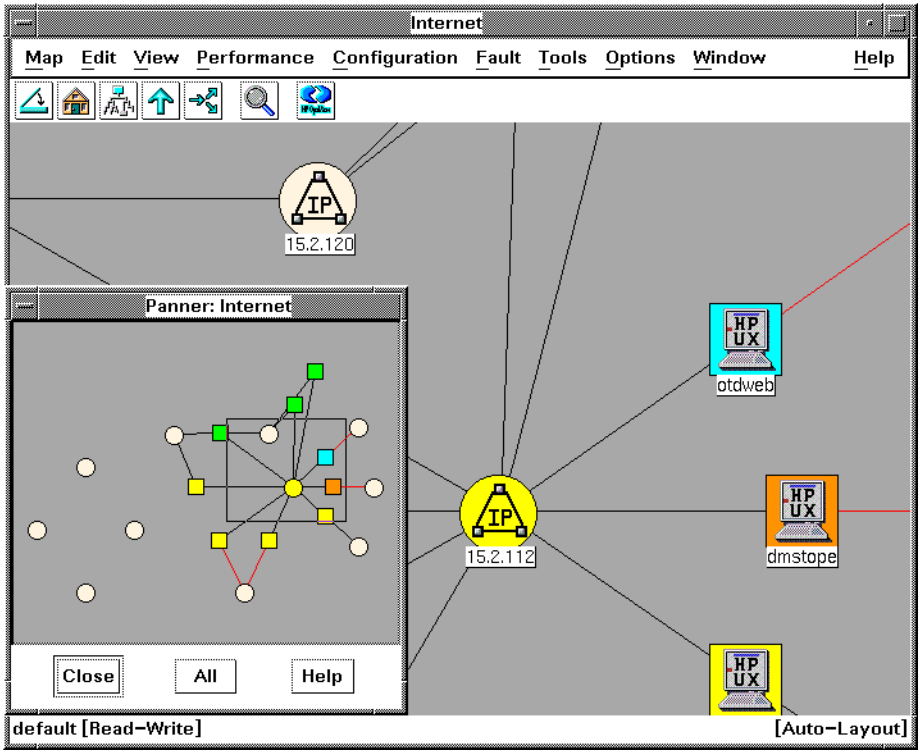
HP OpenView Network Node Manager has two features that make it easy to use maps that contain hundreds, even thousands, of nodes:

- The Panner
- The Quick Navigator

The Panner

The panner, shown in Figure 1-3, provides a way to zoom in on regions of a submap, and drag the zoomed region around. See *Managing Your Network with HP OpenView Network Node Manager*, or NNM's online help, for details.

Figure 1-3 The Panner



The Quick Navigator

The Quick Navigator is a special window that you can customize easily for quick access to your most frequently visited submaps. Your tool bar has a Quick Navigator button, which is described in the online help.

Each operator can customize their own Quick Navigator window by following the simple directions in the online help.

2 The Scalable Architecture of Network Node Manager

This chapter contains more detailed information about the distribution and scalability features of HP OpenView Network Node Manager (NNM). It includes diagrams and descriptions of the major elements of the distributed, scalable NNM architecture, and shows how data moves through the system.

NOTE

The information in this chapter is not introductory; readers are assumed to have read the previous chapter. Readers are also assumed to be conversant with NNM's key features for network monitoring, data collection, event handling, and so on. This includes a functional understanding of some key services (background processes on UNIX systems) in non-distributed NNM, including `ovw`, `ipmap`, `snmpCollect`, `ovtopmd`, and `netmon`. See *Managing Your Network with HP OpenView Network Node Manager* for details about the key services in Network Node Manager.

Later chapters contain guidelines on how to deploy NNM effectively in your organization, and specific configuration procedures to use with the scaling and distribution features of NNM.

The Distribution Model for Network Node Manager

Figure 2-1 illustrates the key components of the distribution scheme implemented by NNM. This diagram is fairly complex, and much of the remainder of this chapter is devoted to explaining many of the concepts it illustrates.

The diagram shows several components that can, and often do, run on different systems. These include full NNM Advanced Edition management stations, NNM Advanced Edition 250 management stations, NNM Advanced Edition and NNM Starter Edition stations in the role of collection stations, and management consoles. These components can be run in any combination of supported operating systems (except for management consoles: Windows operating system servers can only serve Windows operating system clients).

When viewing Figure 2-1 on page 35, remember that the NNM Advanced Edition can function as a management station, collection station, or both. The NNM Starter Edition can only function as a collection station.

NOTE The Windows operating system workstation provides status and topology information on IP and IPX devices. The IPX information exported by the Windows operating system collection station is passed to the UNIX system management station, thus providing a graphical representation of the IPX network topology and status of the individual devices.

In Figure 2-1, stations with no local management console are labeled as collection stations; stations that do have a local management console are labeled as management stations.

NOTE The labeling of management and collection stations in Figure 2-1 is for illustrative purposes only; most (if not all) collection stations also serve as management stations. See “About Management Stations and Collection Stations” on page 22.

Again, whether a particular system is a management station or collection station may be a matter of whose point of view one has at the moment. From the point of view of an operator at Management Console 1, it appears that NNM Station C is a collection station (just like NNM Station D). But from the point of view of an operator at Management Console 4, NNM Station C is the management station he is using.

Each management station can support approximately 15 management consoles via a LAN connection. The management consoles run the per-operator services, freeing up resources on the management station. A later section of this chapter covers this in more detail.[1] As shown in Figure 2-1 at NNM station A, it is still possible to run the per-operator services on the management station, routing the display to an X terminal, or, for NNM on a Windows operating system, a Windows operating system console connected to a Windows operating system collection station.[2]

The number of devices that a full NNM management station can manage in a distributed configuration varies with several different factors:

- The management station's resources (CPU, memory, and disk storage)
- Network performance
- Status and discovery polling intervals
- The level of data collection and threshold monitoring you have configured
- The number of collection stations being managed
- The number of nodes being managed by each collection station

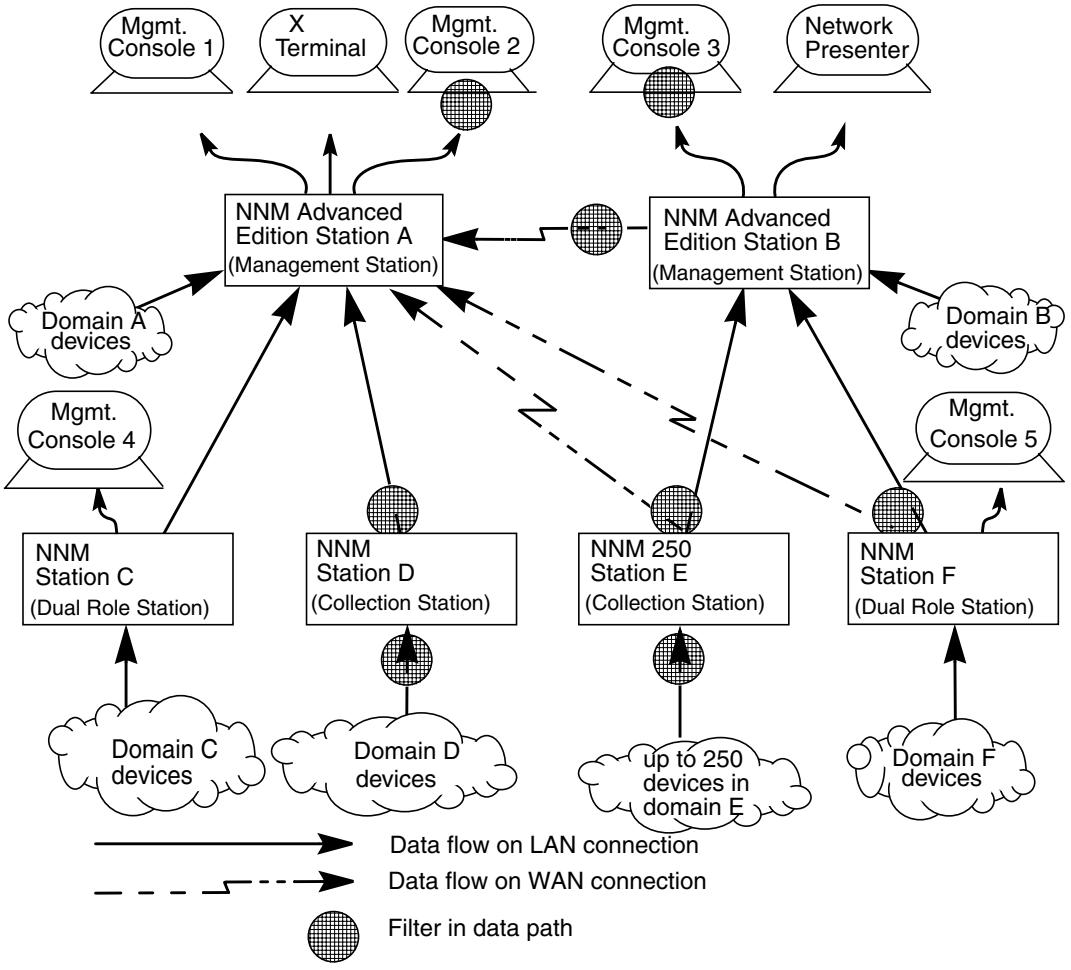
For more information on selecting and sizing the correct computer system for your NNM management station, see the latest version of the *NNM Performance and Configuration Guide*.

To attain your desired IP monitoring response times, you need to maximize the management station resources and set polling intervals to the maximums you can tolerate.

1. See "Management Consoles" on page 70
2. See also "Management Consoles and X-Terminals" on page 73

The only difference between an NNM Advanced Edition management station and an NNM Advanced Edition 250 management station is that the NNM Advanced Edition 250 management stations can manage only up to 250 nodes per license.

Figure 2-1 The Distribution Model for Network Node Manager



With NNM, it is also possible to obtain management information from one or more NNM stations acting as collection stations. In Figure 2-1, most of the NNM stations (Stations B, C, D, E, and F) are operating in the role of collection stations.

Stations C and F illustrate how an NNM station can have the role of a management station (possibly making use of collection stations if it is an NNM Advanced Edition product), a collection station (serving one or more management stations), or both roles at once.[3]

It is possible for a collection station to serve more than one management station; this is illustrated by NNM 250 Station E in Figure 2-1, and also by NNM Station F.

Network Connections between Distributed Components

The connection between a management station and a collection station can be either WAN or LAN-based. WAN connections to collection stations are common. When part of the network you need to manage is remote (that is, accessible via WAN rather than LAN connections), you should consider setting up a collection station at the remote site; distributing NNM will minimize traffic on the WAN connection while providing full network management functionality.

In contrast, it is important to note that a management console must have a LAN connection to its management station. This is because several NNM components (including the event system, topology manager, and HP OpenView Windows database) require high speed communication between the management station and the management console. This performance has been achieved on UNIX systems by using NFS to implement management consoles, and this is the reason management consoles are not supported across WAN links. NNM on Windows operating systems use Windows operating system file sharing for remote access to the files (for the case of a Windows operating system console to a Windows operating system management station), and, while WAN links are supported, they are slow.

In Figure 2-1, the LAN and WAN links between management and collection stations were arbitrarily chosen for illustrative purposes. In contrast, the links from management consoles to management stations are necessarily LAN connections.

3. See “About Management Stations and Collection Stations” on page 22.

Filters

The function of most filters is to select network management data that is of sufficient value to be kept, and to eliminate other data.[4] Data that is useful at one point (say, a site management station) can be filtered out before it gets sent to another (say, regional management station) where it is unneeded.

NNM uses two basic types of filters:

- A **data-stream filter** acts as a sieve through which objects flow. This type of filter contains criteria that permit some objects to proceed, and blocks other objects. Downstream from a data-stream filter, the data stream includes only those objects that the filter passed; other objects are blocked, and thus eliminated from the data at the point of the filter.

  Note that once an object has passed a data-stream filter, later changes to the filter have no effect on it. It remains in the data stream.

- A **set-defining filter** is applied to a pool of candidate objects, and determines which objects belong in the final set based on the criteria contained in the filter. Before a new object can be added to the set, it is tested by the set-defining filter to determine if it is a valid member or not.

  If a set-defining filter is changed, all candidate objects are reevaluated to determine which ones belong in the new set and which do not. Then the set is reconstituted out of valid members.

The difference may seem abstract at first. As an analogy, you can compare a data-stream filter to a wire-mesh sieve, of the kind used to sort stones. Once a large stone has passed through the sieve, it is in the sorted pile beneath regardless of whether you later switch to a finer mesh in the sieve.

Imagine, on the other hand, that your sieve behaved like a set-defining filter. In that case, changing to a finer mesh in the sieve would cause all the source material to be refiltered, yielding a sorted pile of uniformly smaller stones.

4. See “Persistence Filtering” on page 68 for an important exception.

In either case, remember that a filter specifies or describes the objects you want it to pass, *not* the objects you want it to reject. When the filter is applied, it passes the selected objects through, eliminating the rest from the data stream. As a user, you can apply filters only to object data; however, applications can filter events also.[5]

Filtering is based on the information contained in a subset of the fields of the object database (`ovwdb`).[6] These fields are candidates to be all or part of the criteria by which objects are assessed in a filter.

Filters can eliminate unnecessary data at different points in the system:

- Discovery filtering at any NNM station.
- Exported topology filtering at the collection station.
- Map filtering at the management station.
- Failover filtering on the management station.

Table 2-1, Comparing Types of Filters, compares the types of filters. See “Filter File Example” on page 191 for more information.

Table 2-1 Comparing Types of Filters

Filter Type	Apply on Incoming Data	Reevaluate If Filter Changes	Purpose	Applied So Data Is	Reference
Discovery	X		Limit scope of objects added to database.	Excluded	page 40
Topology	X	X	Limit information forwarded from collection station to management station.	Excluded	page 41

5. Interested application developers should see the *HP OpenView Integration Series: SNMP Developer’s Guide*.
6. See “Filterable Objects and Attributes” on page 183 for specifics.

Table 2-1 Comparing Types of Filters (Continued)

Filter Type	Apply on Incoming Data	Reevaluate If Filter Changes	Purpose	Applied So Data Is	Reference
Map	X	X	Show only items of interest to operator on map.	Excluded	page 42
Persistence	X	X	Disable on-demand submaps for third party applications.	Included	page 68
Failover	X	X	Limit nodes polled by management station when collection station fails.	Included	page 52
Important Nodes	X	X	List unreachable important nodes as primary alarms in the Alarm Browser rather than children of a connector down event.	Included	*Managing Your Network with HP OpenView Network Node Manager*
DHCP Range	X		Identify DHCP (shared or floating) IP addresses	Included	*Managing Your Network with HP OpenView Network Node Manager*

Discovery Filtering

Discovery filters specify which devices an NNM station is actively discovering and monitoring. The purpose of discovery filtering is to reduce the discovery and monitoring effort of the station. Different stations have independent (though possibly identical) discovery filters.

Discovery filtering limits the scope of objects that `netmon` adds to the collection station topology database. To unmanage objects and limit the set of nodes that are polled at all, refer to *Managing Your Network with HP OpenView Network Node Manager*. The filter may be set to pass, for example:[6]

- Gateways.
- Bridges, routers, and hubs.
- All devices.
- Nodes based on their `sysObjectID`.
- Objects inside or outside of a particular range of IP addresses.

By default, Segments and Networks pass the discovery filter.

Discovery filtering is achieved by configuring the `netmon` service[7]; the filter is then applied to all newly discovered objects. Objects that are rejected by the discovery filter never appear in any topology or object database.

Discovery filters are data-stream filters; changes to a discovery filter affect new data *only*. All objects that previously passed the filter remain in the data stream, regardless of whether they would currently pass or not, and polling is still done on all previously discovered objects whether or not they would now pass the filter. You can, however, use the `ovtopofix` command to change the set of previously discovered objects.[8]

7. See “Configuring a Discovery Filter” on page 129, for details, *netmon* and *xnmpolling* reference pages in NNM online help (or the manpages on UNIX systems), the NNM online help, and “Distributed Services Architecture” on page 48.
8. See the *ovtopofix* reference page in NNM online help (or the UNIX system manpage) for details.

Implement any discovery filtering on an NNM station before you begin using it as a collection station; this will improve overall performance, by reducing the amount of synchronization effort.

Topology Filtering

Topology filters specify which topology data gets forwarded to a management station. The result is less management traffic on the network and lower data storage requirements at the management station. By default, the topology filter does not pass Networks and Segments.

A topology filter at a collection station defines the subset of topology data that management stations can see. The idea is to have the topology filter at the collection station pass information about only those objects in which the manager is interested. Data about objects outside the manager's interest doesn't clog up the link to the management station.

Objects rejected by a topology filter remain in the collection station's topology database. They are not, however, provided to a higher-level management station. Each collection station has only one topology filter in place. Note that this means the topology filter of a collection station must meet the needs of all management stations.Through topology filtering, you can choose to provide the following kinds of topology data to interested managers (many other choices exist):[9]

- All topology data, by using no filter at all.
- Nodes with (or specifically without) a certain `sysObjectID`.
- Objects inside or outside of a particular range of IP addresses.
- Nodes from a particular vendor.
- Objects with particular physical address.
- All token ring segments.

9. See "Filterable Objects and Attributes" on page 183 for specifics.

Topology filtering is achieved by configuring the `ovtopmd` service.[10] You should test topology filters at the collection station before the station goes into service.[11]

Topology filters are, technically, data-stream filters. However, in effect, any change to a topology filter affects new and old data *alike*. When a topology filter is changed, the topology database of the management station is resynchronized with the filtered collection station topology database. Some objects may be deleted from the management station topology (if they do not pass the new topology filter), and some objects may be added (if they were previously rejected, but now pass the topology filter). As a result, the database of the management station ends up reflecting the topology being passed by the new topology filter, so the effect is that of a set-defining filter.

NOTE

Topology filters do not affect the events that are passed from collection stations to management stations. This means that any events that you have configured to be forwarded are in fact forwarded from the collection station to the management stations, regardless of whether the objects to which the events refer pass the topology filter or not. Listing particular event sources in the event configuration at the management station can reduce the likelihood of operator confusion.

Map Filtering

The purpose of map filtering is to give the operator visibility of only those objects in which he or she has interest. By default, Networks and Segments do not pass the map filter.

10. See “Distributed Internet Discovery and Monitoring” on page 46, Chapter 4, Procedures for Scaling NNM, and the *ovtopmd* reference page in NNM online help (or the UNIX system manpage), for further details.
11. See the reference page in NNM online help (or the UNIX system manpage) for *ovtopodump*, especially the `-f` option, for instructions on testing topology filters.

Map filtering occurs on a per-map basis, not a per-display basis. All operators who use a particular map have the same filtered view. Additional maps can be created and separately filtered for specific purposes.

Objects that are rejected by a map filter remain in the management station's topology database, and are still subject to status polling. Events from these objects pass through and are visible in the event subsystem, but this can be changed to show only events from objects on the operator's current map.[12]

Like the previously discussed types of filtering, map filtering can be configured so that (regardless of how much other topology data exists in the database) the map seen by the operator displays only the objects of interest to that operator. In general, this means objects that match some combination of attributes and attribute values; among the many choices, this can include:[13]

- All objects, by using no filter at all.
- Connectors, networks, and segments only.
- Nodes with specific `sysObjectID`s.
- Bridges, hubs, and their segments.
- Objects inside or outside of a particular range of IP addresses.
- Only nodes that support SNMP.

Map filtering is generally configured in the NNM interface.[14]

Map filters are set-defining filters. When you change a map filter, all the objects currently in the map are reevaluated to see if they pass the new filter; new objects must also pass the filter before being added to the map.

12. See the *xnmevents* reference page for details on the "filter by map" feature.
13. See "Filterable Objects and Attributes" on page 183 for specifics.
14. See the online help and Chapter 4, Procedures for Scaling NNM, for further details.

Defining New Filters

All filters, discovery, topology, map, and persistence, are configured by creating a text file in which you use a special filter definition language to define filters that pass and reject objects in the ways you want.[15]

NOTE While you can create multiple filter definition files, you can use only one at any given time. Hence, only the filters defined in the currently in-use filter definition file are available.

You can give each filter you define a sensible name. For example, a map filter that would pass only routers might be named `Routers`.

Effective use of filtering lets you monitor a broader domain with a given management station. For example, suppose you have a site with 5000 workstations and PCs connected by 300 hubs and routers.

If you are only interested in the connecting devices, you might be able to use a single collection station with filters configured to discover and monitor only the 300 hubs and routers. Without such filtering, monitoring all 5300 nodes might require three or more collection stations to maintain equivalent performance.

NOTE As a rule, the sooner you can filter out a particular object, the better overall performance of the network management solution will be. For example, by using discovery filtering to discover and monitor exactly and only those objects of interest, you can dramatically reduce the amount of data that all other parts of the system have to handle. The next-best type of filter, to reduce the data load on the distributed management system, is a topology filter. While a local collection station may need broader discovery for its own management domain, the management stations that use the collection station frequently do not require a full copy of the topology at the collection station; topology filtering is the way to prevent unnecessary data from being exported to the management station.

15. See Appendix A, The Filter Definition Language, for details on the syntax, semantics, and mechanics of filter definition.

Filter Utilities

NNM provides the utilities shown in Table 2-2, Filter Utilities, to help you build filters that do what you want and expect.

Table 2-2 Filter Utilities

Utility Name	Description
On Windows operating systems: *install_dir*\bin\ovfiltercheck On UNIX systems: $OV_BIN/ovfiltercheck	Provides syntax validation for a filter file. Can also be used to check individual Set, Filter, and Filter-Expression statements for correct syntax.
On Windows operating systems: *install_dir*\bin\ovfiltertest On UNIX systems: $OV_BIN/ovfiltertest	Tests all or part of a specific NNM topology database against a filter. Objects that pass the filter are reported to the output. Use the -c option to specify a collection station to determine which objects the management station polls on failover.
On Windows operating systems: *install_dir*\bin\ovtopodump On UNIX systems: $OV_BIN/ovtopodump	Use the -f option to compare a filter against the current topology database.

See the reference pages in NNM's online help (or the UNIX system manpages) for these commands to obtain details on their syntax and use.

Distributed Internet Discovery and Monitoring

Distributed Internet Discovery and Monitoring is one of NNM's keys to providing a scalable network management solution. The purpose of this section is to give you a picture of exactly which parts of NNM are distributed, and how data flows through the distributed solution. With this knowledge, you will be able to better understand what is occurring as you deploy NNM.

About Collection Domains

Within the NNM model of distribution outlined in "The Distribution Model for Network Node Manager" on page 33, it is useful to distinguish these two sets of objects:

Management domain

This is the set of all objects that are of interest to the user. In Figure 2-1, the management domain of the operator at Management Console 1 contains all the objects in the whole network (except objects that are removed by intervening filters). On the other hand, the management domain of the operator at Management Console 5 consists of only the objects directly monitored by Station F; no other objects are visible from that station.

Keep in mind that the management domain is an attribute of the management station, not the operator. Different operators might deal with different subsets of the management domain, based on map definitions and map filters.

Collection domain

This is the set of objects that are directly discovered and monitored by an NNM station. This typically includes any objects on which data collection (for trends or thresholds) is occurring.

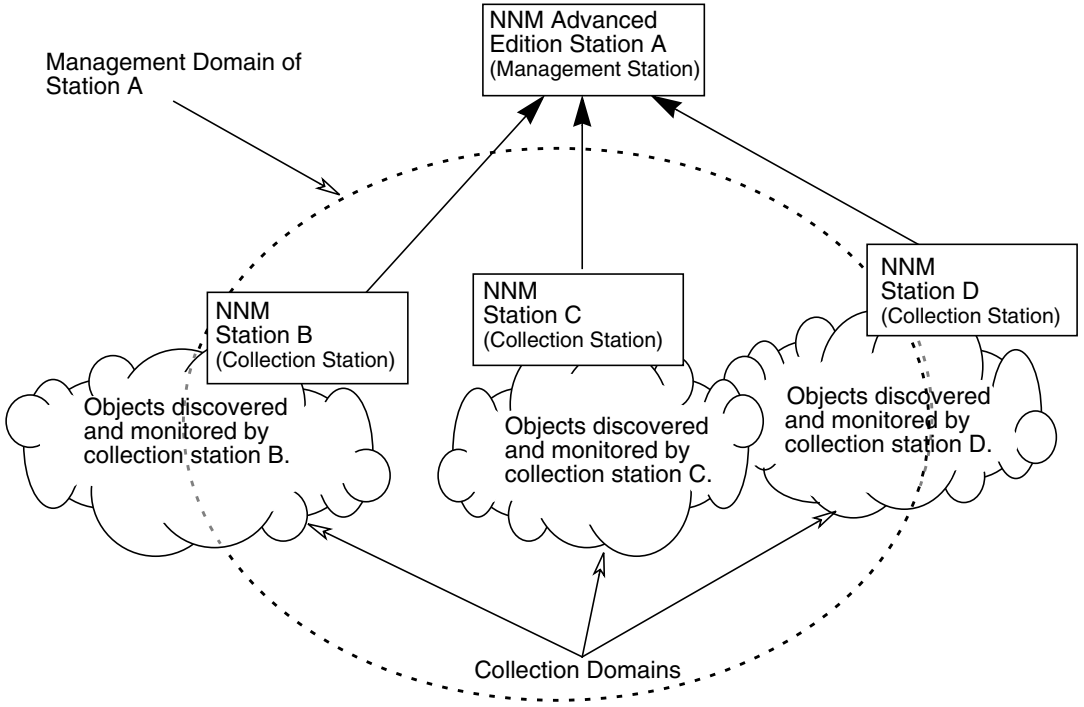
At a high level, these definitions can be depicted as in Figure 2-2.

In the diagram, the management domain of station A does not include all the objects in the collection domains of all the collection stations. Remember, the collection stations are probably management stations too. For each of these, the management domain is coincident with the collection domain.

However, the administrator of the overall network, working from Management Station A, is quite possibly very selective about which objects within the various collection domains of the collection stations are of interest to operators at the management station.

You can use filtering to select exactly those objects, and thus constrain your management domain to those objects that make the most sense for you.

Figure 2-2 Management and Collection Domains

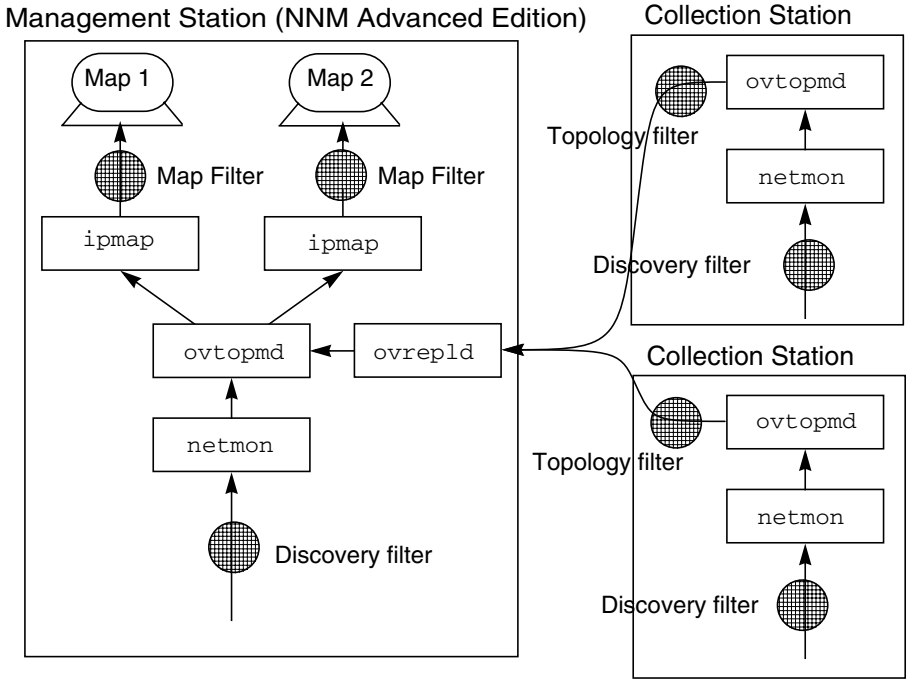


Distributed Services Architecture

Network Node Manager relies on the complex interplay of several different services (background processes on UNIX systems) to provide the level of functionality and performance you need. In making NNM a scalable solution, several of these services appear in locations remote from the management station.

Figure 2-3 is a diagram of the key services involved in a distributed deployment of NNM. Of these, the `ovrepld` service is central to the distribution of network topology; the others have other, stand-alone functionality. The `ovrepld` service is the "replicator" of the topology, responsible for integrating topology data from multiple collection stations into the management station's topology database.

Figure 2-3 Service Components in Distributed Internet Monitoring



The filters may or may not actually be present; they are included in the illustration at all the points where they would potentially exist. Note also that when a collection station is also fulfilling a local management

role, it runs local `ipmap` and `ovw` services too. And finally, understand that in the interest of simplicity, Figure 2-3 omits the role of the event system, which is illustrated in Figure 2-4.

How Distributed Internet Discovery and Monitoring Works

Figure 2-3, shows two collection stations working on behalf of the management station. Each collection station has its own collection domain, which is the responsibility of the local `netmon` service. You can set discovery filters to minimize unnecessary discovery and monitoring.

Having a station take the role of a collection station, or make use of collection stations, requires several other configuration steps. “Configuration Concepts” on page 56 introduces these configurations steps, and “Configuring Collection Stations” on page 134 gives complete details.

After the stations have been configured correctly, start the collection station as usual. The `netmon` service begins IP and level-2 discovery, and (along with the `ovtopmd` service) populates the local topology database.[16] It takes time for this process to finish, but to minimize management traffic on the network, you should allow NNM to finish (or nearly finish) discovery before you begin to use it as a collection station. The `Performance:Network Polling Statistics` tool can help you determine when the collection station has reached a steady state.[17]

After waiting for the collection station to build its topology database and reach a steady state, the management station administrator uses the `xnmtopoconf` command to begin using the discovery and monitoring services of the collection station.[18]

At this point, the management station obtains access to the topology data at the collection station. It requests the collection station to send the objects in the collection station’s topology database across the network. If the collection station has a topology filter configured, only those objects that pass the filter are provided to the management station.

---

16. See *Managing Your Network with HP OpenView Network Node Manager* for details about these and other services (background processes on UNIX systems) in Network Node Manager.
17. See the online help about this tool for more information about the procedure.
18. See the reference page in NNM online help (or the UNIX system manpage) for *xnmtopoconf*.

Topology data that is not derived from a local `netmon` service is not sent to the management station. This precludes the possibility of cycles (where station "A" discovers and reports "B," which discovers and reports "C," which again discovers and reports "A," …). It also precludes indirect multiple-level hierarchical management.[19]

The `ovrepld` service on the management station replicates the topology data from the collection station in the local topology database. "Replication" does not imply a 100% mirroring of the collection station topology database at the management station. Remember that topology filtering at the collection station means that some of the objects present in the collection station's topology database may *not* be present at the management station. The topology data that *does* pass the topology filter is kept synchronized between the collection station and the management station.

NOTE

The `ovrepld` service provides synchronization of the topology data *only*. It does not affect any other databases (such as `ovwdb`, `snmpCollect`, or `ovw`).

Once the topology merging is done, the `ovtopmd` service treats the new objects just like objects obtained from the local `netmon` service.

After the synchronization phase is complete, only *changes* in status and topology get sent to the management station. The `ovrepld` service updates the topology database of the management station to reflect the changes.

High Availability Management Information

One way to ensure your network management information is always available is to run your management stations on HP Service Guard clusters. If the primary processor goes down, processing automatically and transparently rolls over to another processor in the cluster. Configure Service Guard NNM stations using the `ov.conf` configuration file parameter `HOSTNAME=`. See the *ov.conf* reference page in NNM online help (or the UNIX system manpage) for more information.

19. See "Hierarchical Management" on page 99, especially Figure 3-4 and Figure 3-5, for more information on multilevel hierarchical management.

As part of the Service Guard interface, NNM offers `ovwrs` (self-restarting `ovw`). This interface runs `ovw` while monitoring the status of the console server process on the management station. If the connection to the server is lost, it informs the user and re-establishes its connection to the server. See the reference page in NNM online help (or the UNIX system manpage) for *ovwrs* for more information.

Collection Station Failover

NNM also provides capability to continue data collection when a collection station goes down. The management station's `ovrepld` service performs periodic status checks to verify the status and state of the currently managed collections stations. When a status check fails, the status of the collection station is downgraded one level. The default interval between checks is five minutes (but this can be reconfigured via the `xnmtopoconf` command). Thus, it will take 20 minutes before a "Collection Station Critical" event is generated; an event confirmation popup dialog is presented to the operator. The nodes affected by the failure are left with the status they had at the time the collection station stopped responding.

Collection Station failover enables a management station to take over status polling responsibilities for a collection station which is temporarily unreachable. When the collection station returns to normal, the management station returns to its previous state, delegating discovery and monitoring of those objects to the collection station. This feature is not intended to provide support for collection stations which will be down for a long time or which have been permanently removed from service.

In order to prevent `netmon` from being overloaded, failover only takes over status polling, not the full responsibilities of the collection station. Discovery and SNMP configuration polling are not picked up.

When the collection station returns to normal operation, the management station removes the failed over topology list from its polling queue and resynchronizes with the collection station. The status from the collection station is considered to be current, which means it takes priority and overwrites the status for the objects where were stored in the management station database.

IP and IPX (Windows operating systems) collection stations can failover to a management station. IPX interfaces from a Windows operating system collection station are not polled by a UNIX system management station.

You control failover operations through the management station's `Options:Network Polling Configuration:IP` menu selection, "`Status Polling`" tab; or through the `xnmtopoconf` flags "`failover`," "`nofailover`," and "`failoverFilter`." Refer to the reference page in NNM online help (or the UNIX system manpage) for more information.

Failover Filtering

The default operation is for the management station to pick up status polling responsibility for all nodes loaded in the management station's topology from that collection station. However, you can control the load on the management station by providing a failover filter that lists specifically which nodes should be polled. The failover filter behaves like a discovery filter which loads networks and segments, and then filters on nodes and interfaces before the data is given to `netmon`. The interfaces and the node to which they are attached are considered a unit. If one interface on the node passes the filter, then the entire node and all of its interfaces that the collection station polls pass the filter as well. You determine which objects are picked up on failover on a collection station by collection station basis.

Failover Considerations

The management station must have access to the interfaces which are to be picked up in a failover situation. A route needs to exist to the interface to be polled.

The possibility of overloading the management station is a consideration when configuring failover. The devices for which continuous status polling is especially important, such as routers and gateways, are good candidates for inclusion in a failover filter. The management station needs information about connection devices in order to detect secondary failures.

Overlapping Collection Domains

Reexamine Figure 2-2, and notice that sometimes the objects in the collection domain of one collection station are *also* within the collection domain of a different collection station.

For example, in Figure 2-2 the collection domain of Station C overlaps that of Station D, and both serve as collection stations for Station A. Such instances are not uncommon in practice. It is easy to imagine a

router that Stations C and D both care about and monitor. At Station A, that one router will appear in the topology data arriving from both collection stations.

When a management station receives multiple versions of the same object from different collection stations, it uses the rules described in the next section to automatically choose one version as the “primary” version; it then treats all other versions as “secondary.”

The station that provides the primary version of an object’s data is called the **primary collection station** for the object; any other station that provides information about that object is a **secondary collection station** for the object.

Note that a collection station can be a primary for one object, and a secondary for another object. In other words, the choice of primary vs. secondary is on a per object basis, not a per station basis. Furthermore, a collection station does not know or care whether it is the primary or secondary station for a given object.

Only the primary version of data is displayed in the operator’s map. Unless you specifically request information from a secondary version, all topology queries return information from the primary version.

Overlapping collection domains cause redundancy and can therefore waste memory, processing and network resources. In order to reduce the overlap, set the DeleteSecondary mode via `xnmtopoconf` on the management station.

The Selection of Primary and Secondary Collection Stations

NNM uses several rules to decide which of several versions of an object is primary (and by implication, which collection station is the primary collection station). The rules are evaluated in order, and the first rule that distinguishes a primary collection station is applied and the process terminates; remaining rules are ignored.

The rules NNM uses to establish the primary collection station for an object are as follows:

1. If only one station is reporting the object, that station is the primary collection station for the object.
2. If the object is an interface (`isInterface == TRUE`), skip to rule 7.

3. If only one station reporting the object is *a.)* managed (meaning currently in active use as a collection station), and *b.)* has noncritical status or has failed over, that station is the primary collection station for the object.
4. If the user has expressed a preferred primary collection station for the object[20], that station is the primary collection station for the object.
5. If only one station reporting the object is managing the object (meaning actively managing it, either by default or via `Edit:Manage/Unmanage Objects`, that station is the primary collection station for the object.
6. If one station reporting the object is a gateway, that station is the primary collection station for the object. (A gateway is a node with multiple interfaces performing IP routing functions.)
7. If the object (A) is completely contained in another object (B), the collection station for the container object (B) is the primary collection station for the object (A). For example, a non-gateway node should have the same primary collection station as the network that contains it.
8. If one of only two stations reporting the object is non-local, the non-local station is the primary collection station for the object.
9. The first station to report the object is the primary collection station for it.

You can express a preferred primary collection station for an object.[21] However, you should be aware that if conditions force NNM to reevaluate the primary collection station, your preference may be temporarily overridden, according to the preceding rules.

20. See the *xnmtopoconf* reference page in NNM online help (or the UNIX system manpage), especially the `-preferred` and `-clearPreferred` options.
21. See the *xnmtopoconf* reference page in NNM online help (or the UNIX system manpage).

For example, suppose the preferred primary collection station you selected for an object goes down and you have configured `nofailover`. If a secondary collection station can pick up reporting for that object, that station becomes the primary collection station for the object until the preferred primary station is again available.

If the down primary collection station has failed over to the management station, the management station does not make its local version of the object the primary. Where failed over objects were in overlapping domains with the management station, `netmon` handles the duplicate objects by only polling the device once and updating all of the objects with the status.

A management station will not pick up status polling on secondary objects during failover. There is already a primary collection station still providing information for those objects.

Overlaps with the Collection Domain of the Management Station

In some cases, particularly when an existing large collection domain is partitioned by collection stations, the collection domain of the management station overlaps the collection domain of a collection station. The question then arises, what should be done with the secondary (local) version of information about objects in the overlap?

You have three choices:

1. "`Allow Overlap`", which means that the `netmon` running on the management station should continue polling the node even though a collection station is also polling the node. This would also accept the associated resource consumption caused by duplicate monitoring (potentially over an expensive link) and duplicate information in the topology database.
2. "`Unmanage Secondary`", which unmanages the local version of objects (networks, segments, and nodes) in the overlap region, but keeps them in the database. (The local unmanaged version of the object will not be shown on any maps.) This eliminates the duplicate polling, but still incurs the data storage overhead. If the collection station should fail, the management station can be made the primary collection station to quickly restore management to the affected objects. This will not happen automatically. Refer to "Changing the Primary Collection Station for an Object" on page 152 or to the *xnmtopoconf* reference page in NNM online help (or the UNIX system manpage) for information on configuring failover.

3. "`Delete Secondary`" deletes the local version of overlapping objects from the topology database. This eliminates the overhead of both duplicate monitoring and duplicate objects in the database. You then rely on the collection station to provide all information about the objects in the overlap.

   When this is the mode, a local version of any overlapping networks is retained in an unmanaged state. The `netmon` on the management station will not discover nodes that belong to an unmanaged network. If a collection station fails, you will have to delete the collection station, then manage the networks and wait for `netmon` to discover the nodes that have been deleted unless you have failover configured.

The third option is the default behavior of an NNM management station. However, only non-connector nodes are deleted; NNM does not delete networks, segments or routers that overlap with the collection station's collection domain. This maintains visibility to the boundary of the collection domain of the collection station even in the event of collection station failure. The local versions of connector objects are put in the "unmanaged" state. If the normal primary collection station for these objects becomes unavailable, the local versions can become the primary version.

You can choose which behavior you prefer by using `xnmtopoconf`.[22]

Configuration Concepts

For a station to either be, or make use of, a collection station requires additional configuration steps.

This section provides only a brief summary of configuration to acquaint you with the general concepts, and gain a better understanding of what is required when deploying distributed IP discovery and monitoring. Detailed steps for configuring management and collection stations are provided in "Configuring Collection Stations" on page 134.

NOTE

NNM does not provide remote configuration utilities. Therefore, the configuration steps used to set up a collection station have to be made from the collection station itself. You can, of course, use `telnet` to log in

22. See the *xnmtopoconf* reference page in NNM online help (or the UNIX system manpage).

and remotely configure the collection station. Similarly, you could redirect an X display from the collection station back to the management console to do the configuration, but the speed of the display will degrade noticeably when using a wide-area network connection. You may have security issues as well in taking either approach.

Overview of Collection Station Configuration

To set up an NNM station to be used as a collection station, you have to do the following:[23]

- Configure the local collection domain, including any discovery filters desired.
- Configure the exported collection domain, by setting up topology filters.
- Configure SNMP security (get and set community names) in a way that will match the SNMP security configuration on the management station.

Once the above configuration steps are done, you need to restart the platform, via `ovstart`.

About the Remote Managers List As noted earlier, a management station designates and begins to use the services of a collection station by using the `xnmtopoconf` command to begin managing the collection station.

This causes the collection station to add that management station to a list of remote managers that it maintains. The remote managers list keeps up with the changes as management stations come and go.

At various places in the NNM interface, such as the `Event Configuration` window (when you are configuring event forwarding), this list is called the "`%REMOTE_MANAGERS_LIST%`".[24] Forwarding events

23. See "Configuring Collection Stations" on page 134 for more details.
24. See "Configuring Event Forwarding and Correlation" on page 155 for more details.

to this list makes system administration much easier, since you are not required to make individual updates whenever DIDM configuration changes.

For example, in Figure 2-1, suppose Station A has been using Station E as a collection station; the list of remote managers that Station E maintains includes Station A.

If Station B now starts using Station E as a collection station, it gets added to Station E's list of remote managers. That list now includes both Station A and Station B.

Overview of Management Station Configuration

To make use of collection stations, a management station needs to be configured as follows:[25]

- The `ovrepld` service must be running (it normally starts with `ovstart`).
- Any nodes on the network that are running NNM are potential collection stations; `xnmtopoconf -print` can help you identify the ones that have been discovered by the local `netmon` service. You have to select which ones you want to use, and manually add in any that have not been discovered locally. Versions of NNM that predate NNM 4.1 can not be collection stations.
- You need to configure SNMP security (get and set community names) using `xnmsnmpconf` in a way that will match the SNMP security configuration on the collection stations.

After this configuration is done, you can use the `xnmtopoconf` command to begin "managing" the collection station. In this context, "manage" means to use the IP discovery and monitoring services of the collection station.

25. See "Configuring for a Specific Collection Station" on page 140 for more details on this procedure.

Distributed Threshold Monitoring and Event Forwarding

The event subsystem for NNM allows forwarding of events (including topology changes, correlated events, and threshold events) from collection stations to management stations.

The services (background processes on UNIX systems) that NNM uses in basic event forwarding are:

- The trap service (`ovtrapd`).
- The postmaster service (`pmd`).

Figure 2-4 illustrates these services, and their relationship to other key NNM files and services.

The `ovtrapd` service is responsible for receiving SNMP traps and inform-requests on port 162 via UDP, the protocol of the SNMP standard. It buffers them, and passes them to the postmaster (pmd) service.

The `ovtrapd` service automatically provides an inform-reply to inbound SNMPv2C inform-requests.

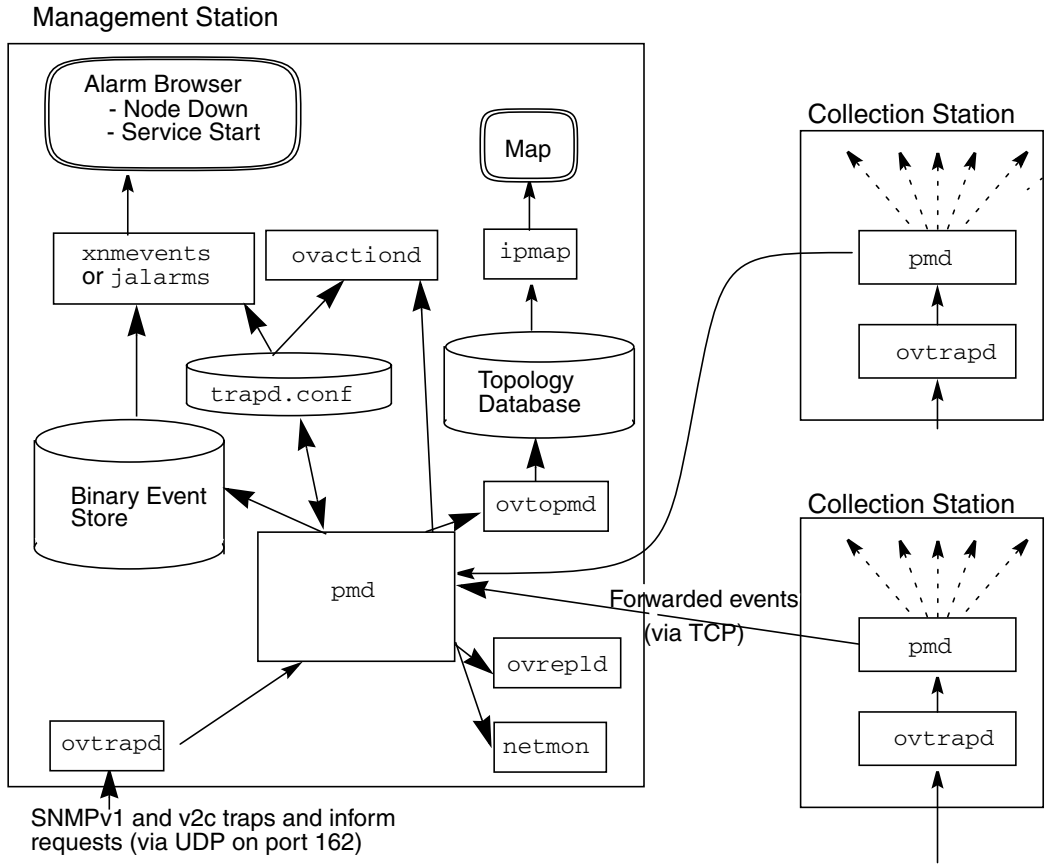
The postmaster service routes the traps it receives from `ovtrapd` to the appropriate subsystem. Moreover, the postmaster service is the central dispatcher that all management applications (both NNM and third-party) use to communicate with each other. It uses the TCP protocol to route events between them so messages are not dropped, even over noisy or congested links.

NNM is able to handle massive bursts of SNMP traps, up to 2000 per second. This is because the `ovtrapd` service can buffer traps while forwarding them to the postmaster for routing. This buffering reduces the likelihood that traps in a burst will be dropped.

The `Event Configuration` window supports event forwarding. It also supports an event-identifier naming convention, where SNMP version 1 traps, SNMP version 2C traps, and SNMP version 2C inform requests

use the same naming conventions for event IDs. See the online help and the *trapd.conf* reference page in NNM online help (or the UNIX system manpage) for details.

Figure 2-4 The Basic Event System in NNM



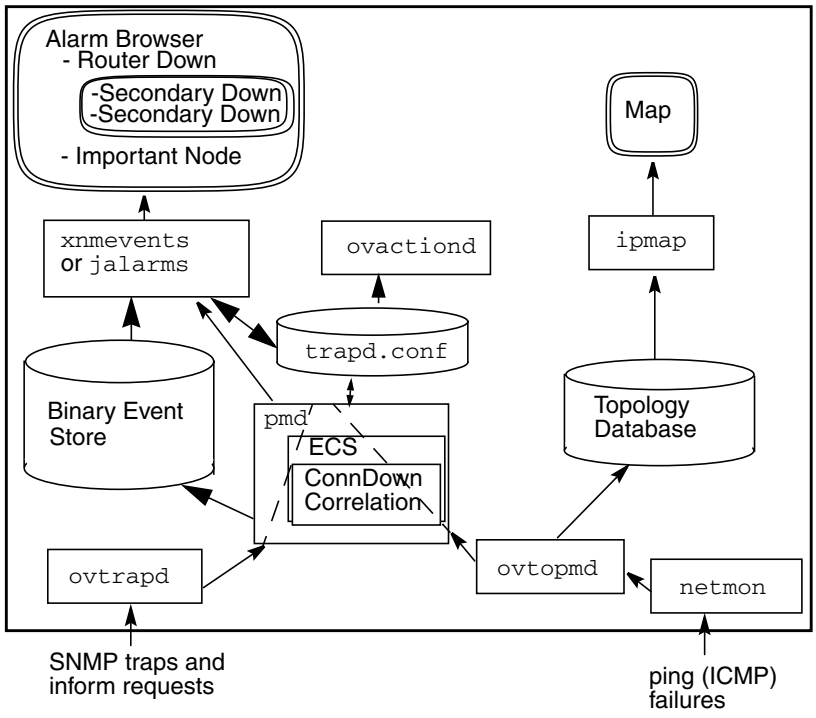
Event Correlation for "Connector Down"

One of NNM's key features is its ability to correlate incoming event information. See *Managing Your Network with HP OpenView Network Node Manager* for a complete description. The most useful correlation

rules for a distributed environment clarify information coming in when a connector (router, gateway, etc.) goes down and all the nodes beyond the connector become unreachable.

Since these events originate from `netmon` polling rather than from SNMP traps coming in through `ovtrapd`, the internal handling and forwarding is somewhat different from basic events. This is shown in Figure 2-5.

Figure 2-5 The Correlated Event System for "Connector Down"



When `netmon` recognizes a ping failure, which in the case of interest happens to be a connector failure, it notifies `ovtopmd` to update the topology database. `ovtopmd` notifies `pmd` that it has done so, and `pmd` runs the event through the Event Correlation System while `ipmap` updates the map. The correlated event then appears with a "children" indicator in the Alarm Browser.

In a distributed environment, `pmd` forwards the topology change for the nodes to the management station. At the management station, the `pmd` informs `ovrepld` to update the topology database. When `ovtopmd` makes that change, it informs `pmd` of the topology change. `pmd` then handles that topology change notice just like a local topology change notice, and routes it through the ECS correlation for "Connector Down." Note that the "Connector Down" event itself was not forwarded. The management station automatically derives that event from the topology change notification.

Operation of the "Connector Down" correlation on the management station receiving the topology changes may be slightly unexpected. Configuring reduced polling frequency for nodes beyond the connector will have no effect since the failure is not in the management station's discovery domain, but in the collection station's. The important node filter configured on the management station will apply as expected. However, the status (color) determination is done by the topology manager at the collection station, so is already in effect by the time the management station receives the topology change event. Status (color) configuration information at the management station does not apply to forwarded topology events.

The primary failure is always an interface, and interfaces are not displayed in the alarm browser by default. Use `xnmtrap` to enable display of interface events. However, the node status event corresponding to the primary interface failure is displayed. All secondary failure events are visible using the "`Show Correlations`" menu.

Data Collection for Distributed Threshold Monitoring

Threshold events based on data collection serve the same purpose in a distributed monitoring environment as in a centralized environment: to detect abnormal or deteriorating situations, and initiate corrective action. The corrective action might be to alert an operator, or it might be to trigger a predetermined action associated with the event.

A collection station can independently collect data (from the objects that it monitors) for threshold monitoring. When a threshold event is triggered, it can be automatically forwarded to a management station.

At the management station, it doesn't matter whether a threshold event was triggered by local data collection (done at the management station), or forwarded from a collection station. Forwarded threshold events are

automatically consolidated with local threshold events. Whether forwarded or locally triggered, the threshold event looks the same to the management station, and receives the same treatment.

NOTE Topology filters do not affect the events that are passed from collection stations to management stations. This means that any events that you have configured to be forwarded are in fact forwarded from the collection station to the management stations, regardless of whether the objects to which the events refer pass the topology filter or not. Listing particular event sources in the event configuration at the management station can reduce the likelihood of operator confusion. Also, this note does not apply to events internal to NNM.

See “Configuring Event Forwarding and Correlation” on page 155 for details about configuring event forwarding, including threshold event forwarding.

Remote Thresholds with Remote Actions

For some threshold events on collections stations, you may want to consider creating actions at the collection station. Where this is a reasonable tactic, it has the advantage that the action will be invoked even if contact is lost with the central site.

On-Demand Submaps

As described in “On-Demand Submaps” on page 26, the reason for enabling the on-demand submap feature is to improve NNM’s performance and reduce it’s use of memory and disk space. Rather than keep all the submaps in memory (which, for a large map, can require large amounts of memory), you can specify that many of the submaps only be created at the moment you need them.

A **persistent submap** is created and maintained for the life of the map. In contrast, a **transient submap** is created (and removed) only as necessary; that is, “on-demand.” Transient submaps are created and maintained by the `ipmap` service, and removed by the `ovw` service.

Figure 2-6, A Typical Hierarchy of Persistent Submaps, and Figure 2-7, A Typical Hierarchy of Transient Submaps, show the difference between persistent submaps and transient submaps.

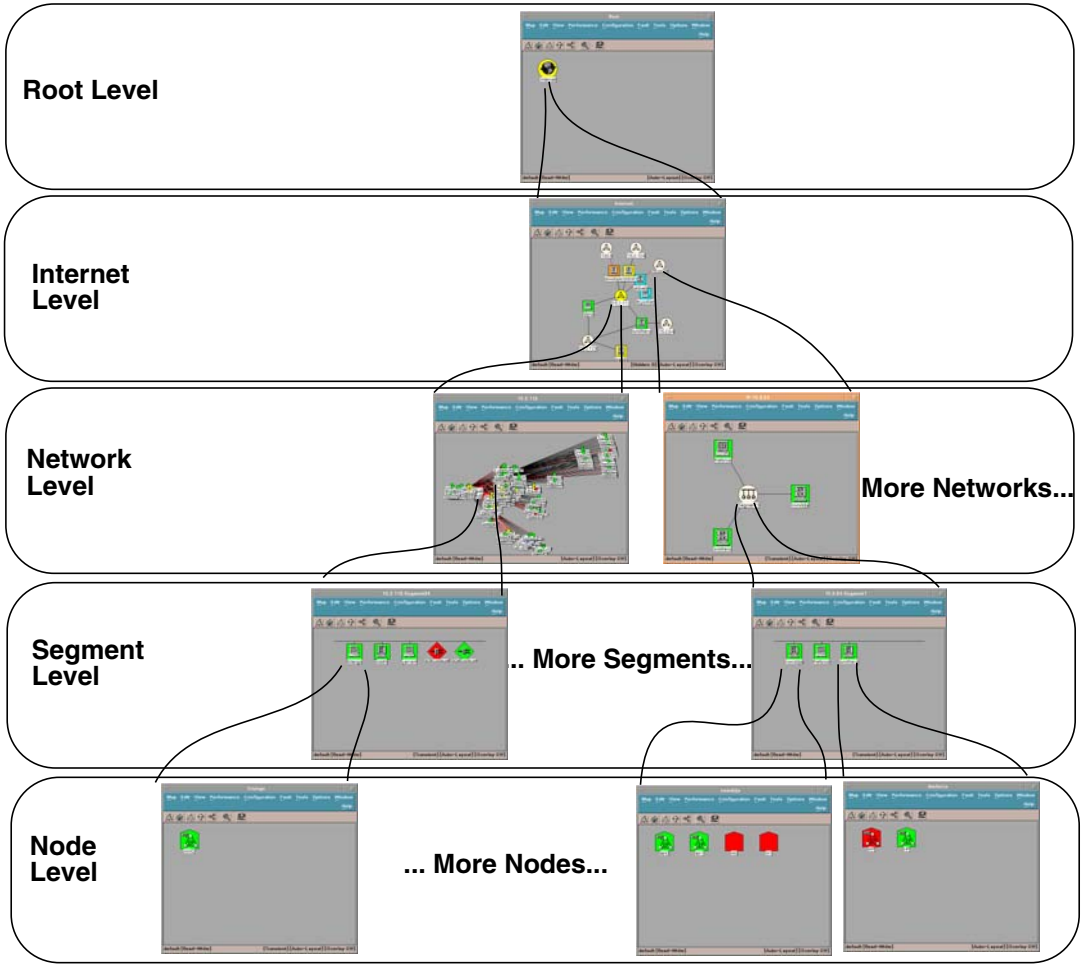
All the submaps in Figure 2-6 are persistent; this is the default when you first install NNM on UNIX systems. On Windows operating system management stations, the default is `Internet Level`. In contrast, in Figure 2-7 only the “Root” and “Internet” submap levels are persistent. Submaps from the “Network” level down are transient submaps. These are created only as they are needed. The `ipmap` service does not allocate memory for them, nor does it maintain topology changes in them. This saves both memory and execution time. Note that status from transient submaps is propagated up in the usual way.

The visible submaps (when on the same levels as the crossed-out ones) are also transient submaps, but the operator has navigated into that part of the submap hierarchy, and so these submaps are currently resident in memory.

You use the `IP Map Configuration` dialog box to set submaps to be persistent down to whatever level of the submap hierarchy you select. On Windows operating systems, the initial configuration for demand level is `Internet Level`.

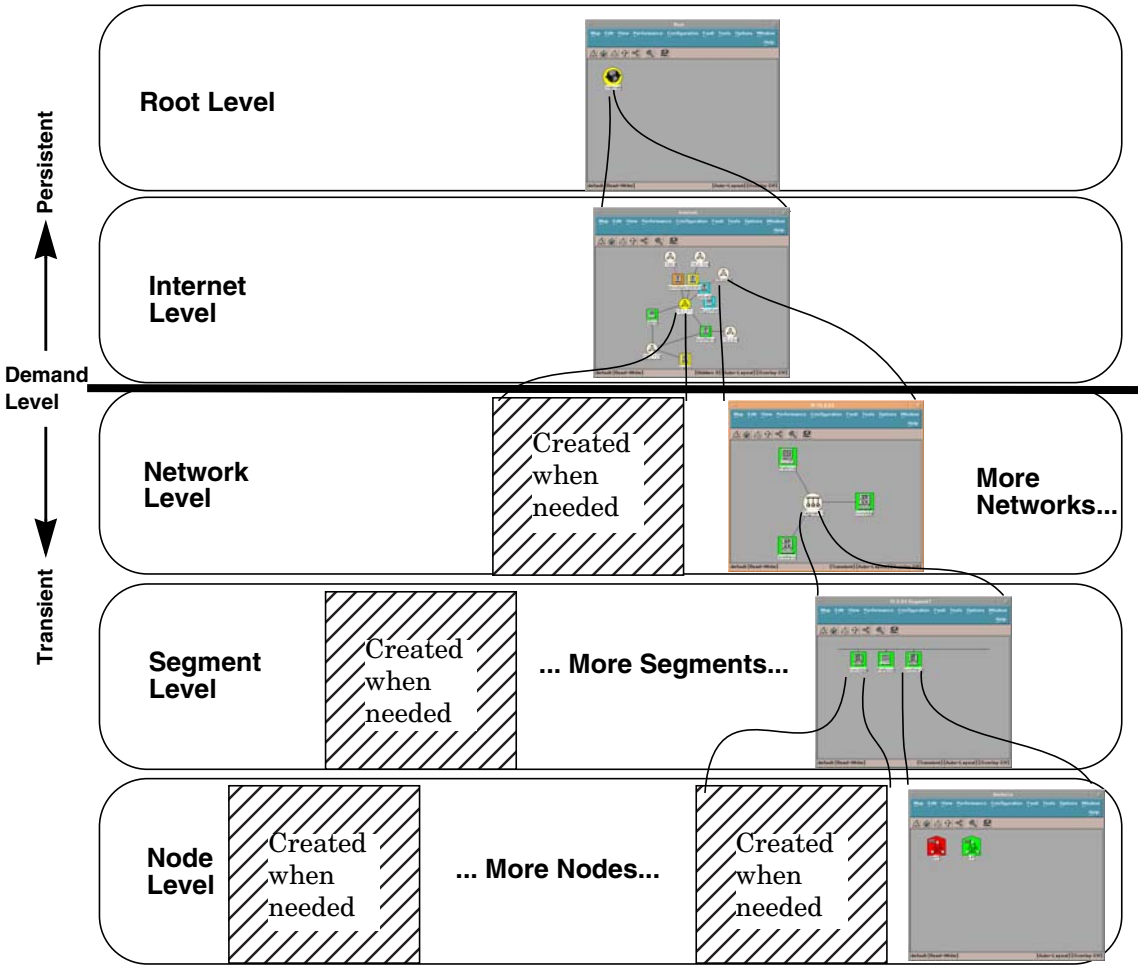
For example, in Figure 2-7, the level of persistence is set to `Internet Level`, so submaps at and above that level are persistent.

Figure 2-6 A Typical Hierarchy of Persistent Submaps



You can choose to make submaps persistent down to the `Internet Level` (as shown Figure 2-7), `Network Level`, `Segment Level`, or `All Levels` of the submap hierarchy. Submaps below the level you select will be transient. Of course, selecting `All Levels` means no submaps are transient.[26]

Figure 2-7 A Typical Hierarchy of Transient Submaps



26. Also see “Persistence Filtering” on page 68.

Most users of this feature will start with submaps set to be persistent at the `Internet Level` (as in Figure 2-7), and set it to lower levels only to improve submap-opening performance. This is typically necessary only when the higher-level submaps are densely populated.[27]

If you want to reset the on-demand submap feature to its default, use the `IP Map Configuration` dialog box in NNM to set submap persistence to `Unset`. On Windows operating systems, `Unset` sets the persistence level to `Internet Level`. On UNIX systems, `Unset` sets the persistence level to `All Levels`.

With on-demand submaps enabled at any level, most NNM features and operations work as they always have. Locate operations and IP status propagation, in particular, are not limited to what is currently in memory. Both functions work across all submaps, regardless of whether they are persistent or transient.

In cases where a located item is not yet in memory, the necessary submap is created, on-demand, if and when the user opens that submap from the locate dialog box.

Transient Submaps and the Submap List Dialog

The only feature that changes its behavior when the on-demand feature is enabled, is the `Submaps in Map` dialog box. Only submaps that are currently in memory (that is, either persistent, or transient but currently in memory) can appear in the `Submaps in Map` dialog box.[28]

In other words, when the on-demand submap feature is enabled, submaps that are not presently in memory are not present in the `Submaps in Map` list. This behavior may alter the usefulness of the `Submaps in Map` dialog box, depending on the expectations of the operator.

To reduce the possibility of operator confusion when the on-demand submap feature is enabled, you may wish to remove the following menu item (which opens the `Submaps in Map` dialog box):

```
Map:Submap->Open
```

27. See also the “Caution” block on “Configuring Management Consoles” on page 109.
28. See the online help for more information about this dialog box.

See *Managing Your Network with HP OpenView Network Node Manager* for information about how to remove a menu item.

Persistence Filtering

Unlike the filters described earlier, the purpose of persistence filtering is not to reduce the amount of unnecessary management data in the data stream. Instead, the purpose of persistence filtering is to let some older applications that are not inherently compatible with the on-demand submap feature operate normally when the on-demand submap feature is enabled.

NOTE This is probably not an issue on Windows operating systems, since there are few "older" applications. However, some third-party developers may port their older UNIX system applications without changes.

Suppose an application has been created with the implicit assumption that the object or objects it is interested in are always available in memory. Such an application may fail or exhibit unstable behavior if the object it needs resides in a transient submap, and is thus not in memory at all times.

For example, a vendor you use may have developed an application that adds statistics component symbols under the router symbols created by the `ipmap` service. Such an application may assume that routers are part of the map (that is, present in memory) as soon as they are discovered. The on-demand feature, when enabled, invalidates that assumption.

A persistence filter can provide backwards compatibility for this type of application. The necessary persistence filter for the example application passes all routers. Then any submap that contains a router is made persistent, so that the expectation of the application is met.

A persistence filter tells the `ipmap` service that any submap containing an object that passes the filter should be made into a persistent submap. The object will then be in memory, as the application expects.

Persistence filters are set-defining type filters. A persistence filter is automatically applied to existing map and object data to determine which submaps must be persistent. When you change a persistence filter, all the objects currently in the topology database are reevaluated to see if

they pass the new filter. Submaps containing objects that pass the new filter are made persistent. Submaps that do not contain such objects are made transient if they are below the current persistence level. [29]

Comparing Map Filters and Persistence Filters

A persistence filter is different from a map filter. A map filter specifies a subset of the topology data that is allowed on the map. The persistence filter states which objects should be made persistent in memory; those objects are made persistent by making the submaps they reside on persistent.

You can think of this as a two step process. First, the map filter passes a set of objects that are allowed to be present on the map. Then the persistence filter tests these objects to see which, if any, must be persistent in memory.

No matter what the persistence filter specifies, the user still has access to all of the objects that passed through the map filter.

When an object is found to meet the criteria in the persistence filter, all submaps containing that object are made persistent. That is, a submap is made persistent if it contains even one persistent object; it might also contain objects that do not meet the persistence specification criteria. In addition, the child submap of the persistent object is also made persistent.

Because persistence and map filters are both set-defining filters, if you change either for a given map (via the `IP Map Configuration` dialog box; see the online help for details), the `ipmap` service recalculates which submaps should be persistent. This might take a long time for larger databases.[30]

29. See “On-Demand Submaps” on page 64 for more details on persistence filtering. See “Specifying a Persistence Filter” on page 125 for configuration information.
30. See also “Map Filtering” on page 42.

Management Consoles

Distribution of management consoles is done via NFS (or Windows operating system file sharing) mounting of the map database and the appropriate configuration files, and by then using the distributed RPC capability of the object database service, `ovwdb`, the topology database, and the event system.

Figure 2-8 is a diagram of the resulting services, and where they are located. As shown, each management console actually runs the user-specific `ovw`-related services. The management station keeps the common databases, and runs the services (background processes on UNIX systems) common to all users.

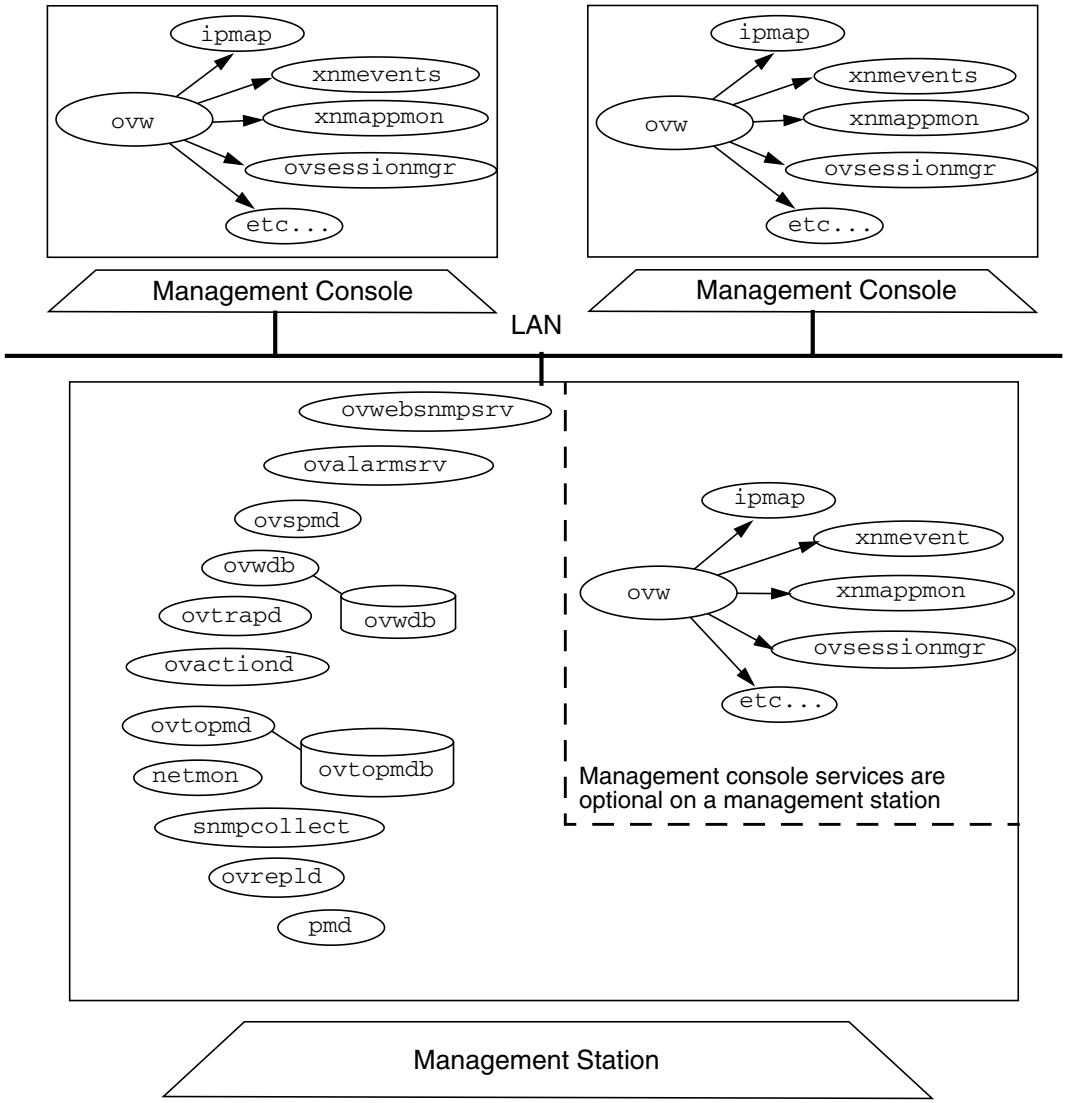
This relieves the management station of a significant demand for its resources (memory and CPU), since otherwise each operator would have independent copies of these services executing on the management station.

Relocating the `ovw`-related services allows the management station to apply more of its resources to network management and monitoring activities, and less to supporting the operators.

The object database remains centrally located on the management station, and is accessed by the management consoles over the network sockets. In addition to the various `ovw`-related services that communicate with the NNM services via socket connections, the map database and configuration files use NFS (or Windows operating system) file sharing.

NOTE Management consoles are *not* the same as NFS diskless clients. NFS diskless is an operating system concept. As described above, the distribution mechanism for management consoles uses NFS and sockets.

Figure 2-8 Management Consoles



Operators can each have their own maps. If, on the other hand, they share a map, only the first operator to open the map can edit it; others have read-only access. When using remote console into a management server that is being accessed by other remote or local console operators, maps currently open by other operators are available in read-only access. NNM will not change that status as maps are closed. The remote console must close and reopen the map to change access.

A remote console may open more than one map at a time by launching additional instances of NNM, however, this may result in performance problems on the remote console station.

Management Consoles and On-Demand Submaps

The on-demand submap feature is an attribute of a map, and is shared by all operators viewing a particular map. In other words, if the on-demand submap feature is enabled for a given map, any operator at a management console who opens that map has the same on-demand setting.

Of course, if multiple management consoles are displaying separate maps, how (or even whether) one map uses the on-demand submap feature has no effect on any other consoles viewing any other maps.

Customized Views of the Network

You can have several management consoles that each run a different map (but all maps stem from the one topology database on the management server). Moreover, each map can use a different map filter.

The result is that each management console can have a customized view of the network topology.

To set up an arrangement of customized views, you need to follow these general steps:

1. Decide on the nature of the customized views you want to present. For example, one view might include only the connectors in your network; another might include only certain servers.

2. For each custom view you want, create a filter that passes only the objects you want visible in that custom view.[31]
3. At each management console, create a new map (`Map:New`), and apply the appropriate map filter for the operator at that console.

Management Consoles and X-Terminals

In principle, it is possible to run several X-terminal sessions from a management console, or directly from the management station. You might speculate that this would allow NNM to support more operators.

However, due to internal constraints, this approach *does not* allow the management server to support more operators. You should limit the number of operators per management station to avoid degrading the performance of the interface.

Attaching X-terminals to a management console might nevertheless have cost savings, if the management console host has sufficient CPU, memory, and disk space to support the X-sessions, and low cost X-terminals are available and suitable for the operators. The key services described earlier are still off-loaded from the management station to the management console.

31. See “Filters” on page 37 and Appendix A, The Filter Definition Language, for more information on creating filters.

Interface Managed-state Automation

As you probably know, NNM lets you “unmanage” interfaces. NNM does not do status polling on unmanaged interfaces, so setting interfaces to the Unmanaged state significantly lightens the load on the management station.

NNM lets you automatically manage or unmanage interfaces based on filters. Interface managed-state automation makes it easier to manage large environments if many of your discovered interfaces do not require active status polling.

For more information see “Configuring Interface Status Polling” on page 160.

Microsoft Terminal Server Access

You can connect to Network Node Manager (on a Microsoft® Terminal Server) from a remote workstation through a Terminal Server Client. You can do any of the following:

- Install or upgrade NNM on the remote server.
- Perform administrative NNM functions such as backup/restore of NNM database.
- Open NNM and display the network map onto the local workstation.
- Start multiple Terminal Server Client sessions to the same remote NNM management station, open NNM in each session and display the network maps onto the local workstation.
- Start multiple Terminal Server Client sessions to different remote NNM management stations, open NNM on each NNM management station and display the network maps onto the local workstation.

The following examples provide a few of the possible implementation strategies available to network managers when running the NNM application on servers with the Microsoft Terminal Server features enabled.

Accessing a Single NNM Station From Multiple Terminal Server Clients

In this configuration, (Figure 2-9) multiple network operators (running the Terminal Server Client on their local workstation) are able to access a Windows Server (configured to run in multisession mode) and run NNM.

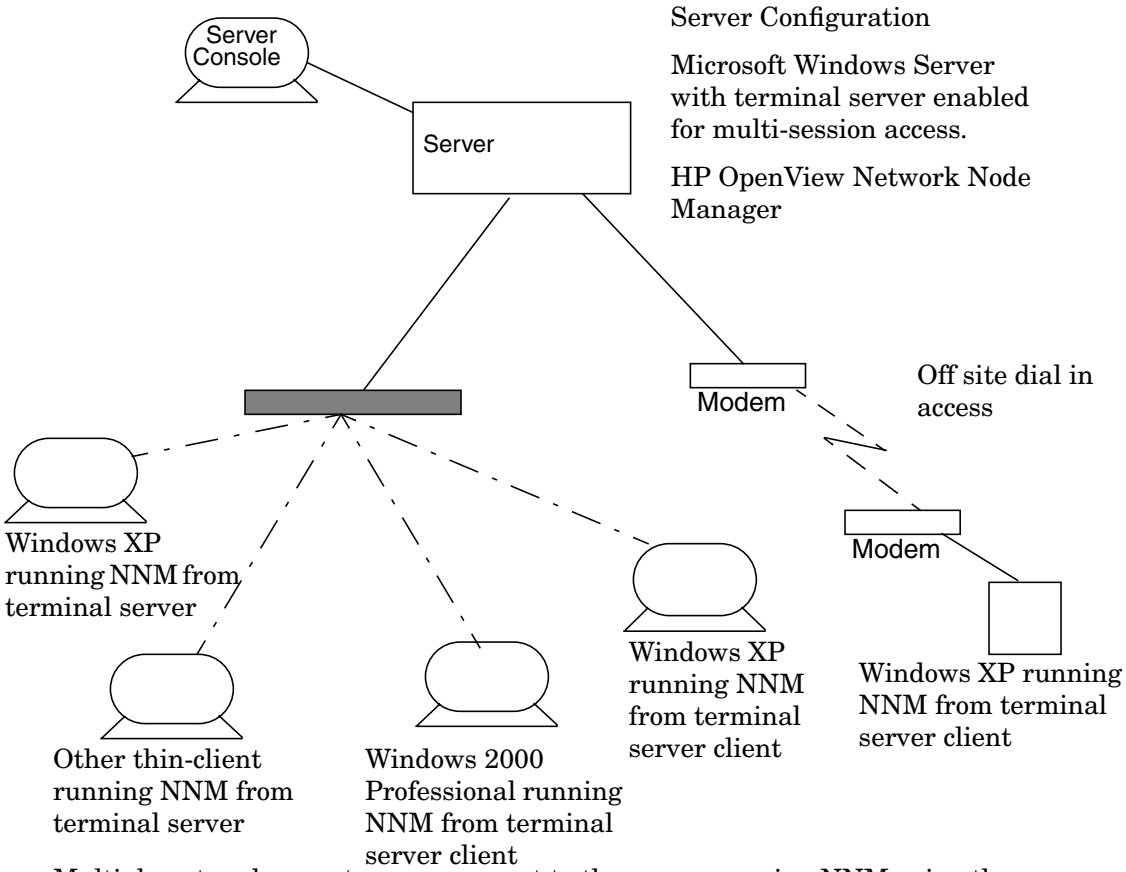
Each network operator is able to open NNM and monitor a different segment of the network. If needed each operator can open a different map and perform all associated network management functions. In the multisession mode, the number of connections is limited by the number

of client licenses available. Accounts with administrator privileges can be used to start and stop NNM processes, run NNM database backups, and install NNM patches.

Figure 2-9 Multiple Terminal Server Clients Accessing a Single NNM Station

Multisession Terminal Server Access Enabled.

Server located in computer room with secured access



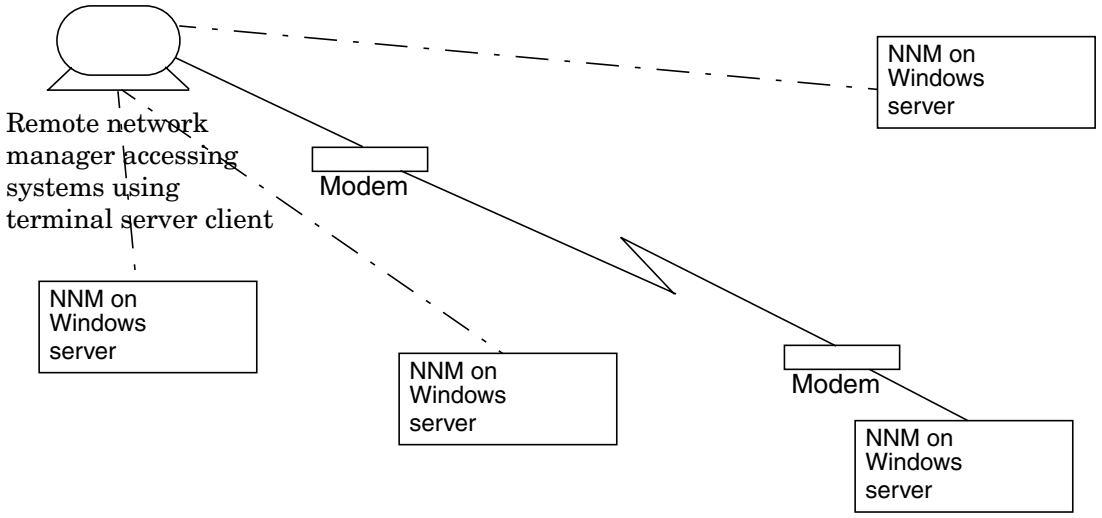
Multiple network operators can connect to the server running NNM using the terminal server client. The operators can open NNM maps and monitor the network segment from their local workstation.

Accessing Multiple NNM Stations From a Single Terminal Server Client

In this configuration, (Figure 2-10) a single network manager is able to access multiple Windows servers configured to run under Remote Administrator mode. The NNM application is installed on each Windows server. The remote network manager, using the Terminal Server Client, is able to connect to each remote system, monitor the network, and perform all network management functions. With this configuration a network manager has the ability to access and display multiple NNM sessions on their local workstation.

The remote administration mode limits the number of connections to the server (maximum of two) and only grants connections to accounts that have administrator privileges.

Figure 2-10 Accessing Multiple NNM Stations



Server Configuration

Microsoft Windows Server with terminal server enabled and installed as Remote Administrator mode.

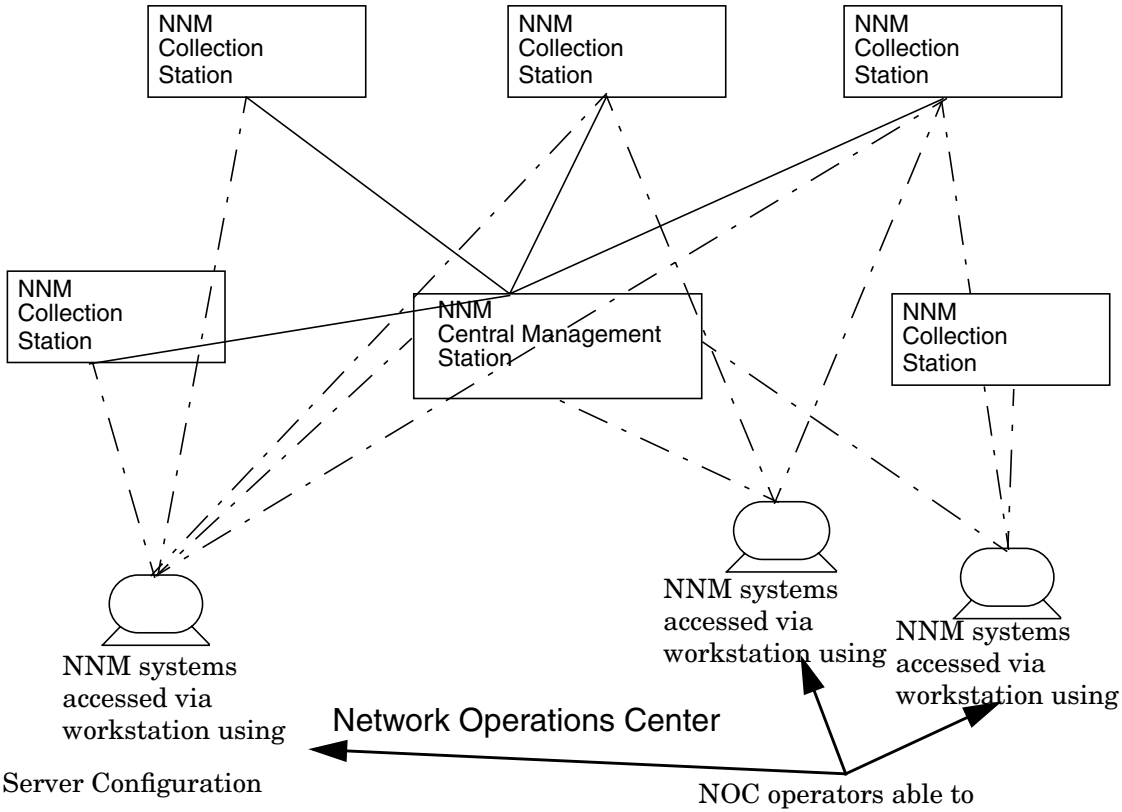
HP OpenView Network Node Manager running on each server to manage the local network

Accessing Multiple NNM Stations From Multiple Terminal Server Clients

In this configuration, (Figure 2-11) a central Network Operations Center (NOC) is responsible for monitoring the entire network. The network management architecture is constructed of multiple Windows servers running NNM configured as collection stations, each collection station forwards information to a central NNM management station. The NOC operators are responsible for monitoring the health of the network.

The Terminal Server Client enables operators to access multiple NNM stations and to monitor different segments of the entire network from a single workstation. Operators can open multiple NNM sessions on each NNM station to view more than one map. Operators with administrator privileges can remotely start and stop NNM processes, run NNM database backups, and install NNM patches.

Figure 2-11 Multiple NNM Stations Accessed by Multiple Terminal Server Clients



3 Strategies for Deploying Network Node Manager

This chapter contains guidelines and ideas for deploying an HP OpenView Network Node Manager (NNM) solution that meets your goals and fits well with your organization.

The purpose of this chapter is to help you understand some of the reasonable models you can use as you plan your network management solution, and some criteria you can use to select a model.

It includes general approaches and general tips to help you succeed. With this knowledge, you will be able to better plan and implement your distributed, scalable network management solution.

NOTE

The information in this chapter is not introductory; readers are assumed to have read the previous chapters. Readers are also assumed to be conversant with NNM's key features for network monitoring, data collection, event handling, and so on. This includes a functional understanding of some key elements in non-distributed NNM, including `ovw`, `ipmap`, `ovtopmd`, and `netmon`. See *Managing Your Network with HP OpenView Network Node Manager* for details about these key services (background processes on UNIX systems) in Network Node Manager.

The remaining chapters and appendices contain specific configuration procedures to use with the scaling and distribution features of NNM.

Planning the Right Approach

Now that you understand Network Node Manager's scalable and distributable features, your questions naturally turn to issues like:

- "What should I *do* with all this?"
- "Can I just ignore NNM's scalability features for now? Should I?"
- "How would I employ distribution and scalability on a massive scale?"
- "What makes sense in my organization?"

You have a wide range of choices in your deployment of NNM. The simplest choice, and one that will make sense to some, is to use none of the scalability features at all. Others, who manage large, dispersed networks, will elect to scale up their use of NNM to deal with that large environment. Still others will find that a mix of some scalability features meets their requirements.

You can combine several different models as you plan your strategy. Each fits well with a particular set of goals. Your choice of approach will be driven, in large measure, by the nature of your organization and the problems you are solving with your network management solution.

Remember as you plan that you are never "locked-in" to one strategy; as your organization changes, you can adapt your use of NNM to keep up with those changes.

Organizations differ in many ways. Organizational variables to consider as you begin planning your deployment of NNM include:

Size — The needs of a small organization, with perhaps a few hundred nodes, are very different from the needs of a corporate campus where you may have thousands or tens of thousands of networked objects. The networks in larger organizations are inevitably larger and more complex than in small organizations. Consequently, the larger the organization, the stronger are the reasons to use more of the scalability features of NNM.

Structure — Some organizations, like the military, have clear hierarchies of authority and reporting. Others, like universities, may consist of looser affiliations between

cooperating but largely autonomous groups. These structural differences will have a significant effect on how network management is conducted.

Geography Some organizations, like a hospital, are geographically compact but require reliable network management. Within the confines of the hospital, NNM may have few requirements for scaling. But suppose that hospital is part of a medical services company; the central site, located thousands of miles away, may want to perform network management functions in *all* their many hospitals. Then issues of distribution and scalability are very important.

Trends The network that supports an organization is in a constant state of change. Yet it is important to understand general trends in how a network is evolving: is it stable? is it becoming more geographically dispersed? bigger? more expensive to manage? You will want your network management plans to account, as best you can, for the trends in your network.

General Considerations about Scalability and Distribution

The following general points are useful when planning your deployment of NNM. Margin notes summarize key concepts from the adjoining text.

Limits to netmon affect the maximum number of nodes you can monitor.

Due to the nature of ICMP pings, the `netmon` service (background process on UNIX systems) has limits to the amount of polling it can perform in a given period of time. You can maximize the number of nodes the `netmon` service can handle (while maintaining a consistent five-minute status polling cycle) by increasing the polling interval, especially for less important nodes. See “Configuring SNMP” in *Managing Your Network with HP OpenView Network Node Manager* for details on how to set special polling intervals. For more current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*.

You may also want to take advantage of interface managed-state automation, as described in “Interface Managed-state Automation” on page 74.

If the environment you want to manage has more nodes than you can monitor efficiently with a single `netmon` service, you will want to have another `netmon` service, in the form of another collection station, to off-load some of the polling work from the management station.

WAN performance also limits remote monitoring.

A WAN link introduces propagation delays that compound the above issue. That means that if your environment is split, with part of it accessible via LAN and the rest via WAN, the actual limit of managed nodes for a `netmon` service will be less than the limit stated above. How much less depends on the proportion of nodes on the LAN link vs. on the WAN link. In general, you will want to avoid monitoring nodes over a WAN link, unless you have only a few nodes at the remote site, or the nodes there are unimportant, and can receive infrequent polling.

In some circumstances, however, remote monitoring is necessary. As a rule-of-thumb, if a `netmon` service is monitoring remote sites, reduce the number of nodes it handles by one-tenth for each remote site. Thus, if a collection station is monitoring, say, 3 remote sites, you would expect it to handle up to 1400 nodes, rather than 2000.[1] The calculation for this estimate is as follows: 2000 - ((0.1 x 2000) x 3) = 1400.

In addition, remote monitoring via a WAN has very high operating costs in network traffic compared to the operating cost of monitoring via a remote collection station. Each status and configuration poll and response would have to cross the link, rather than simply any topology changes.

Take advantage of NNM wherever you can.

Given the low performance of WAN links, you can always achieve higher NNM performance by using a collection station for remote (WAN) nodes.

Scalability is part of every NNM station. This includes using NNM Advanced Edition 250 as a management station or collection station as well as using NNM Starter Edition 250 as a collection station. It makes excellent sense to take advantage of NNM wherever it exists in your network. For example, you might have a “hidden” NNM station at a remote site, running NNM as the basis for another application like HP OpenView Operations. Consider using it as a collection station.

1. Again, this estimate assumes that reliable 5-minute polling of all nodes is required.

If your remote site has no NNM installation, you may want to install a collection station if you have perhaps fifty or more nodes to be monitored; NNM 250 might be a good choice for a small remote site. As an alternative, you could have a collection station located at one remote site, and monitoring multiple other remote sites.

When deciding whether to use the NNM Advanced Edition or the NNM Starter Edition, remember that the NNM Advanced Edition can function as a management station, collection station, or both. The NNM Starter Edition can only function as a collection station.

The final layout of your solution will depend at least in part on an analysis of the costs involved with each alternative you consider.

Ensure that you can always contact your collection stations.

Relying on a remote station for network management carries certain risks, but that risk can be driven down or eliminated.

As an example, suppose some collection station is your sole source of information for a part of your network, and the main gateway between the management station and that collection station goes down. By default, it takes up to twenty minutes before a "Collection Station Critical" event is generated, and an event confirmation popup dialog is presented to the operator. The nodes affected by the failure are *not* marked "Unknown".[2]

You can diminish the likelihood and effect of such a situation in three ways.

- First, arrange to have overlapping management domains or failover coverage, particularly for key connection nodes between the management station and collection stations.
- Second, make contingency arrangements that allow alternative communication paths to the collection station. You can establish redundant routes to the collection station via multiple gateways, or by adding a direct link from the manager to the collection station if that is possible. Consider lower-cost approaches that may have lower (but still acceptable) performance in an emergency.

  Third, use HP Service Guard for your management and/or collection stations.

2. See the note about `ovrepld`'s periodic status checks in "How Distributed Internet Discovery and Monitoring Works" on page 49.

Use as many collection stations as you can efficiently manage.

The more collection stations you have, the smaller the impact of temporarily losing any one of them, and the easier it is to roll management of that region to a different collection station. Within the limits of your ability to configure and manage collection stations, use as many collection stations as possible to ensure redundancy and off-load polling. For more current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide.*

Consider the flexibility of NNM 250

The NNM Starter Edition 250 or NNM Advanced Edition 250 or products are limited to managing 250 nodes per license, but you can purchase multiple licenses to increase that number incrementally. Check current pricing to decide if multiple NNM 250 licenses are preferable to a full NNM product in your situation. NNM Starter Edition 250 can act as a collection station for multiple management stations, but cannot receive data from other collection stations. NNM Advanced Edition 250 can act as a collection station for multiple management stations, and can received data from other collection stations.

Partition your Internet submap to reduce the visual complexity

Because it is feasible to manage many thousands of nodes, your internet submap may become densely populated, and difficult to work with. You can solve this problem by creating Partitioned Internet submaps to create reasonable groups of nodes. See “Partitioned Internet submap” in the index of *Managing Your Network with HP OpenView Network Node Manager* for details.

NOTE

Be sure to consult the latest version of the *HP OpenView Network Node Manager Performance and Configuration Guide* for more information about configuring NNM in large environments.

General Strategies for Deployment

This section gives a few general pointers on how to deploy a distributed network management solution in your organization.

Plan ahead

Create a master plan for your organization that substantially outlines your goals for network management distribution and scaling. Keep in mind that your plan will continue to evolve along with your network.

Network planners, operations managers, and HP OpenView administrators need to collaborate in the planning process. They each bring a unique perspective, and getting early and continuing consensus important to the success of the project.

Describe and diagram the current state of your network management solution. Do the same for the final state you intend to achieve. Work together to identify each step that will be needed to migrate to the new arrangement.

Start small

You don't have to implement every aspect of your final plan all at once. Begin by setting up just one piece of the final picture.

You might choose to implement on-demand submaps on a management console. Or, you might decide to set up one collection station to see how that goes. Choose something that makes sense in your final plan, implement it, and then test it thoroughly before you proceed.

Go slowly

Add more small pieces of your final plan one at a time. As you go, keep testing and rechecking your implementation to ensure that you are obtaining the results you want.

Be particularly attentive to your filters. Carefully designed and implemented filtering can be central to the effectiveness of your solution, and the set of filters provided in the default filter file will be useful to many people. Those who need to design other filters should be aware that creating filters can be complicated, and requires considerable attention to detail.

Also, review your polling intervals (under the menu item `Options:SNMP Configuration`). Poll your most important nodes more often (for example, every 3 minutes) than your less important nodes (for example, every 15 or 30 minutes). This will help you maximize the benefits of scalability.

Keep records

Maintaining a large network management solution is not trivial. Along with the planning documents, keep records of what steps have been taken and when. Note

any issues that came up with a step, and, more importantly, the resolution of those issues. This “paper trail” can be useful in the event that you want to roll back to an earlier configuration for troubleshooting or re-engineering your distribution scheme.

Backup your system NNM pauses all services yet queues information coming in from collection stations, so you can backup your database without missing any information. Refer to *Managing Your Network with HP OpenView Network Node Manager* for more information.

Deployment Strategies for NNM

You have a number of choices about which of the scalability features of NNM you want to use, and to what degree you want to use them. This section describes several models of deployment to help you understand some alternatives to consider as you put scalable NNM into operation.

The following models are considered in this chapter:

- Fully centralized
- Centralized hierarchical
- Hierarchical
- Cooperative independent

NOTE

This chapter uses the above models to show how different organizational structures can be reflected in the deployment of network management within the organization. The deployment models in this chapter are not, by any means, the only way to convey the ideas presented here. This can not be overemphasized. Few organizations, if any, will use one of these models in a pure form. As you read this section, you are encouraged to look for ideas and approaches that can be applied in some part of your own organization. It can be helpful to imagine that the strategies in this section apply to very small parts of your overall network management solution. For example, a management station that has a hierarchical relationship to one collection station might simultaneously have a cooperative relationship with another management station.

Use of Scalability Features in NNM Deployment

All of these features are available to you in any way you choose to deploy NNM. The following table shows the most common scalable features used in each type of NNM scenario.

Table 3-1 Scalability Features in Use Models

Scalability Feature	Centralized page 93	Centralized/ Hierarchical page 96	Hierarchical page 99	Cooperative page 104
Scalable Product Structure	X	X	X	X
Distributed Collection Stations, page 48		X	X	X
Management Consoles, page 70	X	X	X	X
Network Presenter web GUI, *Managing Your Network with HP OpenView Network Node Manager*,	X	X	X	X
On-Demand Submaps, page 64	X	X	X	X
Discovery Filtering, page 40	X	X	X	X
Topology Filtering from Collection Station to Management Station, page 41			X	X
Map Filtering for Operators, page 42	X	X	X	X
Event Forwarding, page 59		X	X	X
Threshold Event Forwarding, page 59		X	X	X
High Availability - Service Guard, page 50	X	X	X	X

Table 3-1 Scalability Features in Use Models (Continued)

Scalability Feature	Centralized page 93	Centralized/ Hierarchical page 96	Hierarchical page 99	Cooperative page 104
High Availability - Failover with Failover Filtering, page 52		X	X	X
High Availability - Overlapping Domains, page 55			X	X

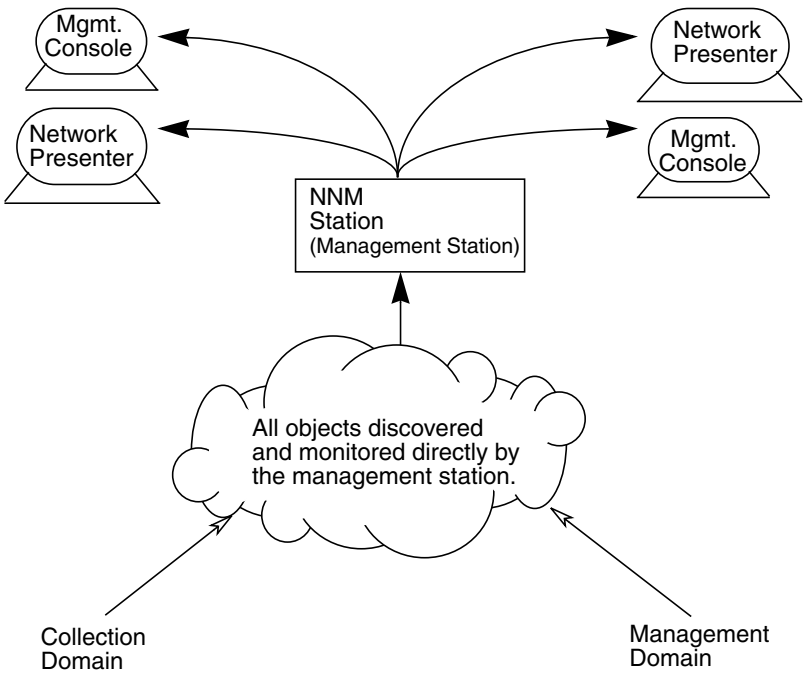
Fully Centralized Management

In a fully centralized network management scheme, one management station performs all monitoring and data collection functions for a whole management domain. There is little or no remote network management hardware or personnel. The primary focus is probably network management, with primary attention to the main connectors, like bridges, hubs, and routers.

Users of non-distributed network management products (including early versions of Network Node Manager) naturally use this model, since it is the only strongly supported model available. In fact, for many situations, a fully centralized usage of NNM is a highly functional and reasonable choice. Even in this model, scalability can be very useful.

Figure 3-1 illustrates a fully centralized network management scheme.

Figure 3-1 Fully Centralized Management Model



Notice that the collection domain and the management domain are identical. This is characteristic of this management model.

Remember that the management domain is an attribute of the management station, not the operators. You may have several operators, each of whom has a different view of the overall topology (via map filtering), and who therefore deal only with subsets of the management domain.

Implementation Tactics

Using a fully centralized model is probably the simplest (though not always the most efficient) way to use Network Node Manager. Depending on the nature of your situation, it might make sense to use one or more of the following scalability features in a fully centralized management situation:[3]

- Management consoles, for multiple operators and Network Presenter GUI for location independence.
- Map filtering, for specialized views of the topology.
- On-demand submaps, for better performance, especially on low-end management consoles.
- Discovery filtering, to eliminate unnecessary objects, enhancing performance.

  Consider an HP Service Guard cluster for the critical management station.

These features are especially useful if you have a large environment that you are managing centrally, and you have several specialized operators to monitor it. They are also useful features in a smaller management station, where you want to get the maximum performance from the available system.

3. See “Filters” on page 37, “On-Demand Submaps” on page 64, and “Management Consoles” on page 70.

Implications of Using the Fully Centralized Management Model

A centralized management model is simple to understand and control. Using a centralized model puts the management responsibilities (and expertise) in one location, which works well in some organizations. However, if you do not use the scalability features of NNM, performance of the map may become problematic in a very large environment.

If you have to monitor remote nodes over a WAN, a centralized approach causes unnecessary traffic on those expensive links. Furthermore, in a very large network, the overall performance of your network management solution is probably slowest if you use a fully centralized approach, and will require that you tune polling intervals so that the `netmon` service doesn't fall behind in polling.[4]

You may also want to take advantage of interface managed-state automation, as described in "Interface Managed-state Automation" on page 74.

For the most current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*.

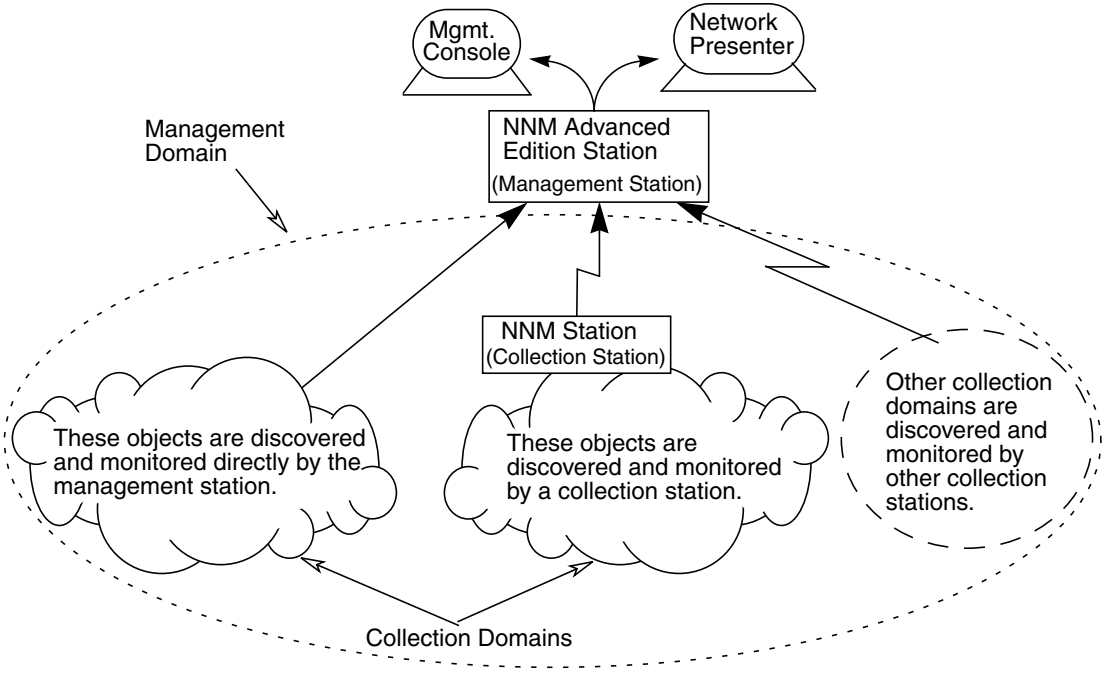
4. See the `Performance:Network Polling Statistics` menu item, and associated online help, for information on how to see if `netmon` is keeping up with polling.

Centralized-Hierarchical Management

Sometimes, one primary group has responsibility for a large network management environment. The centralized-hierarchical model involves a network that for some reason is not suitable for monitoring by a fully centralized model. The environment may be too large,[5] or you may have sizeable remote sites where network management support is minimal.

In Figure 3-2, all operators are located at the management station; this is the main characteristic of a centralized-hierarchical situation.

Figure 3-2 Centralized-Hierarchical Management Model



5. See “General Considerations about Scalability and Distribution” on page 84.

While the management station in the diagram has its own collection domain, it keeps a hierarchy of several remote collection stations that provide distributed network management services. Human support for network management at the remote sites is minimal and probably does not extend to operating NNM from a console.

Implementation Tactics

Implementing a centralized-hierarchical management model is, like all distributed models, more complex than a simple centralized approach.

All of the scalability features mentioned for the fully centralized model are still applicable in exactly the same way, and for the same purposes, as for the centralized-hierarchical model:[6]

- Management consoles, for multiple operators and Network Presenter GUI for location independence.
- Map filtering, for specialized views of the topology.
- On-demand submaps, for better performance, especially on low-end management consoles, or with large maps.
- Discovery filtering, to eliminate unnecessary objects, enhancing performance.

Consider HP Service Guard cluster for the critical management station.

Additionally, to implement this model you would consider the following tactics for the use of other scalability features:

- The collection station has no demand for topology data. Two filtering rules follow from this:
  - ❏ Discovery filters at the collection station should be set up to restrict discovery and monitoring to only those objects that are of interest to the central management station.
  - ❏ The collection station has no topology filters in place, because all the known topology (based on the discovery filters) is of interest to the central management station.
- Configure collection station failover so that the management station can pick up for critical collection stations.

6. See “Filters” on page 37, “On-Demand Submaps” on page 64, and “Management Consoles” on page 70.

- You will probably want to forward all threshold events that you configure for the collection station. The most likely forwarding would be to the `%REMOTE_MANAGERS_LIST%`, so that any interested manager would receive the events.

  One potential exception to this rule would occur if you only want a local action performed for a particular threshold event, and no higher level attention to the matter.

- If you are dealing with a large environment and using the centralized-hierarchical model, you may want the collection stations (even though essentially unmanned) to perform remote trend data collection. You would then plan to pull that data to the central management station for merging, report generation, and analysis.[7]

Implications of Using Centralized-Hierarchical Management

The centralized-hierarchical model is somewhat more complex to deploy and maintain than the fully centralized model, because each collection station requires special configuration. However, it has many advantages for environments that are either geographically dispersed, or too large to monitor from a single station.

You can manage networks with up to about 25,000 devices (assuming one interface per device, on average), and roughly fifteen operators. Performance of the map in a large environment can be improved by using map filtering and on-demand submaps.

A centralized-hierarchical model (like a fully centralized approach) puts the management responsibilities (and expertise) in one location, which works well in some organizations.

If you have to monitor remote nodes over a WAN, and still want centralized network management, a centralized-hierarchical approach can reduce traffic on expensive WAN links. The risk is slightly higher, since you are relying on the remote collection stations for data.

Finally, in a large network, the overall performance of your network management solution is probably better, because you are not performing polling over the lower performance WAN links.

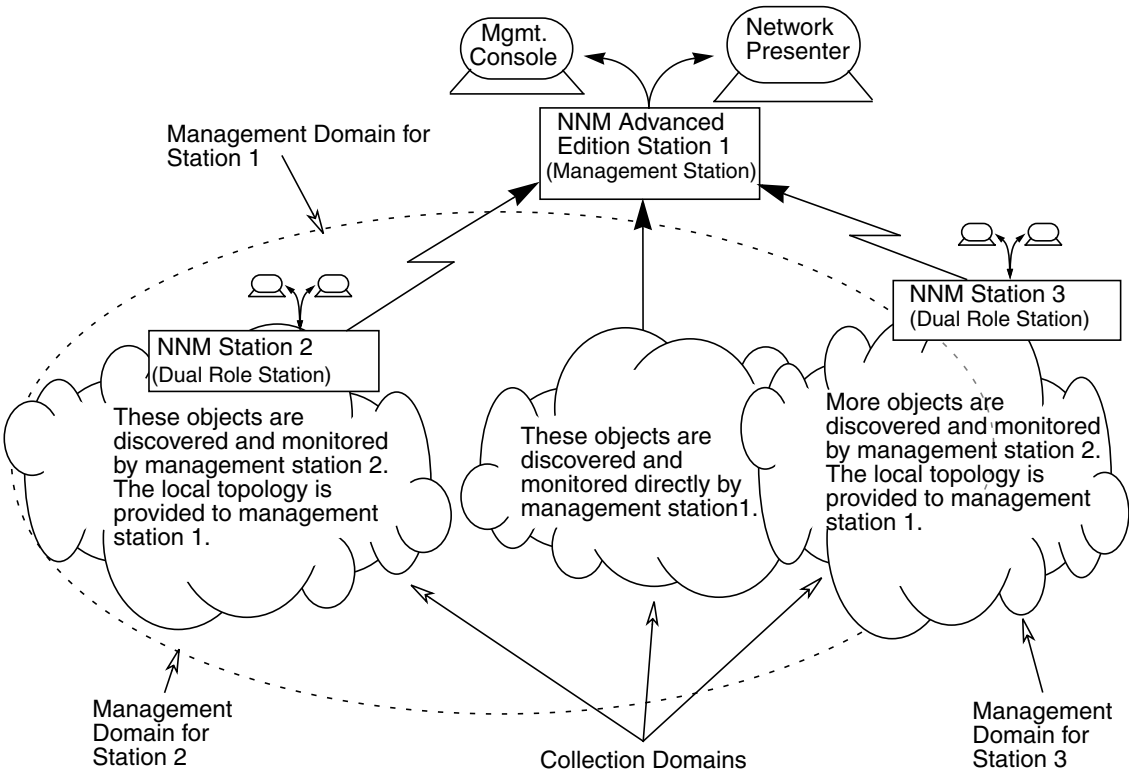
---

7. See the reference page in NNM online help (or the UNIX system manpage) for *ovcoltosql*.

Hierarchical Management

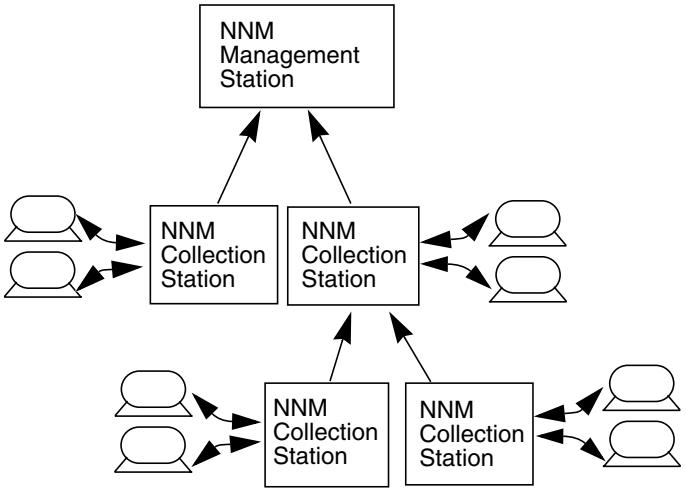
In many large organizations, a central group has broad responsibility for the network resources of the organization, usually for the major gateways and connecting infrastructure of the network (commonly called the network “backbone”). In the outlying sites, local network managers control the local configuration, usually including local connectivity, server availability, etc. This hierarchy of responsibility is reflected in the scheme illustrated in Figure 3-3, where NNM Stations are 2 and 3 are independent management stations, while also acting as collection stations for NNM Station 1.

Figure 3-3 Hierarchical Management Model



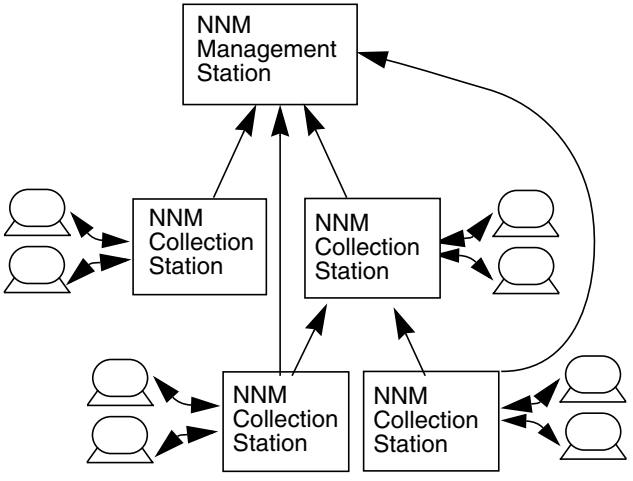
While multiple levels are possible in a hierarchical environment, Network Node Manager passes topology and status data only one level. That is, topology data received from a collection station can not be forwarded again. Only topology and status data that is derived from a local `netmon` service can be forwarded to remote managers.

Figure 3-4 Unsupported Multilevel Hierarchical Management



To monitor the multilevel hierarchy shown in Figure 3-4, you would have to arrange the collection stations as shown in Figure 3-5.

Figure 3-5 Single-tier Approach to a Multilevel Hierarchy



In Figure 3-5, the top-level management station uses both the first- *and* second-tier collection stations directly, to obtain all the information it needs.

Topology filters at the second-tier must pass all the objects needed by either first-tier or top-level management stations, but the same would be true even if multiple-level management were possible.

Implementation Tactics

Implementing a hierarchical management model is not much more complex than implementing a centralized-hierarchical approach.

All of the scalability features mentioned for the other models are applicable in the same way, and for the same reasons, in the hierarchical model:[8]

- Management consoles, for multiple operators and Network Presenter GUI for location independence.
- Map filtering, for specialized views of the topology.
- On-demand submaps, for better performance, especially on low-end management consoles, or with large maps.
- Discovery filtering, to eliminate unnecessary objects, enhancing performance.

  Consider HP Service Guard cluster for the critical management station and important collection stations.

Additionally, to implement the hierarchical model you would consider the following tactics for the use of other scalability features:

- Discovery filtering at the collection station should be set up to restrict IP discovery and monitoring to include only objects that are of interest to the operators at either the collection station or management station.
- Topology filtering should be implemented at the collection station, to expose only that part of topology that the management stations need. For example, the topology filter might pass only key connector nodes.

8. See “Filters” on page 37, “On-Demand Submaps” on page 64, and “Management Consoles” on page 70.

- The interests of operators at the management stations and the collection stations may overlap on some objects. Map filtering can be cooperatively designed and implemented between the management stations and the collection stations, so that any overlap is considered and deliberate.
- Configure collection station failover so that the management station can pick up for critical collection stations.
- You might want to forward some threshold events that you configure on the collection station, most probably to the `%REMOTE_MANAGERS_LIST%`, so that all interested managers receive those events.

  Or, you might want to duplicate a MIB variable via a MIB expression. This lets you set two different thresholds on the value, one that triggers local action and one that triggers attention at the management station.

  Of course, some thresholds might be of local interest only, and not be forwarded to the managers.
- If you are dealing with a large environment and using the hierarchical model, you may want collection stations to perform remote trend data collection.

  You would then plan to pull that data to the central management station for merging, report generation, and analysis. The administrators at the remote site may also have local use for the same trend data, or a subset of it.[9]

Implications of Using the Hierarchical Management Model

The hierarchical management model takes full advantage of the benefits that scalability offers. On the other hand, it can be complex to deploy and maintain in a large environment.

9. See “Trend Data Collection in a Distributed NNM Solution” on page 28 for more information.

Implementing a fully hierarchical management scheme makes it possible to manage larger environments than any other approach. Such environments are often geographically dispersed, too large to monitor effectively from a single site, and staffed, at least some of the time, with network management personnel at the remote sites.

Performance of the maps at any management station can always be improved through map filtering and using on-demand submaps. Your experiences will depend on the resources you have available, and the performance tuning you implement.[10] For the most current test results and product performance information see the latest version of the HP OpenView NNM *Performance and Configuration Guide*.

A hierarchical model (unlike any centralized approach) disperses the management responsibilities and expertise across the organization. Failures at any point in the system are less likely to have wide ranging consequences.

A hierarchical approach can significantly cut traffic on costly WAN links.

Finally, in a large network, the overall performance of your network management solution is probably better this way than with any other approach, in part because situations that develop in the collection domain of the collection stations can be generally handled by the local operators.

10. See “General Considerations about Scalability and Distribution” on page 84 for ideas.

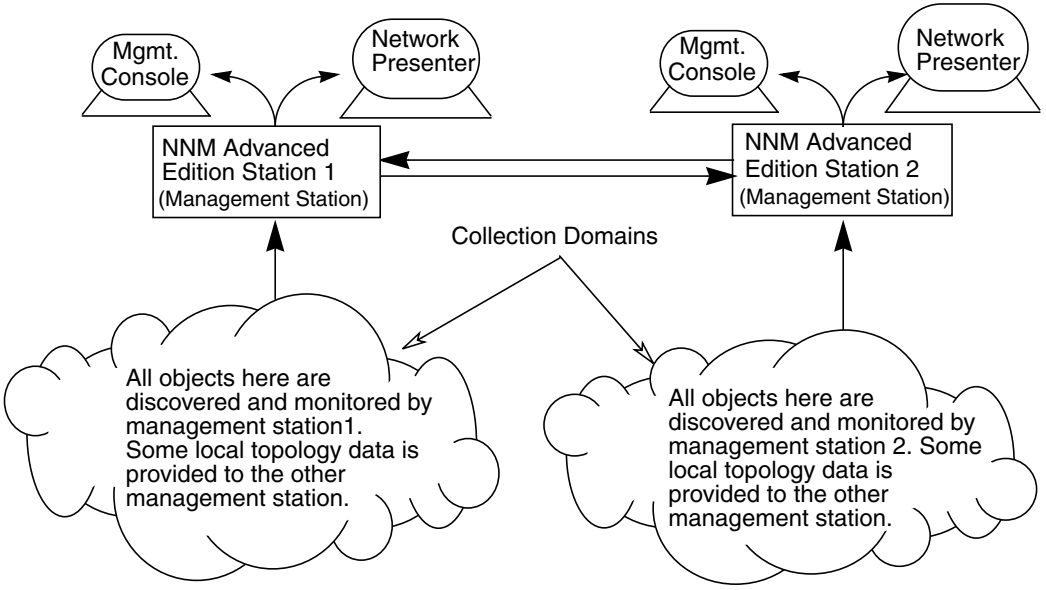
Cooperative Independent Management

In some situations, two largely independent sites need to share information at the boundaries of their management domains. Administrators at each site have primary responsibility for their part of the overall managed domain. This is shown in Figure 3-6.

In addition, these administrators are likely to have a strong interest in the major connecting nodes they share with other sites, in the primary servers at other sites, or in other key nodes or resources that reside in the domains of other sites.

This peer relationship may or may not reflect the organizational structure. You might choose this model because of the need for tight cooperation, even though one site is actually “higher” in the organizational hierarchy.

Figure 3-6 Cooperative Independent Management Model



Implementation Tactics

Implementing a cooperative management model is similar to implementing a hierarchical approach. In fact, cooperative management could be viewed as simply a variant of the hierarchical model, where each management station is, from the perspective of the other, just a subordinate collection station.

Again, all of the scalability features useful in the centralized model are still applicable in exactly the same way, and for exactly the same reasons, in the cooperative model:[11]

- Management consoles, for multiple operators and Network Presenter GUI for location independence.
- Map filtering, for specialized views of the topology.
- On-demand submaps, for better performance, especially on low-end management consoles, or with large maps.
- Discovery filtering at each management station can eliminate unnecessary objects, enhancing performance.

  Consider HP Service Guard cluster for critical management or collection stations.

Additionally, to implement the cooperative model you would consider the following tactics for the use of other scalability features:

- Discovery filtering at each management station should be set up to restrict IP discovery and monitoring to those objects that are of interest to the operators at that location.
- Topology filtering should be implemented at each management station, to expose only that part of topology that the other management stations need.
- You might want to forward some threshold events that you configure on the management stations, most probably to the `%REMOTE_MANAGERS_LIST%`, so that operators on both sides are informed. Other thresholds will be of local interest only, and not be forwarded to the managers.

11. See “Filters” on page 37, “On-Demand Submaps” on page 64, and “Management Consoles” on page 70.

- When using the cooperative independent model, you may want to collect trend data on the shared resources. Depending on the actual configuration of the resources, you may want one or all management stations to perform trend data collection. If multiple stations are collecting trend data, you might want to choose one to act as the central location for merging the data, report generation, and analysis. The administrators at each management site may also have local use for the same trend data.

Implications of Using the Cooperative Independent Model

The cooperative independent management model is very similar in technical complexity to a more purely hierarchical approach, and likewise requires that the sites cooperate in nontechnical ways as well. For example, scheduled downtime of critical or shared resources at one site should not come as a surprise to the other sites.

In implementing a cooperative independent management scheme, you can assume that each site has complete responsibility for local management of non-shared resources and objects.

A management station with a cooperative relationship to another management station can, at the same time, have a hierarchical relationship with one or more other collection stations.

The cooperative model acknowledges that management responsibilities and expertise are not centralized in the organization, and that unique skills and requirements exist at different sites. While a failure at one point in the system is unlikely to have wide ranging consequences, a local problem may get little attention from a remote site.

Finally, the overall performance of your network management solution is good, in part because situations that develop in the collection domains are generally handled locally.[12] Thus, a cooperative approach provides benefits without creating significant traffic on costly WAN links.

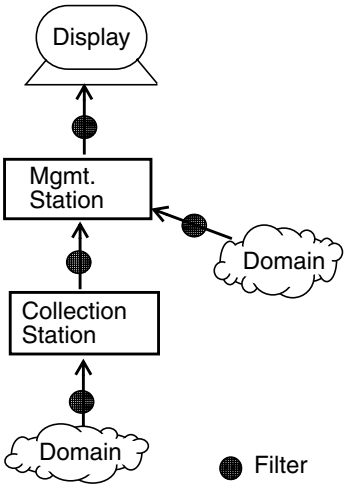
12. See “Filters” on page 37, “On-Demand Submaps” on page 64, and “Management Consoles” on page 70.

4 Procedures for Scaling NNM

This chapter is not an introduction to NNM procedures. Readers are assumed to be conversant with NNM's key features for network monitoring, data collection, event handling, and so on. Readers who are looking for that information should turn to *Managing Your Network with HP OpenView Network Node Manager* instead. That manual also covers basic configuration of NNM station options that are not exclusive to distributed management.

This chapter covers procedures for scaling NNM. These procedures are:

- Configuring management consoles.
- Configuring on-demand submaps.
- Configuring distributed internet monitoring.
  - — Data collection
  - — Collection station
  - — Management station
  - — Domain overlap and failover
- Configuring event forwarding.
- Using distributed data collection.



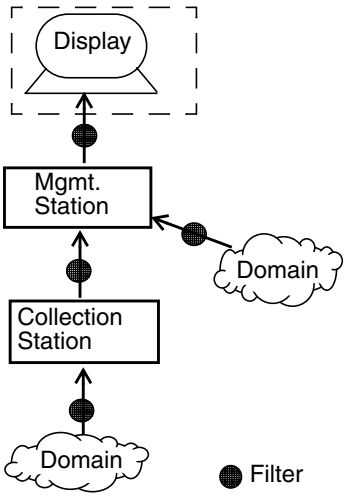
The accompanying diagram serves as your roadmap through this chapter. As new areas of configuration are introduced, they are circled on the diagram.

All of the documentation for HP OpenView Windows and Network Node Manager is written with the assumption that you have activated environment variables on UNIX systems. For information about environment variables, see the reference page in NNM online help (manpage on UNIX systems) for *ov.envvars*.

NOTE Refer to the Release Notes, Supported Configurations topic for the operating systems that are supported with this release.

Configuring Management Consoles

Distribution of the HP OpenView Windows Graphical User Interface (GUI) is done via NFS mounting (Windows operating system file sharing for Windows operating system servers) of the map database and the appropriate configuration files, and by using the distributed RPC capability of the object database service (or background process on UNIX systems), `ovwdb`.



This section has five components:

- Configuring an NNM server on a machine running the Windows operating system (Windows operating system file sharing).
- Configuring a client on a Windows system for server on a Windows system (Windows operating system file sharing).
- Configuring a server on a UNIX operating system.
- Configuring a client on a Windows system to use a server on a UNIX server (NFS).
- Configuring a client on a UNIX system.

NOTE The management console (client) and the server must run the same revision of NNM. Refer to the Release Notes, Supported Configurations topic for the operating systems that are supported with this release.

To set up management consoles, the following must be true:

- A management server on UNIX must be configured as an NFS server and the management console (client) must be configured as an NFS client.
- You must have `root` access on UNIX system servers and UNIX system clients. Windows systems require `Administrator` rights.

- The client must have read-write access to a UNIX system server's file system via NFS.
- User and group IDs must be synchronized between the client and server.

The basic steps in installing management consoles are:

1. Install NNM on the server system, following the instructions in the *HP OpenView Network Node Manager Installation Quick Start*. (You do not need an additional license for the client.)
2. Mount the server's HP OpenView data directories on the client system.
3. Install NNM on the client system.
4. For UNIX system clients, run the `ovwsetupclient` configuration script on the client to have it use the mounted databases for maps and objects.
5. Run `ovstart` on the server system.
6. Run `ovw` on the client system.

CAUTION

If a "lightweight" system (for example, a HP9000/715 with 64 Mbytes of RAM) is configured as a management console client for a server with a large database, it is possible that the resulting performance of the client system will be unacceptably slow, or that one or more of the background processes that run on the management console will run out of memory.

This is because `ipmap` and the `ovw` process sizes grow in proportion to the number of objects that are associated with persistent maps. For example, suppose that management station has 40,000 objects in its default map and the on-demand submap feature is disabled. If a lightweight management console tries to open this map, the synchronization process on the management console will cause `ovw` and `ipmap` to grow so large that either performance will be exceedingly slow, or one of the processes will run out of memory and fail.

The solution to this problem is to reduce the number of objects that are associated with persistent maps at the management console. This can be done by using the on-demand submaps feature, or map filters, or both. If the map is to be shared by both the management station and the console, the demand level should be raised at the *management* station *prior* to starting the management console. Alternatively, a copy of the map can be

created at the management station, and that copy can be configured with a higher demand level and/or filters. This new map can then be started up by the management console. For the best performance at the management console, set the demand level to Internet.

Configuring a Windows System Console Server

1. Share the *install_dir* directory from the NNM management server as *ShareName*. You may use Windows Explorer using a right mouse click. Give the users `Full Control` access.
2. Start the NNM services from the Control Panel at the NNM management server. These services must run during installation, or installation will fail.

Installing a Management Console on Windows for an NNM Server on Windows

First, install NNM on the Windows operating system machine that will be the management server, then follow these steps. You will need to know the name of the server to which the console will connect and the *ShareName* on the server.

1. On the system to be an NNM remote console, verify that you can connect to the *ShareName* share. You may use Windows Explorer.

NOTE

During installation, the *install_dir*\conf*.auth files are edited to give the remote console access to the manager's database and services. If you experience problems connecting to a management server, you may want to verify the rights granted in these files. You may also edit these files to customize rights for remote console operators.

2. On the NNM remote console system, insert the CD-ROM and run `setup.exe`. When installing, pick the `[Remote Console Installation]`. This removes any databases or local configuration files (these files might be in place if you had previously installed the full NNM management server). Supply the machine name of the NNM management server and share name *ShareName*.

3. You can now run NNM from this remote console as if you were at a full management server. The remote console has no icons for `[ovstop]` and `[ovstart]`, since to use these from a remote console, you must explicitly name what to start-up or shutdown. However, you can run these services from the command line.

Remote console operator rights to view maps are controlled the same way as operators viewing maps directly on the management server. Use NTFS file system access assignments to the *`install_dir`*`\databases\openview\mapdb` directory and subdirectories. Refer to *Managing Your Network with HP OpenView Network Node Manager* for details.

UNIX System Server Configuration

The server system is the management station on which you installed NNM and whose map and object database you want to export to the client system. You must be `root` to perform these steps. In the following steps, replace *`client_name`* with the hostname of the client from which you want to use these databases.

The following steps allow all users, including `root`, to run `ovw` sessions on the remote client.

1. Install NNM on the server as described in the *HP OpenView Network Node Manager Installation Quick Start* for the system.
2. If management consoles are connecting to this server from a Windows operating system, verify that the `pcnfsd` process is running on the server. This process lets your Windows operating system consoles know the name of the management station and that the station is running the `pcnfsd` daemons.

NOTE

This step explains how to configure the `pcnfsd` process to authenticate NFS access requests from Windows clients. Although the `pcnfsd` process is commonly used for authentication, third-party providers of Windows NFS client products offer alternative methods, such as NIS authentication. If you prefer to use the `pcnfsd` process, complete this step. If you skip this step and use a third-party authentication procedure, it must be configured to allow access to user IDs in the range of 0-60002.

a. Create the file `/etc/pcnfsd.conf`. It will contain one line that reads:

   ```
   uidrange 0-60002
   ```

   Save and close this file.

b. Start PCNFS.

   - On HP-UX:

     Edit the `/etc/rc.config.d/nfsconf` file. In it, change the PCNFS_SERVER= line to look like this:

     ```
     PCNFS_SERVER=1
     ```

     Save and close this file. To start the service, run the command:

     `/sbin/init.d/nfs.server start`

   - On Solaris:

     Verify that the PCNFS product is installed on the management station and that `pcnfsd` is running:

     `ps -ef | grep pcnfsd`

     If it is not running, start it with `/opt/SUNW/pcnfs/scripts/pcnfs start`.

     Run this command:

     `ps -ef | grep -e "statd|lockd"`

     If you do not see both the `/usr/lib/nfs/statd` and the `/usr/lib/nfs/lockd` processes running, execute:

     `/etc/init.d/nfs.client start`

3. Ensure that NFS is running on the server.

   - **On HP-UX:**

     a. Run this command:

        `ps -ef | grep nfsd`

     b. If the `nfsd` process is not running, edit the file `/etc/rc.config.d/nfsconf` and change `NFS_SERVER=0` to `NFS_SERVER=1` and run the following command:

        `/sbin/init.d/nfs.server start`

- **On Solaris** (NFS must already be installed):

  a. Run this command:

  ```
  ps -ef | grep nfsd
  ```

  b. If the `nfsd` process is not running, execute:

  ```
  /etc/init.d/nfs.server start
  ```

- **On Linux:**

  a. Run this command:

  ```
  /etc/rc.d/init.d/nfs status
  ```

  b. If NFS is not running, start NFS:

  ```
  /etc/rc.d/init.d/nfs start
  ```

4. Configure the exportation of the HP OpenView file system to the client as follows:

   - For an HP-UX server, edit the `/etc/exports` file. You will either modify two existing lines or add new ones, depending on the contents of this file.

     -- If the line `/etc/opt/OV/share -root=` is present, add your console system's name after the equal sign. If there is already a system name after the equal sign, then add a colon (:) followed by your console system's name after the first system name.

     -- If the line `/etc/opt/OV/share -root=` is not present, add it and include your console system's name.

     -- Do the same for the line `/var/opt/OV/share -root=`.

     When you have finished editing, the file should look like this:

     ```
     /etc/opt/OV/share -root=First_System:Console_System
     /var/opt/OV/share -root=First_System:Console_System
     ```

     or

     ```
     /etc/opt/OV/share -root=Console_System
     /var/opt/OV/share -root=Console_System
     ```

     Save and close the `/etc/exports` file.

   - For a Solaris server, edit the `/etc/dfs/dfstab` file by adding the following two lines to it:

```
share -F nfs -o root=Console_System /etc/opt/OV/share
share -F nfs -o root=Console_System /var/opt/OV/share
```

Save and close the `/etc/dfs/dfstab` file.

- For a Linux server, edit the `/etc/exports` file. Depending on the contents of the file, either modify or add two lines:

```
/etc/opt/OV/share <IP_address_of_console>
(rw,no_root_squash)
/var/opt/OV/share <IP_address_of_console>
(rw,no_root_squash)
```

Save and close the `/etc/exports` file.

5. Run the following two commands to unexport and re-export the directories from the server:

On HP-UX:

```
/usr/sbin/exportfs  -au
/usr/sbin/exportfs  -av
```

On Solaris:

```
share -F nfs -o root=client_name /var/opt/OV/share
share -F nfs -o root=client_name /etc/opt/OV/share
```

Use the fully qualified domain name when specifying the *client_name*.

On Linux:

```
exportfs  -ra
```

6. Verify that NNM is running on the management station. Type

```
/opt/OV/bin/ovstatus | more
```

You should see a list of processes that are currently running. Check the list to verify that the `ovwdb` process is running. If `ovwdb` is not, type

```
/opt/OV/bin/ovstart ovwdb
```

Then, type **`/opt/OV/bin/ovstatus | more`** again to verify that `ovwdb` is now running.

Installing Management Consoles on Windows for UNIX System Management Servers

To set up a remote management console on a Windows operating system to use a UNIX management station server, first satisfy the following prerequisites:

- Make sure that the management station server is not already mapped as a network drive on your Windows operating system console. If it is, remove the network drive connection.
- Verify that you have a compatible NFS client product installed on your Windows operating system console. (See the NNM Release Notes for a list of NFS client products that have been tested with NNM.)
- You will need to know the name of the server to which the console will connect and the two directories exported in step 4 and 5 of “UNIX System Server Configuration” on page 112.

Once the prerequisites are satisfied, perform the following steps:

1. Verify or configure the following settings in the NFS client software on the system running Windows that will be a management console.
   - The name of the management station you are connecting your management console to. Indicate the station by name, not by IP address.
   - The root login name and password for connecting to the management station.
   - The file locking feature, which must be enabled.
   - The file protection, which must be configured to “`rwxrwxrwx`” for each NFS mount. You can reconfigure file protection through the GUI of your NFS client product. Changing file permission this way allows any created files (such as map databases) to be available to all users.
   - The filename option, which must be set to Preserve Case.
   - The NFS protocol version, which must be version 2.
2. On the management console machine, mount the two directories exported in step 4 and 5 of “UNIX System Server Configuration” on page 112.

3. On the management console machine, install NNM.
   a. Insert the NNM product CD-ROM into the management console system's drive.
   b. Start the installation wizard program. You will see various dialog boxes that explain the licensing agreement and that ask you to enter your name and company.
   c. A Setup Options dialog box appears, prompting you about the kind of NNM installation you want to do. Select the option, "Remote Console".
   d. A Windows Remote Console dialog box appears, asking you to specify the operating system being used by the management station.
   e. The next dialog box asks for two full paths to certain file systems on the management station. These file systems contain primary HP OpenView files and application files.
      - The first path requested is the one to `/etc/opt/OV/share`. A default value of `G:\` is provided, but you may need to modify this.
      - The second path requested is the one to `/var/opt/OV/share`. A default value of `H:\` is provided, but you may also need to modify this.
4. To complete the NNM installation, follow the rest of the steps in the installation wizard program.
5. Once installation is completed, it is recommended that you change the configuration of your PCNFS server product so that the authentication is that of the user using NNM on the Windows operating system console, instead of `root`. For example, if a user called `ovuser` is running NNM on both the management station and the Windows operating system station, the configured PCNFS authentication should be `ovuser` instead of `root`. By changing the authentication, you ensure that any files generated during an NNM session will be accessible from both the station and the Windows operating system console.

UNIX System Console Configuration

Before you begin, you will need to know the name of the server to which the console will connect. Perform the following steps on each client machine:

1. Install NNM on the console as described in the *HP OpenView Network Node Manager Installation Quick Start* for the system.
2. Ensure that the console is running as an NFS client.
   - **On HP-UX:**
     a. Run this command:

        `ps -ef | grep biod`

     b. If the `biod` process is not running, edit the file `/etc/rc.config.d/nfsconf` and change `NFS_CLIENT=0` to `NFS_CLIENT=1` and run the following command:

        `/sbin/init.d/nfs.client start`
   - **On Solaris**:
     a. Run this command:

        `ps -ef | grep -e "statd|lockd"`

     b. If you do not see both the `/usr/lib/nfs/statd` and the `/usr/lib/nfs/lockd` processes running, execute:

        `/etc/init.d/nfs.client start`
   - **On Linux:**
     a. Run this command:

        `/etc/rc.d/init.d/nfs status`

     b. If NFS is not running, start NFS:

        `/etc/rc.d/init.d/nfs start`
3. Choose a mount point for the server's HP OpenView file systems. To make sure this file system is mounted at boot time, you must modify the appropriate file shown below. You must be `root` to modify these files. Replace *`server_name`* with the name of the server whose databases you want to use on this client.

NOTE As an alternative to these steps, NFS access may be provided via the NFS automounter. Consult your operating system's manual for details.

- On **HP-UX:**
  a. Run these commands:

     mkdir -p /opt/OV/nfs/*server***/var/opt/OV/share**

     mkdir -p /opt/OV/nfs/*server***/etc/opt/OV/share**

  b. Add these lines to the `/etc/fstab` file:

```
server:/var/opt/OV/share /opt/OV/nfs/server/var/opt/OV/share nfs rw 0 0
server:/etc/opt/OV/share /opt/OV/nfs/server/etc/opt/OV/share nfs rw 0 0
```

  c. Run these commands:

     /etc/mount s*erver***:/etc/opt/OV/share**

     /etc/mount *server***:/var/opt/OV/share**

- **On Solaris**:
  a. Run these commands:

     mkdir -p /opt/OV/nfs/*server***/var/opt/OV/share**

     mkdir -p /opt/OV/nfs/*server***/etc/opt/OV/share**

  b. Add these lines to the `/etc/vfstab` file:

```
server:/var/opt/OV/share - /opt/OV/nfs/server/var/opt/OV/share/ nfs - yes rw
server:/etc/opt/OV/share - /opt/OV/nfs/server/etc/opt/OV/share/ nfs - yes rw
```

  c. Run these commands:

     /etc/mount *server***:/etc/opt/OV/share**

     /etc/mount *server***:/var/opt/OV/share**

- On **Linux:**
  a. Run these commands:

     mkdir -p /opt/OV/nfs/*server***/var/opt/OV/share**

mkdir -p /opt/OV/nfs/*server***/etc/opt/OV/share**

b. Run these commands:

mount *server***:/etc/opt/OV/share**
/opt/OV/nfs/*server***/etc/opt/OV/share**

mount *server***:/var/opt/OV/share**
/opt/OV/nfs/*server***/var/opt/OV/share**

4. Set up the client by running the following script:

$OV_BIN/ovwsetupclient /opt/OV/nfs/*server*

NOTE If you have used the NFS automounter, the path may be different.

5. To verify that the configuration is correct, run the following command:

ovw -server

Look at the return value from the `ovw -server` command on your display. It should look similar to:

```
%ovw -server
your server name shows up here
```

6. Run **ovw** on the management console.

If you are running on an HP Service Guard cluster, you may run **ovwrs** rather than **ovw**, if NNM is configured as a package. Refer to the *ovwrs* man page for more information about restartable **ovw**.

Increasing Security for Management Consoles

Authorization files on the server allow you to limit which consoles may access the information on the server. If you have modified default authorizations, add the server hostname and each client hostname to the authorization file for `ovwdb` and `ovw`. Be sure to add the line *client_name* `+` for each client you are giving access to the server. The files `ovwdb.auth` and `ovw.auth` are located in `$OV_CONF`.

```
server_name +
client_name_1 +
```

client_name_2 +

where *server_name* is the name of the system where your map database resides. The addition of these lines enables any user from *client_name_1* or *client_name_2* to use the database on *server_name*.

NOTE A third authorization file also exists, `ovspmd.auth`. This file limits which consoles have permission to run `ovstop` and `ovstart`. **You should be careful about giving too many users this permission.**

Undoing a Management Console

If you want to undo the management console setup and run NNM in a normal manner, you must follow these steps in the order listed.

CAUTION Failure to follow these steps could result in damage to your databases.

- On UNIX systems:
  1. Run the following command on the console:

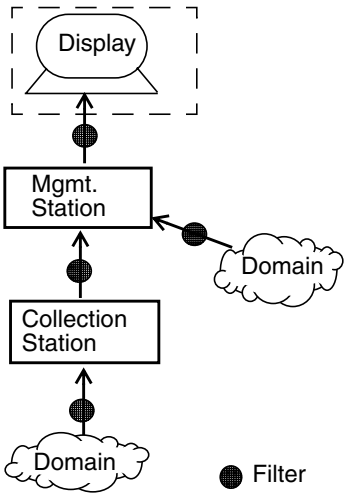
     `$OV_BIN/ovwsetupclient -u`

  2. Remove the lines that were added to the `/etc` files in the previous Server Configuration and Client Configuration sections.
- On Windows systems:
  1. Reinstall NNM from the installation CD.

Configuring On-Demand Submaps

This section discusses how to:

- Configure on-demand submaps.
- Specify a persistence filter.
- Integrate applications with on-demand submaps.



Configuring Demand Level

You can set the demand level of maps as you create them or after you have already created the map. You can set the submap demand level using the `-D` option on the `ipmap` command or using the `IP Map Configuration` dialog box. See the reference page in NNM online help (or manpage on UNIX systems) for *ipmap* for information setting the demand level using the command line.

To set the demand level as you are creating the map, use the following procedure:

1. Select `Map:New...` from the menu bar; the `New Map` wizard (dialog box on UNIX systems) appears.
2. Select `IP Map` from the `Configurable Applications` list. Figure 4-1 illustrates the list under the Windows operating system.
3. Click `[Configure For This Map...]`; the `IP Map Configuration` dialog box differs according to the operating system, as shown in Figure 4-2 and Figure 4-3.
4. On UNIX systems, in the `IP Map Configuration` dialog box, use the scroll bar in the top pane to scroll down to the `------ On-Demand Specification -----` section of the pane.
5. Use the option button under the question "`To what level should submaps be persistent?`" to specify the demand level. You can specify one of the following:

`Unset`

Resets the demand level to the platform default. On systems running the Windows operating system, this is `Internet Level`. On UNIX systems, it is `All Levels`.

`All Levels`

Specifies all submaps as persistent. This is the same as specifying level 1 when using the `ipmap -D` command.

`Segment Level and Higher`

Specifies that segment, network, and Internet level submaps are persistent. This is the same as specifying level 2 when using the `ipmap -D` command.

`Network Level and Higher`

Specifies that network and Internet level submaps are persistent. This is the same as specifying level 3 when using the `ipmap -D` command.

`Internet Level`

Specifies that Internet level submaps are persistent. This is the same as specifying level 4 when using the `ipmap -D` command.

6. Click `[Verify]` to verify that the values are valid.
7. Click `[OK]` to close the `IP Map Configuration` dialog box.
8. On systems running the Windows operating system, click `[Next]` to continue filling in the wizard windows. On UNIX systems, click `[OK]` from the `New Map` dialog box after you have completed the required fields.

You can use the `Map:Properties` menu from the menu bar to change the demand level after the map has already been created.

Click the `[Applications]` tab in the `Map:Properties` dialog box on systems running the Windows operating system. Using the `Map Properties` dialog box, follow the above procedure for setting the demand level of a new map. You must have read-write access to the map in order to change its demand level.

Refer to “On-Demand Submaps” on page 64 for more information about the on-demand submap feature.

Figure 4-1 Windows Map Properties Property Sheet, Applications Tab

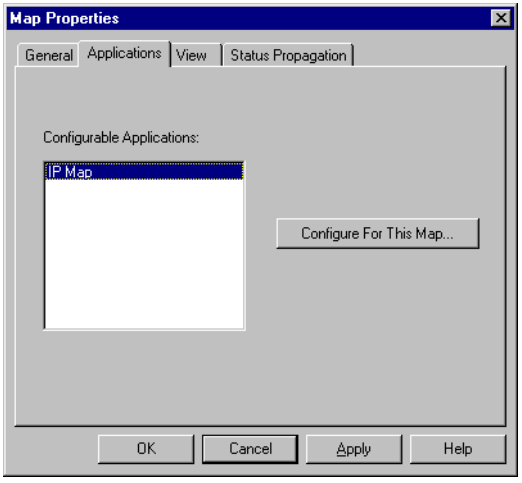


Figure 4-2 Windows Map Configuration Dialog Box

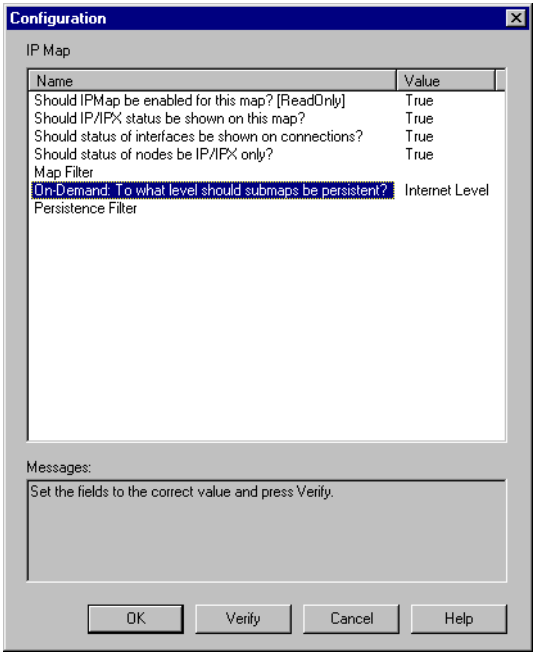
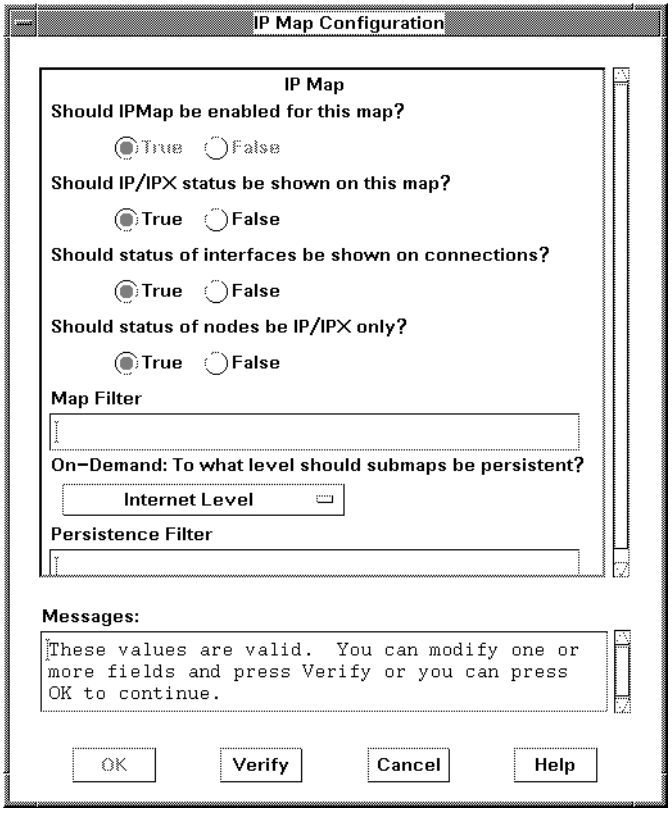


Figure 4-3 IP Map Configuration Dialog Box on UNIX Systems



Specifying a Persistence Filter

To specify a persistence filter for a new map, use the following procedure:

1. Create a filter in the file *install_dir*\conf\C\filters (`$OV_CONF/$LANG/filters` on UNIX systems). You can either use a predefined filter or create a new filter. Refer to Appendix A, “The Filter Definition Language,” on page 169 for information on filters.
2. Select `Map:New` from the menu bar; the `New Map` wizard (dialog box on UNIX systems) appears.
3. Select `IP Map` from the `Configurable Applications` list (in the second wizard window on systems running the Windows operating system, as shown in Figure 4-1).

4. Click [Configure For This Map...]; the IP Map Configuration dialog box appears, as shown in Figure 4-2 and Figure 4-3.
5. On Windows systems, select Persistence Filter. The Persistence Filter dialog box opens.

   On UNIX systems, in the IP Map Configuration dialog box, use the scroll bar in the top pane to scroll down to the ------ On-Demand Specification----- section of the pane.
6. In the Persistence Filter text field, enter a filter name or expression.
7. Click [Verify] to verify that the values are valid.
8. Click [OK] to close the IP Map Configuration dialog box.
9. On Windows systems, click [Next] to continue filling in the wizard windows.

   On UNIX systems, click [OK] from the New Map dialog box after you have completed the required fields.

You can use Map:Properties from the menu bar to change the persistence filter after the map has already been created.

Using [Applications] tab in the Map:Properties dialog box, as shown in Figure 4-1 on page 124, (or the Map Description dialog box on UNIX systems), follow the above procedure for setting the persistence filter of a new map. You must have read-write access to the map in order to specify a persistence filter.

Refer to “Persistence Filtering” on page 68 for more information about persistence filters.

Integrating Applications With On-Demand Submaps

Before integrating a new application with NNM, you should have a good understanding of how your application works so your application can produce the same results whether you are using the on-demand submap feature of NNM or not. This may require a persistence filter to use when on-demand submaps are enabled.

For example, if your application adds a hub symbol to a segment submap and needs the status of the hubs to propagate up the submap hierarchy, then you need to specify a persistence filter.

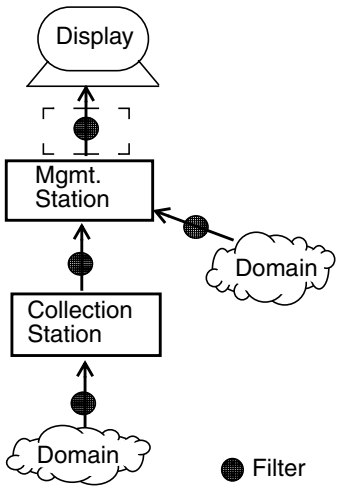
Another example of when a persistence filter is needed is if your application, at start-up, queries the map for all objects of a certain type and performs an operation for each of those objects found. Your application needs to ensure that the objects of this type are on the map by including a persistence filter that specifies this object type.

Use the following procedure to create and specify a persistence filter.

1. Determine what objects your application requires in the map.
2. Start your application running on NNM.
3. Select an object that your application has added and that you want to remain persistent in the submap.
4. Select `Edit:Object Properties` from the menu bar.
5. From the `Object Description` dialog box, view the object attributes for `Capabilities`.
6. Determine which capabilities are unique to that object type, for example, which are set to true.
7. With this information you can create a filter that will allow this object type to pass through.
8. Follow the steps for configuring the demand level and specifying a persistence filter.

Configuring a Map Filter

A map filter allows the operator to display only the objects of interest to that operator. Objects that are rejected by a map filter remain in the management station's topology database and are still subject to status polling. Use the following procedure to configure a map filter on a management station:



1. Create a filter in the file *install_dir*\conf\C\filters (`$OV_CONF/$LANG/filters` on UNIX systems). You can either use a predefined filter or create a new filter. Refer to Appendix A, "The Filter Definition Language," on page 169 for information on filters.
2. On systems running the Windows operating system, select the `Applications` tab in the `Map:Properties` dialog box.

   On UNIX systems, select `Map:Properties` from the menu bar. The `Map Description` dialog box appears.
3. Select `IP Map` from the `Configurable Applications` list and click `[Configure For This Map]`. The `IP Map Configuration` dialog box appears, as shown in Figure 4-2 on page 124 (Figure 4-3 for UNIX systems).
4. You must have read/write rights to the current map.

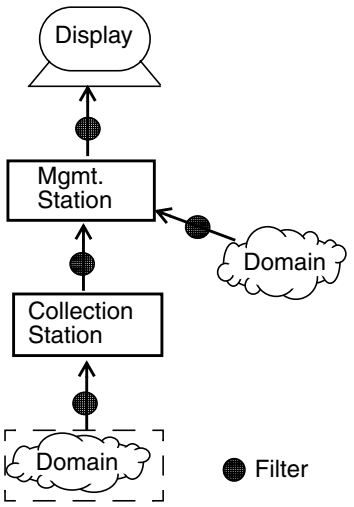
   On systems running the Windows operating system, select `Map Filter`, and enter the filter name.

   On UNIX systems, enter the filter name that you created in the text box under `Map Filter`.
5. Click `[Verify]` to make sure the filter name or expression you specified is a valid filter.
6. Click `[OK]` to save the changes and close the dialog box.

Configuring Data Collection for a Station

This section describes how to perform operations required on all collection and management stations:

- Configure a discovery filter.
- Manage and unmanage a locally monitored object.



Configuring a Discovery Filter

A discovery filter reduces the number of devices a management station or collection station is actively discovering and monitoring. A discovery filter can be configured on a management station or a collection station. Use the following procedure to configure a discovery filter:

1. Create a filter in the file *install_dir*\conf\C\filters (`$OV_CONF/$LANG/filters` on UNIX systems). You can either use a predefined filter or create a new filter. Refer to Appendix A, "The Filter Definition Language," on page 169 for information on filters.
2. Select `Options:Network Polling Configuration:IP/IPX` (`Options:Network Polling Configuration:IP` on UNIX systems) from the menu bar. The dialog box tabs under the Windows operating system are shown in Figure 4-4, Figure 4-5, and Figure 4-6. The `Network Polling Configuration` dialog box for UNIX systems appears as shown in Figure 4-7.
3. Make sure the `Discover New Nodes` option button is on in the IP and IPX tabs, as needed. (There is no IPX on UNIX systems.)
4. On Windows systems, click the `[Use Filter]` option button on the `[General]` tab and choose a filter from the list box.

   On UNIX systems, click the `Use Discovery Filter` option button to the on position. The corresponding text box will ungray.
5. Enter the filter name as defined in the filters file into this text box.

6. Click [OK] to save the changes and close the dialog box.

The discovery filter will be applied to all newly discovered objects. Polling is still done on all previously discovered objects whether or not they would have passed the filter.

To apply the filter to previously discovered objects you need to perform the following steps (from a command window on the Windows operating system):

1. Stop netmon by typing

   ovstop netmon

2. Once netmon is stopped, execute the following command:

   ovtopofix -f *filtername*

3. Then start netmon by typing:

   ovstart netmon

Figure 4-4 Windows Network Polling Configuration, General Tab

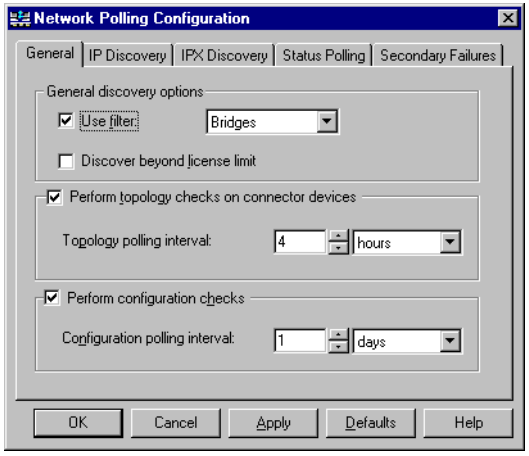


Figure 4-5 Windows Network Polling Configuration, IP Discovery Tab

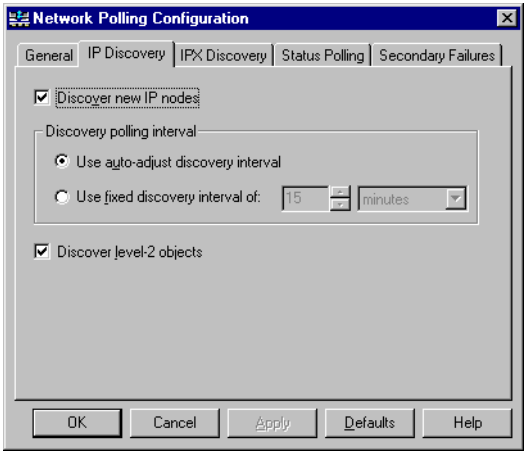


Figure 4-6 Windows Network Polling Configuration, IPX Discovery Tab

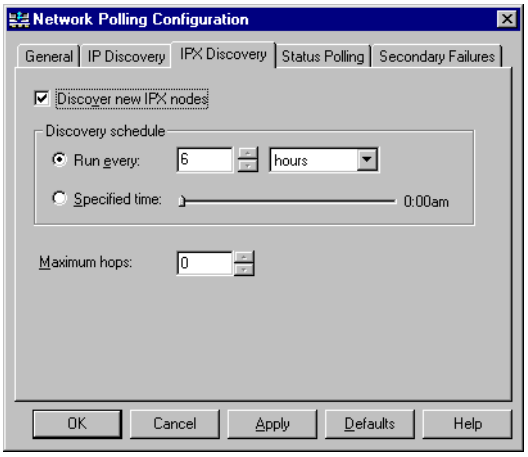
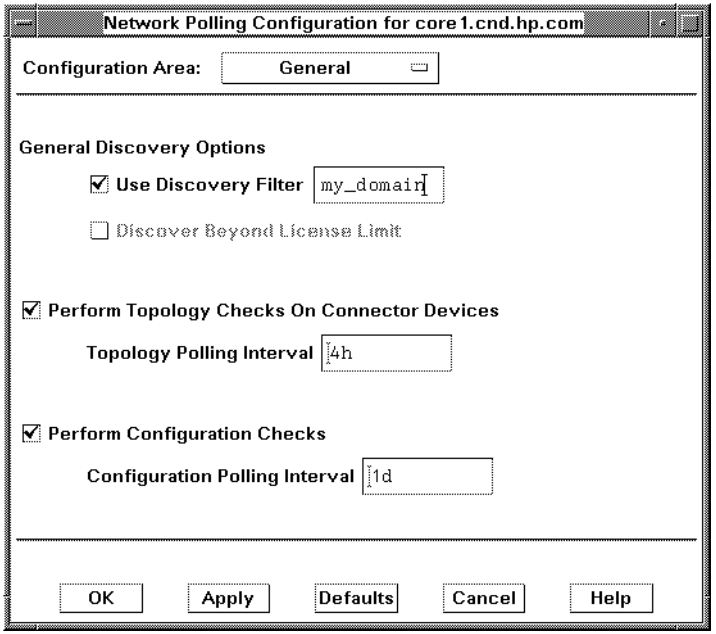


Figure 4-7 UNIX System Network Polling Configuration Dialog Box



Managing and Unmanaging Locally Monitored Objects

The managed state of a version of an object must be set on the collection station monitoring the object version. For example, in the examples shown in “Determining the Primary Collection Station for an Object” on page 151, to make `gw1` unmanaged on the management station, the node must be unmanaged at that collection station (`remstn1`). `remstn1` will detect the change in managed state and reflect that in the management station’s topology data. For locally monitored objects, it may be desirable to change the managed state of the object. If the local management station is the primary station for an object, this can be accomplished through the normal map editing operations. For non-primary versions of locally monitored objects, the managed state can be changed by specifying the `-manageobj` and `-unmanageobj` options of `xnmtopoconf`:

`xnmtopoconf [-manageobj | -unmanageobj]` *`station object`*

For example, to manage the locally monitored version of node `gw1` execute the following command (from a command window on systems running the Windows operating system):

```
xnmtopoconf -manageobj local gw1
```

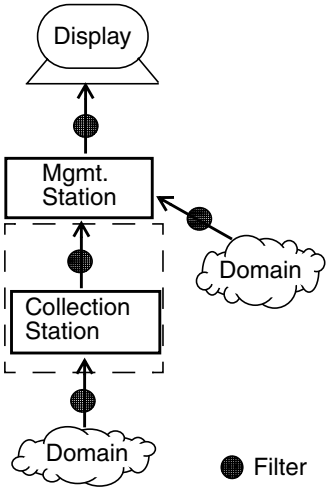
NOTE

If the overlap mode for the local station is set to `Unmanage` or `Delete`, the system may change the managed state of the object during a transition between different primary stations for the object. To make your changes permanent, set the overlap mode to `Allow` and manually set the managed state of all networks and gateways using the `-manageobj` and `-unmanageobj` options of `xnmtopoconf`.

Configuring Collection Stations

This section describes how to:

- Configure security for a collection station.
- Configure a topology filter for a collection station.
- Change the topology filter a collection station.
- Configure a Windowsoperating system IPX collection station.



Configuring Security for a Collection Station

On the collection station, you need to configure SNMP security to allow the exporting of topology information. Use the following procedure to do this (on UNIX systems, you must be `root` to execute this procedure).

1. Make sure NNM is not running. If NNM is running,

   Windows — Stop all NNM services by clicking the `NNM Services - Stop` icon.

   UNIX — Exit all sessions and issue the command **`ovstop`**

2. Make sure the SNMP master agent is not running by executing the following command:

   Windows — Stop `SNMP EMANATE Adapter for Windows` and `SNMP EMANATE Master Agent` using the `Control Panel Services` applet.

   UNIX — **`/usr/sbin/snmpd -k`**

3. Edit the set community name line in the *`install_dir`*`\conf\SnmpAgent\snmpd.conf` (`/etc/SnmpAgent.d/snmpd.conf` on UNIX systems) file to read as follows:

`set-community-name: name #enter community name`

where, *name* is the community name of your choice. By default, NNM does not install a default set community name. You must set this value before attempting to manage a station.

4. Restart the SNMP master agent by executing the following commands:

   Windows Restart `SNMP Emanate Adaptor` and `SNMP Emanate Master Agent` using the `Control Panel Services` applet.

   UNIX **`/usr/sbin/snmpd`**

5. Start NNM by executing the following command:

   Windows Click the `NNM Services - Start` icon.

   UNIX **`ovstart -v`**

6. Make sure the `ovtopmd` service (background process on UNIX systems) is running by executing the following command:

   Windows Click the `NNM Status` icon.

   UNIX **`ovstatus ovtopmd`**

Configuring a Topology Filter for a Collection Station

A topology filter at a collection station defines the subset of topology data that management stations can access. For more information, see “Topology Filtering” on page 41. Use the following procedure to configure a topology filter on a collection station:

1. Create a filter for the *install_dir*`\conf\C\filters` (`$OV_CONF/$LANG/filters` on UNIX systems) file. You can either use a predefined filter or create a new filter. Refer to Appendix A, “The Filter Definition Language,” on page 169 for information on filters.
2. Make sure NNM is not running. If NNM is running, exit all sessions and issue the following command:

   Windows Click the `NNM Services - Stop` icon.

   UNIX **`ovstop`**

3. Edit the `ovtopmd.lrf` file to include the `-f` option followed by the desired filter name or filter expression name. For example,

```
ovtopmd:ovtopmd:
OVs_YES_START:pmd,ovwdb:0 -f Routers: OVs_WELL_BEHAVED:15:
```

   where, `Routers` is a predefined filter contained in the standard filter definition file.

4. Reconfigure `ovtopmd` by executing the following command (from a command window on systems running the Windows operating system):

   `ovaddobj ovtopmd.lrf`

5. Start the NNM services by executing the following command:

   Windows Click the `NNM Services - Start` icon.

   UNIX **`ovstart -v`**

   The start-up of NNM may take a little longer the first time, to adjust for `ovtopmd` to apply the filter to the existing topology database.

Only information that passes the topology filter is passed on to the management station. Existing information on the management station that does not pass the newly-configured filter will be removed from the management station's topology database.

Changing the Topology Filter for a Collection Station

You can change the topology filter for a collection without stopping NNM using the following procedure.

1. Create a filter for the *`install_dir`*`\conf\C\filters` (`$OV_CONF/$LANG/filters` on UNIX systems) file. You can either use a predefined filter or create a new filter. Refer to Appendix A, "The Filter Definition Language," on page 169 for information on filters.

2. Execute the following command (from a command window on systems running Windows):

   `ovtopmd -f` *`filtername`*

   where filtername is the new filter that you want to apply to the database. This command sends an event that tells `ovtopmd` to apply the new filter to the topology database.

To remove the topology filter, execute the following command (from a command window on systems running Windows):

```
ovtopmd -f ""
```

NOTE To make the filter application permanent, perform steps 3 and 4 from the "Configuring a Topology Filter for a Collection Station" on page 135.

Configuring IPX on a Collection Station under Windows

If you want to prevent IPX information from being passed to the NNM management station, do one of the following:

- During installation of the NNM collection station, disable IPX discovery. The following procedure prevents IPX discovery from occurring on the collection station.

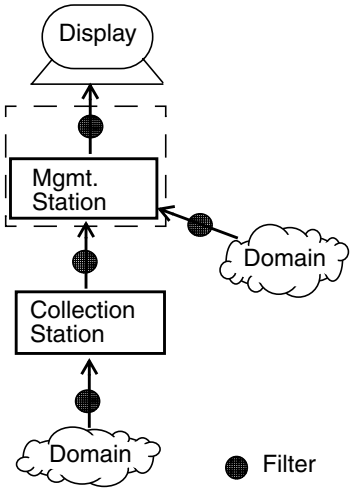
  1. Choose the `Custom Installation` option, and disable the automatic starting of network discovery.
  2. Once installation is complete, start the NNM user interface.
  3. Select the `Options:Network Polling Configuration:IP/IPX` menu item.
  4. From the `[IPX Discovery]` tab, click on the `[Discover New Nodes]` button to disable IPX discovery.
  5. Start the IP discovery services using the `NNM Services - Start` icon in the `HP OpenView` program group.

- If you want IPX discovery enabled on the NNM collection station, proceed with a normal installation. After the system is installed, configure the `IPOnlyObjects` filter as a topology filter. Refer to "Configuring a Topology Filter for a Collection Station" on page 135 for this information. This prevents any IPX networks or IPX-only objects from being propagated to the management station. However, nodes that are discovered to support both IP and IPX will still contain IPX interfaces, with only a physical address and no IP address.

Configuring Management Stations

This section describes how to:

- Determine if a node is a collection station.
- Determine all objects monitored by a specific collection station.
- Configure a management station to use a specific collection station.
- Unmanage a collection station from a management station.
- Remove a collection station from a management station.
- Configure a remote communication timeout and retry count.
- Perform a full synchronization.



There are important differences between the scalability features delivered with the NNM Starter Edition and Advanced Edition. The NNM Advanced Edition can function as a management station, collection station, or both. The NNM Starter Edition can only function as a collection station.

Determining if a Node is a Collection Station

There are several ways to determine what nodes are collection stations.

- The easiest is to use the Station view available from the `Tools:Views->Station` menu, or at this URL: `http://`*`nodename`*`:7510/topology/stationView`, where *nodename* is an NNM management station.
- Alternatively, you can execute the following command (from a command window on systems running a Windows operating system):

  `xnmtopoconf -print`

This will give you a list of current collection stations and tell you if they are managed or unmanaged. The nodes with a managed status are currently acting as collection stations.

- Finally, you can also use ovw to determine which nodes are acting as collection stations. Use the following procedure:

  1. From the menu bar select `Edit:Find->Object By Attribute`. The `Find by Attribute` dialog box appears.
  2. In the `Objects Attributes` list, select `isCollectionStationNode`.
  3. Click `[Apply]`. The known collection stations will be in the `Located and Highlighted` list in the `Locate Attribute` dialog box.

NOTE It is possible for collection stations to exist without having a node object. For example, when a collection station is being used from a remote location and it is not part of the locally discovered topology.

Determining All Objects Monitored by a Collection Station

You can see all the objects monitored by a given collection station by specifying the `ovtopodump` command with the `-r` and `-C` options together. For example, (from a command window on systems running a Windows operating system)

```
ovtopodump -rRISC stn2
```

If there is both a node and a collection station with the same name, it may be necessary to use the object ID to distinguish the two objects with the `-r` option. Another option is to print all objects and use the `grep` command to find the objects from that collection station. For example,

```
ovtopodump -RISC | grep stn2
```

Configuring for a Specific Collection Station

You can configure a management station to connect to a collection station. Use the following procedure to do this (you must be `Administrator` on systems running a Windows operating system or `root` on UNIX systems to execute this procedure):

1. Make sure the NNM services (background processes on UNIX systems) are running:
   - On Windows, use the `Control Panel Services` applet to check status. If the NNM services are not running, use the `Control Panel Services` applet to start the services.
   - On UNIX, execute the following command:

     `ovstatus`

     If the NNM services are not running, execute the following command:

     `ovstart`
2. Determine what collection stations already exist as described in “Determining if a Node is a Collection Station” on page 138to get a list of current collection stations and see if they are managed or unmanaged. If the collection station you want does not appear, continue to the next step. If the collection station *does* appear, continue with step 4.
3. Add the collection station to the topology by executing the following command:

   `xnmtopoconf -add -unmanage` *`collectionstation`*

TIP

When NNM is first started, all the collection stations attempt to report to the management station at the same time, slowing down the system. Unmanage them and bring them on one at a time.

This same command is used for the floating host name of a Service Guard cluster on UNIX systems. As part of your Service Guard configuration process, you updated the `ov.conf` configuration file `HOSTNAME= parameter`. Refer to the *ov.conf* reference page in NNM online help (or manpage on UNIX systems) for more information.

4. Configure the `set` community name to be used when communicating with the collection station by selecting `Options:SNMP Configuration` from the menu bar. This displays the `SNMP Configuration` dialog box.
5. In the "`Specific Nodes`" tab (`SNMP Parameters` pane of the dialog box on UNIX systems), specify in the `Set Community` text field the same set community name that you configured on the collection station.
6. Click `[OK]` to close the `SNMP Configuration` dialog box.
7. To make sure the collection station is configured correctly, execute the following command (from a command window on systems running a Windows operating system):

   xnmtopoconf -test *collectionstation*

   If you see SNMP errors or timeouts reported, refer to "Troubleshooting Collection Station Communication" on page 148 for more information.
8. Manage the collection station by executing the following command:

   xnmtopoconf -manage *collectionstation*

   This causes the management station to start monitoring and synchronizing with the remote collection station. As part of this, the management station will be added to the collection station's internal list of interested managers. This internal list is referred to as `%REMOTE_MANAGERS_LIST%` in various places in the interface, such as the event configuration system, when forwarding is being configured.

Unmanaging a Collection Station

To unmanage a collection station from a management station, execute the following command (as `Administrator` on systems running a Windows operating system or `root` on UNIX systems) from the management station (from a command window on systems running a Windows operating system):

xnmtopoconf -unmanage *collectionstation*

This removes the management station from the list of active management stations (`%REMOTE_MANAGERS_LIST%`) for this collection station. The collection station however remains in the management station's database. This command may take a few minutes to complete.

All objects for which the collection station is the primary collection station are either assigned a new primary collection station or have their status set to unknown.

Since new primary collection stations are being assigned, this command may take some time to complete.

Removing a Collection Station from a Management Station

Determine what collection stations already exist as described in “Determining if a Node is a Collection Station” on page 138to get a list of current collection stations and see if they are managed or unmanaged.

To remove a collection station from the management station’s database, execute the following command (as `Administrator` on systems running a Windows operating system or `root` on UNIX systems) from the management station (from a command window on systems running a Windows operating system):

`xnmtopoconf -delete` *`collectionstation`*

This will remove all the symbols representing the collection station objects from the maps on the management station. When a collection station is deleted, a new primary is picked automatically. If none exists, the node is deleted from the management station.

This command may take some time to complete since all data reported from that collection station is removed. If you will be using this data in the future, you may want to unmanage the collection station instead of removing it.

Configuring a Remote Communication Timeout and Retry Count

A WAN link connecting a management station and a collection station will likely have a slow response time and a larger packet loss rate due to congestion. The following procedure will determine the appropriate set of timeouts to use for remote communication.

1. Ping the remote collection station by executing the following command (from a command window on systems running a Windows operating system):

ping *collectionstation*

For Solaris, use **ping -s** *collectionstation*.

After a period of time, stop the ping and note the average round trip time and the packet loss rate.

2. Configure the timeouts and retry count for the remote collection station, based on the behavior received from the ping.

   For example, for a reasonable timeout configuration, take the average ping round-trip time, double it, and add a little. For a retry count, the higher the packet loss rate, the more retransmissions you should specify to occur before a failed SNMP request. You should keep a retry count of at least 3.

3. Select `Options:SNMP Configuration` from the menu bar; the `SNMP Configuration` dialog box appears, as shown in Figure 4-8 (Figure 4-9 for UNIX systems).

4. In the `SNMP Parameters` section of the dialog box, specify the timeout configuration in the `Timeout` text box and the retry count in the `Retry Count` text box.

5. Click `[OK]` to save the changes and close the dialog box.

6. Verify communication with the remote collection station by executing the following command (from a command window on systems running a Windows operating system):

   xnmtopoconf -test *collectionstation*

If the test fails, refer to “Troubleshooting Collection Station Communication” on page 148, for more information.

Figure 4-8 Windows SNMP Configuration Dialog Box

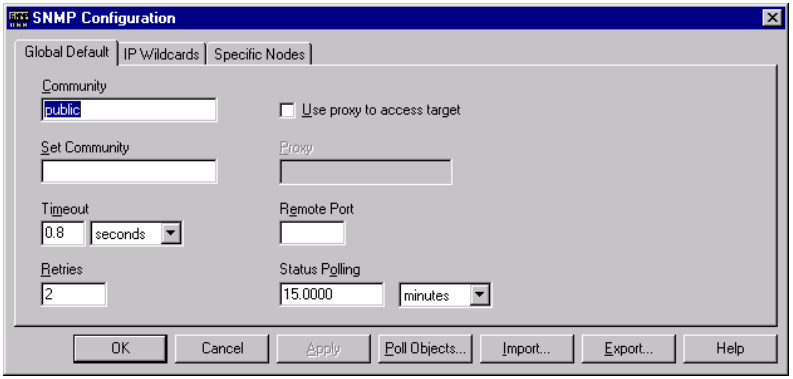
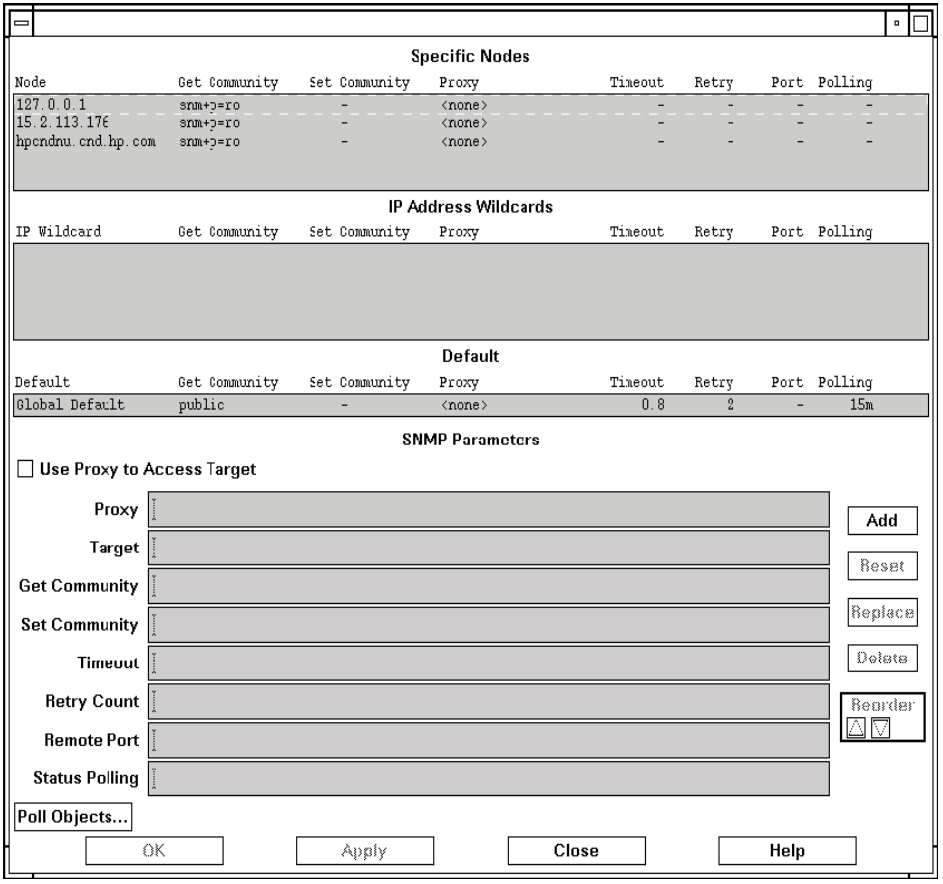


Figure 4-9 UNIX System SNMP Configuration Dialog Box



Performing a Full Topology Synchronization

A full topology synchronization causes a complete check of every attribute of every object reported by the remote collection station. This includes adding objects that are unknown and deleting objects that are not known at the remote collection station.

Perform a full synchronization by executing the command (from a command window on aWindows operating system):

nmdemandpoll -s *collectionstation*

All results and actions taken are reported to the command line. The output consists of the following:

- General processing status. There will be output noting the general flow of the synchronization process as well as general information related to the remote collection station (for example, hostname, port, version).
- Detailed output concerning the processing of all objects, regardless of whether any changes were detected.
- Warnings about objects that have conflicts or other errors in updating the objects in the database.
- Information about objects that no longer exist in the remote site's database and that are being removed from the local topology database.

TIP When resynchronizing peer-to-peer management stations, the simultaneous resynchronization can cause NNM to slow down and fail. Unmanage one until the other has resynchronized, then remanage it to synchronize it.

Identifying Collection Station Communication Problems

You can check the status of a collection station or management station using the NNM Station view.

You can open an NNM Station view from Home Base. See the *Using Extended Topology* manual for more information about Home Base. For other ways of opening an NNM Station view, see the *Launching Dynamic Views* section of the HP OpenView web online help.

The status color of the local management station indicates whether or not it is responding to NNM status polls. The status color of a collection station indicates whether or not the local management station is receiving data from that remote collection station:

- Tan color: The collection station is unmanaged.
- Green color (management station): The management station operational status is normal.

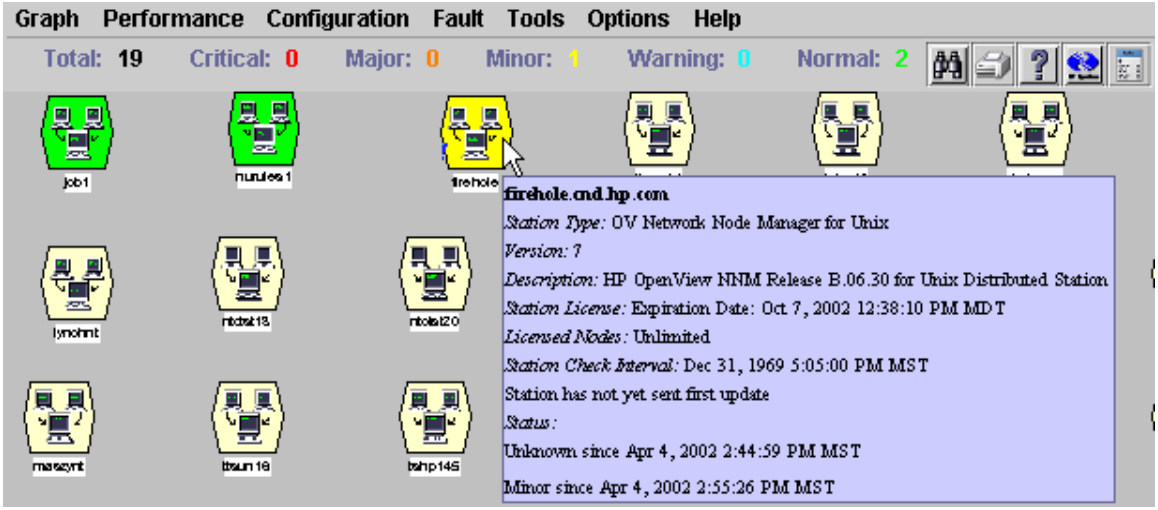
- Other color (management station): The management station operational status is abnormal.
- Green color (collection station): The collection station to management station communication status is normal.
- Other color (collection station): The collection station to management station communication status is abnormal.

In Figure 4-10 on page 147, the NNM Station view shows you that management station `job1` is managing `nurules1`. The status of both collection station `nurules1` and management station `job1` is normal.

However, the communication status of collection station `firehole` recently changed, as shown by the exclamation point, and is now showing an abnormal status color. You need to troubleshoot the communication from collection station `firehole` to management station `nurules1`. See “Troubleshooting Collection Station Communication” on page 148for troubleshooting techniques.

NOTE By resting your mouse pointer over collection station `firehole`, NNM displays some additional information.

Figure 4-10 Station View: Identifying Communication Problems



Troubleshooting Collection Station Communication

To troubleshoot the communication between a management station and a collection station use `xnmtopoconf`, as shown in the following procedure.

1. To determine the status of the managed collection stations, execute the command (from a command window on systems running a Windows operating system):

   `xnmtopoconf -print`

   This will produce a list of managed collection stations and their status as follows:

```
STATION            STATUS    INTERVAL OVERLAP MODE    FAILOVER   FAILOVER FILTER
ab.cnd.hp.com      Unmanaged 5m       AllowOverlap    off        <none>
be.cnd.hp.com      Critical  2m       AllowOverlap    ON         Routers
cat.cnd.hp.com     Unmanaged 5m       AllowOverlap    off        <none>
dog.cnd.hp.com     Normal    5m       AllowOverlap    configured <none>
emu.cnd.hp.com     Critical  5m       AllowOverlap    off        <none>
local              Normal    5m       DeleteSecondary off        <none>
yak.cnd.hp.com     Unmanaged 5m       AllowOverlap    off        <none>
```

   The `STATUS` is determined by the status checks, which are performed, by default, at 5 minute intervals. The `STATUS` can be one of the following values:

`Unmanaged`	The collection station is currently not being monitored.
`Unknown`	The collection station has not yet had a status check. The initial synchronization may be in progress.
`Normal`	The collection station has responded to the most recent status check.
`Warning`	The collection station has not responded to the most recent status check, but it did respond to the status check before that.
`Minor`	The collection station has not responded to two consecutive status checks.
`Major`	The collection station has not responded to three consecutive status checks.

`Critical`	The collection station has not responded to four consecutive status checks.

2. If the results of this command produce any collection stations with a status other than unmanaged, unknown, or normal, execute the following command to help determine the problem with this Windows operating system):

   `xnmtopoconf -test` *`collectionstation`*

   For example, executing this command on `hobbes.msr.hp.com` produces output similar to the following:

```
Testing station hobbes.msr.hp.com.
ICMP round trip time = 120 ms.
  Ping test to hobbes.msr.hp.com (15.31.16.141) succeeded.
  Get sysObjectID test failed:
    Error in SNMP reply: No response arrived before timeout
Test for station hobbes.msr.hp.com failed
```

   In this example, the get `sysObjectID` test failed. (Refer to the information below for potential causes.)

   Executing `xnmtopoconf -test` on a collection station with a status other than unmanaged, unknown, or normal can produce one of the following results:

   - `Ping test to the` *`collection station`* `failed:`

     This could be because the node is down or that the SNMP timeout or retry count needs to be increased.

   - `Get sysObjectID test to the` *`collection station`* `failed:`

     This could be because

     — the SNMP agent on the collection station is down.

     — the SNMP timeout or retry count needs to be increased.

     — the get community name is incorrectly configured.

   - `Get topology info test to the` *`collection station`* `failed:`

This could be because

— one or more of the services (background processes on UNIX systems) are not running on the collection station. Run `ovstatus` on the collection station to determine which services are not running. If necessary, run `ovstart` to start the services.

— the get community name is incorrectly configured.

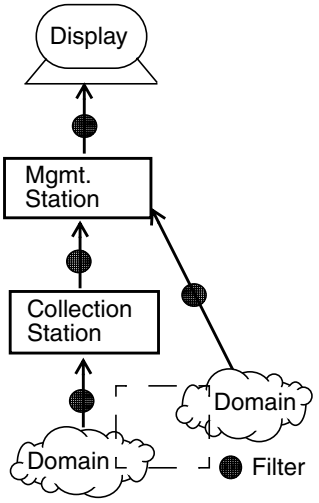
- `Set event forwarding test to the` *`collection station`* `failed:`

  This could be because the set community name is incorrectly configured on either the collection station or the management station.

Configuring Domain Overlaps and Failover

This section describes how to:

- Determine the primary station for an object.
- Change the primary station for an object.
- Change the overlap mode.
- Configure collection station failover.



Determining the Primary Collection Station for an Object

To determine the primary collection station for a given object, execute the following command (from a command window on systems running a Windows operating system):

ovtopodump -C *object*

This will print the primary version of the object with the name of the collection station reporting that version. For example, executing

ovtopodump -C gw1

produces the following output:

```
OBJECT ID(S)    OBJECT      STATUS       IP ADDRESS       STATION
146/145           gw1       Normal       10.2.112.1       remstn1
146/145           gw1       Normal       10.2.112.1       remstn1
146/595           gw1       Normal       10.2.104.1       remstn1
146/1158          gw1       Unmanaged    10.2.120.1       remstn1
```

In this example, the node `gw1` is a gateway being monitored by the collection station `remstn1`.

You can see which collections stations are monitoring an object by using the `-RISC` option of `ovtopodump`. For example, executing

ovtopodump -RISC gw1

produces the following output:

```
OBJECT ID(S)    OBJECT      STATUS      IP ADDRESS        STATION
146              gw1        Normal      10.2.104.1        remstn1
146/145          gw1        Normal      10.2.112.1        remstn1
146/595          gw1        Normal      10.2.104.1        remstn1
146/1158         gw1        Normal      10.2.120.1        remstn1
146             &gw1        Unmanaged   10.2.104.1        local
146/145         &gw1        Unmanaged   10.2.112.1        local
146/595         &gw1        Unmanaged   10.2.104.1        local
146/1158        &gw1        Unmanaged   10.2.120.1        local
146             &gw1        Unknown     10.2.104.1        stn2
146/145         &gw1        Unknown     10.2.112.1        stn2
146/595         &gw1        Unknown     10.2.104.1        stn2
146/1158        &gw1        Unknown     10.2.120.1        stn2
```

In the second example, the node `gw1` has the collection station `remstn1` as its primary station, but is also known to be monitored by `stn2` and by the local management station. (The special station name `local` is used to represent locally monitored objects.) The "`&`" symbol designates the secondary versions of the objects. (Meanings of other symbols in this location are documented in the *ovtopodump* reference page in NNM online help (or manpage on UNIX systems)).

Changing the Primary Collection Station for an Object

The system will choose a primary collection station for an object, based on the rules as described in "The Selection of Primary and Secondary Collection Stations" on page 53. This choice should be adequate for most purposes. However, you can explicitly override the choice of primary by specifying the `-primary` option to the `xnmtopoconf` command. For example, executing the following command makes the designated collection station the primary for the listed objects. This requires `root` access on UNIX systems or `Administrator` privileges on systems running a Windows operating system.

xnmtopoconf -primary *collectionstation object*

For example, if it was desired that `stn2` be the primary for `gw1`, the following command could be executed (from a command window on systems running a Windows operating system):

```
xnmtopoconf -primary stn2 gw1
```

This temporarily overrides the system's choice of primary. If something changes to affect the choice of primary according to the described rules, the system will change to a new primary. The choice can be made stronger by specifying the `-preferred` option of `xnmtopoconf`. For example, execute the following command (from a command window on systems running a Windows operating system):

`xnmtopoconf -preferred -primary stn2 gw1`

to mark the version from `stn2` as the preferred choice for the primary collection station for this object. So long as `stn2` is managed and in a noncritical state it will be the primary collection station for `gw1`. You can clear the preference by specifying the `-clearPreferred` option (from a command window on systems running a Windows operating system):

`xnmtopoconf -clearPreferred stn2 gw1`

Changing the Overlap Mode

The overlap mode determines how the system behaves when an overlap is detected between the local management station and a remote collection station. The overlap mode applies to the collection station from the management station's perspective. The overlap mode is applied when an object is made a secondary to another remotely monitored version of the object. (See "Overlaps with the Collection Domain of the Management Station" on page 55 for a complete description.) To change the overlap mode use the `-overlap` option of `xnmtopoconf` (from a command window on systems running a Windows operating system):

`xnmtopoconf -overlap` *`mode`* **`local`**

where *`mode`* can be one of the following:

- `Allow Secondary Local Objects`
- `Unmanage Secondary Local Objects`
- `Delete Secondary Local Objects` (default)

The mode applies to all monitored objects for the local management station that are made secondaries for other remotely monitored objects. The mode cannot be changed on an individual object basis.

Collection Station Failover

Collection station failover allows a management station to pick up status polling responsibility for a collection station that fails. To enable failover (from a command window on systems running a Windows operating system):

xnmtopoconf -failover *stationlist*

The station or stations in the *stationlist* will have status polling automatically taken over when the collection station is downgraded to critical. See the *xnmtopoconf* reference page in NNM online help (or manpage on UNIX systems) for details. If the collection station is currently critical, `-failover` causes the management station to take over status polling immediately.

To disable failover, use `xnmtopoconf -nofailover`.

Failover Filters

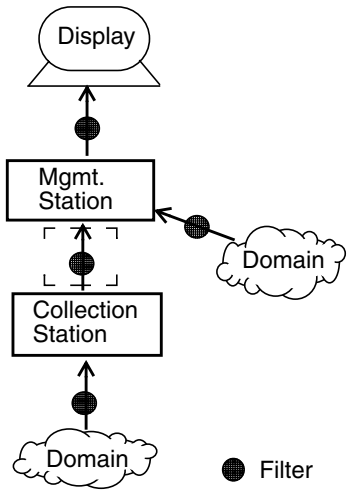
To specify which objects from the collection station's topology to include in status polling, specify a failover filter to `xnmtopoconf` (from a command window on systems running the Windows operating system):

xnmtopoconf -failoverFilter *filter stationlist*

A new or changed filter is applied to the objects associated with the collection stations in the *stationlist*. If the collection station is not failed over, the filter will be applied the next time it fails. However if the objects from a station are already failed over, `netmon` on the management station releases the objects associated with the collection station(s) and then reloads the collection station topology and applies the new filter as it loads the topology.

Configuring Event Forwarding and Correlation

You can configure the event system to forward events to multiple management stations. You can forward an event to remote managers that are using your station as a collection station, other management hosts, or a combination of these. Forwarding events provides distribution of events for the SNMP data collector and for distributed internet discovery and monitoring.



NOTE When forwarding threshold events, be sure that you forward the corresponding rearm event. See the *Managing Your Network with HP OpenView Network Node Manager* manual for information on configuring threshold and rearm events.

Forwarding an Event

You can modify events to have them forwarded to multiple management stations. When you create a new event, you can specify to have that event forwarded. HP recommends that you use multiple varbinds with individual pieces of data rather than using a full string. This allows the receiving machine to specify the formatting. This procedure explains how to modify an event.

1. Select `Options:Event Configuration` from the menu bar; the `Event Configuration` window appears. From the command line, you may run `xnmtrap`.

2. Select an enterprise from the `Enterprise list (the Enterprise Identification` pane on UNIX systems).
3. Select an event that you want to forward from the `Event Identification` pane.
4. On Windows systems, select `Edit:Events->Modify Event` from the menu bar in the `Event Configuration` window; the `Modify Events` dialog box appears. See Figure 4-11.

   On UNIX systems, select `Edit:Modify Event` from the menu bar in the `Event Configuration` window; the `Event Configurator/Modify Event` dialog box appears. Figure 4-12 shows the `Event Configurator/Modify Event` dialog box.
5. On Windows systems, click the [`Forwarding`] tab.

   On UNIX systems, click on the `Forward Event` check box to expand the dialog box to display the `Forwarded Event Destinations` pane.
6. Complete the `Forwarded Event Destinations` pane ([`Forwarding`] tab) by specifying the destinations to forward the events to.

   You can add destinations in the following ways:

   - Clicking `[Remote Mgrs]` to add all remote management stations to the `Forwarded Event Destinations` list. The remote management stations are specified in a list indicated by the special string `%REMOTE_MANAGERS_LIST%`. This list is created automatically as systems are defined to be collection stations.
   - Clicking `[Add From Map]` to add the destinations that are selected on the map.

NOTE

Even though the `%REMOTE_MANAGERS_LIST%` string exists, the actual file (*`install_dir`*`\bin\databases\remoteMgrs` on stations running a Windows operating system, `$OV_DB/remoteMgrs` on UNIX systems) may not exist, or may not contain the remote management stations you expect. You can verify the contents of the `remoteMgrs` file using a text editor.

   - Typing the destination in the `Destination` text box and clicking `[Add]`. The destination you specify can be a file containing a list of destinations, a hostname, or an Internet address.

7. Click `[OK]` from the `Modify Events` dialog box.
8. Click `File:Save` from the `Event Configuration` window to save the changes.

You can delete destinations by selecting a destination and clicking `[Delete]` to delete a specific destination, or clicking `[Delete All]` to delete all the destinations in the `Forwarded Event Destinations` list. You must click on `File:Save` from the `Event Configuration` window to apply and save the changes.

Figure 4-11 Windows Modify Events Dialog Box, Forwarding Tab

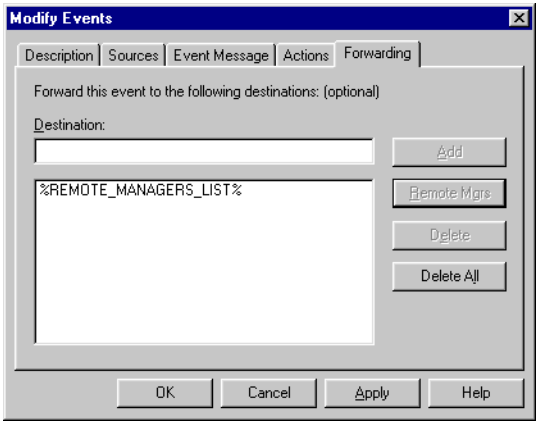
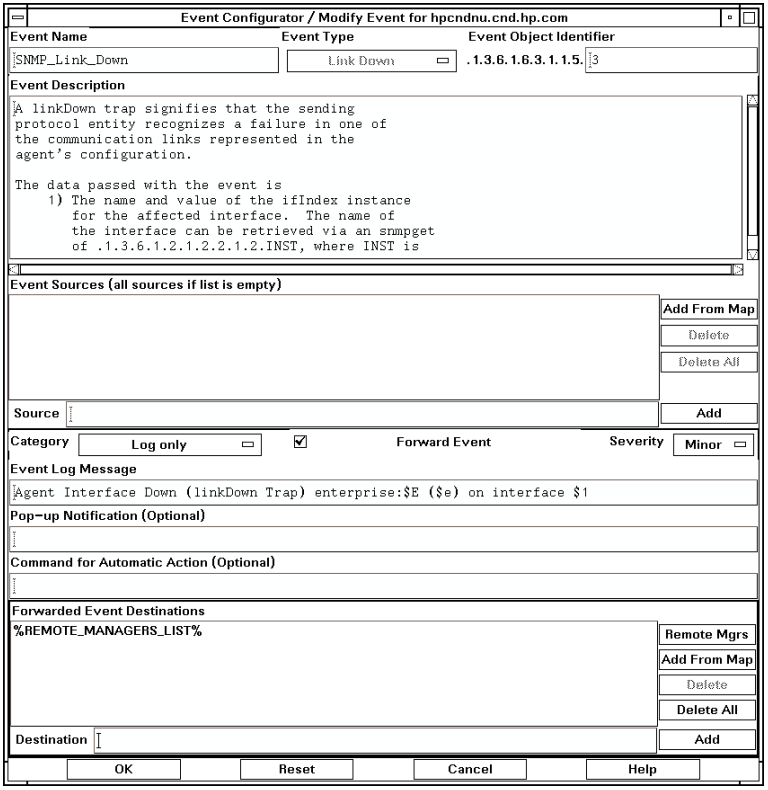


Figure 4-12 UNIX System Event Configuration/Modify Event Dialog Box



Configuring Interface Status Polling

NNM monitors and reports the status of each interface belonging to a managed node. This consumes additional system resources, increases network traffic when monitoring many interfaces, and can consume resources on managed devices.

Not every interface connects critical devices to the network. For example, a large network switch may have many of its interfaces connected to individual desktop computers. These interfaces go down whenever the users turn off their computers. If you are primarily concerned about monitoring the switch and less concerned about monitoring the interfaces, Interface Managed-State Automation can help.

You can use Interface Managed-State Automation to configure NNM with the interfaces you need to monitor. This approach offers the following benefits:

- NNM only shows status changes for important switch interfaces.
- The NNM management station uses fewer resources by not checking the status of unimportant interfaces.
- NNM can manage a larger environment when only monitoring and reporting the status of important interfaces.
- NNM generates less interface polling traffic on the network.

NOTE The Interface Managed-State Automation feature cannot control the managed state of nodes. It only operates upon interface objects already discovered by NNM.

For more detailed information about this topic, see the *ovautoifmgr (1m)*, *ovautoifmgr.conf(4)*, and *OVfilterIntro (5)* reference pages in NNM's online help (or the UNIX manpages).

Understanding Interface Managed-State Automation

NNM allows you to manually control which objects it manages. To do this, select a network interface object (that belongs to a node) and use the NNM menu to either manage or unmanage that interface. If you

configure NNM to manage an interface, NNM will poll that interface at a pre-configured interval to determine its current operational status. If you configure NNM to unmanaged an interface, NNM will not poll it.

Interface Managed-State Automation lets you automatically control the managed state for large quantities of interface objects. This feature is implemented by the `ovautoifmgr` command, which reads a configuration file containing your desired rules. These rules define the following conditions:

- Which nodes in the database should have their interfaces examined for their managed state.
- Which interfaces on these nodes should be managed or unmanaged, based upon the characteristics of the interface.
- Which interfaces on these nodes should be managed or unmanaged, based upon what type of node the interface is connected to.

The rules you develop make use of the NNM topology filter capability, which gives them significant flexibility. This allows the rules to be as simple or as complex as you need. Once you set up the rules, the `ovautoifmgr` command can apply them regularly to NNM, keeping NNM up-to-date as devices are added, deleted, or changed.

The `ovautoifmgr` command runs through the configured rules, making any managed state adjustments as needed, and then terminates. The `ovautoifmgr` command does not run continuously. You need to schedule the `ovautoifmgr` command to run on a periodic basis using the NNM system's scheduling capabilities (cron on Unix systems and the windows scheduler on Windows systems).

NOTE

You cannot limit NNM discovery with the `ovautoifmgr` command. For information about limiting NNM discovery, see "Topology Filtering" on page 41 or *Managing Your Network with HP OpenView Network Node Manager*.

Using Interface Managed-State Automation

When first running the `ovautoifmgr` command, you may want to change the managed state of many interfaces. Subsequent runs of the command will likely change the managed state of only a few interfaces. When

NNM makes many status changes, it can generate a large number of NNM events in a short amount of time. This can cause NNM to become temporarily unresponsive. To avoid this, run the `ovautoifmgr` command for the first time only when NNM responsiveness is not as important.

NOTE When using the `ovautoifmgr` command for the first time, you can reduce the load on the system by temporarily disabling any Event Correlation System circuits, and closing any open maps.

Carefully review the following information before using the Interface Managed-State Automation feature.

Understand your `ovautoifmgr` Schedule

It is important to understand how the how the `ovautoifmgr` schedule affects interface management in your environment. Consider the following possibilities:

- If NNM discovers new interfaces, it must be acceptable to have them managed until the next time you run the `ovautoifmgr` command.
- If you change the network so that an interface now needs to be managed, it should be acceptable to wait until the next scheduled run of the `ovautoifmgr` command for the change to happen.
- If nodes have interfaces that are dynamically created and deleted (for example, with the making and breaking of dialup phone line connections), these interfaces will only be unmanaged by the `ovautoifmgr` command if they happen to exist at the time the command is run. If that interface is later deleted and then re-added, it will again be in the managed state.

Behavior in a Distributed Environment

If you use the `ovautoifmgr` command to unmanage interfaces being managed by a management station, it only affects the status of interfaces being polled by the management station. It does not affect the interfaces being managed by the collection station.

For example, if you are using NNM in a distributed management environment, use the `ovautoifmgr` tool on each collection station to control that station's polling behavior. The collection station managing the interface propagates the status of each interface to the management

station. If multiple collection stations manage the same interface, then the collection station that is considered primary for that object is the one whose status is displayed on the management station.

Unmanaging Several Interface Populations

While a single invocation of the `ovautoifmgr` command uses only one configuration file, you can use the command at different times with different configuration files. Here are some possible uses of multiple configuration files:

- Suppose you want to adjust the managed state of router interfaces and network switches on a alternating days. You can achieve this by building two configuration files: one file that details the rules related to routers and a second file containing rules related to the switches. Using this approach, the scheduling tool would start the `ovautoifmgr` command with a different `-f` option on different days.
- Suppose you want to have certain interfaces managed during the day, but unmanaged during the night. To meet this need, set up two configuration files. The first file could be used to manage these interfaces and the second file could be used to unmanage them. These two invocations of the `ovautoifmgr` command would specify the appropriate configuration file with the -f option.

NOTE As mentioned before, be careful if you are changing the managed state of many interfaces, causing NNM to generate many status changes. These status changes can generate a large number of NNM events in a short amount of time. Avoid using this method to adjust the managed state of a large number of interfaces.

- Suppose you want to unmanage a large number of interfaces, but don't want to slow NNM's performance by unmanaging them all at once. To unmanage these interfaces gradually, you can use several files, separate these interfaces into smaller groups, and spread out the execution times.

Interfaces You Cannot Unmanage

The `ovautoifmgr` command never adjusts the managed state of the following types of interfaces:

- Interfaces contained within an unmanaged node.
- Interfaces created by NNM itself for the purpose of network layout. Such interfaces are not associated with real interfaces on the device, do not have an IP address, and are not status-polled by NNM.
- Interfaces discovered by an NNM Collection Station (CS). You cannot run the `ovautoifmgr` command from an NNM Management Station (MS) and change the managed state of a device being managed by a CS. Only locally discovered interfaces can be managed or unmanaged by the `ovautoifmgr` command. To remedy this, run the `ovautoifmgr` command from the CS.

NOTE Do not use the `ovautoifmgr` command to unmanage all IP interfaces on a node. This could prevent NNM from successfully using SNMP to learn of changes to the node that would otherwise cause interfaces to become managed again.

Other NNM Behaviors When Using the `ovautoifmgr` Command

For network switch or repeater devices, which normally only have one IP interface associated with them, you can use the `ovautoifmgr` command to unmanage the non-IP interfaces, and leave the IP address managed. For routers, which normally have many IP addresses, you can leave at least one reliably accessible IP address managed.

Unmanaging an interface will not affect the ability of NNM to accomplish network layout tasks. Even though the `ovautoifmgr` command may choose to unmanage an interface associated with a switch port, NNM will still be able to detect changes in what is connected to that switch port and will show these changes on the map.

Suppose NNM can only access one IP address of a router. There are several methods you can use depending on your needs:

- If it is critical that status be maintained on all of its interfaces, then you should use the `netmon.snmpStatus` file. See *Managing your Network with HP OpenView Network Node Manager* for more information.
- Suppose NNM can only access one IP address of a router. If you do not need status for all interfaces on this router, then use the `ovautoifmgr` command to unmanage all but one interface.

Here is one final example showing why you might consider unmanaging an interface. Suppose your network contains a device that behaves abnormally when NNM status polls one of its interfaces. To remedy this, use the `ovautoifmgr` command to prevent status polling on all of its other interfaces, allowing status polling of the main interface only.

Troubleshooting Interface Managed-State Automation

When setting up the `ovautoifmgr` command to run regularly, do a trial run with the proposed configuration directly from the command line, and then observe its behavior.

When the `ovautoifmgr` command detects an error condition, it reports this error both in the log file and on the `stderr` output from the program. It is possible for the command to perform differently than you intended while not detecting any errors. To understand how the command executes, run the `ovautoifmgr` command with the -v (verbose) option enabled. This means that the log file will contain additional information that you can study to determine correct operation.

In addition to the default (errors only), the `autoifmgr` command contains three levels of logging verbosity as shown in Table 4-1. They are controlled by how many times the -v flag appears on the command line.

Table 4-1 Logging Verbosity of the `autoifmgr` Command

ovautoifmgr command option	Behavior when using command option
`-v`	Program phases are noted, along with final statistics. No node or interface information.
`-vv`	The same as with the `-v` option, plus logging of each node that is processed.
`-vvv`	The same as with the `-vv` option, plus logging of each interface that is processed.

When very detailed logging is needed, and the managed environment is large, the amount of log output can get very large. Since the log file size is limited, older information may be lost. Therefore, for debugging purposes, it is good to set the size of the log file to be very large, and put the log file on a disk volume that has sufficient space to hold all the data.

The log file may contain the output from several runs of the `ovautoifmgr` command. When debugging, you should use a new log file for each run. See the `ovautoifmgr` reference page (or UNIX manpage) for more information.

In the log output, specific nodes and interfaces are identified by numbers. These numbers are the NNM object identifiers. More details on these objects can be found from the NNM databases by using the `ovtopodump` or `ovobjprint` commands giving them the object identifier.

If a certain node or interface appears to have not been properly processed, use the `ovtopodump` command with the node name to find the node and interface object identifiers. Then you can see where these numbers are referenced in the log file and see what decisions were made.

There are some cases in the log file and in the output of the `ovtopodump` command where a UUID is displayed rather than the NNM object ID. The UUID is a long, multi-character string that looks like this: `90553e90-27e6-71d7-184c-010203040000`. You can use the `ovtopodump` command with the UUID as a parameter to see what object, name, and NNM id number it refers to.

Using Distributed Data Collection

This section discusses how to export data from collection stations into a single Relational Database Management System (RDBMS) for report generation. For each collection station, use the following procedure:

1. Use the Network Node Manager data collector tool to collect MIB data. Refer to the *Managing Your Network with HP OpenView Network Node Manager* manual for information on data collection.
2. Determine which data collector files you want to merge into the RDBMS. These files are located in *install_dir*\databases\snmpCollect (`$OV_DB/snmpCollect` on UNIX systems).
3. Use `ftp` or `rcp` to copy the files from the collection station to a temporary data directory on the management station. You should also copy each data file's corresponding description file. The description file is named identically to the data file, except that an exclamation mark (`!`) is appended to the end of the filename. For example,

   ```
   rcp $OV_DB/snmpCollect/ifInOctets.1 clifford:/tmp/collsta5
   rcp $OV_DB/snmpCollect/ifInOctets.1! clifford:/tmp/collsta5
   ```

   copies the data and description files for instance 1 of MIB variable `ifInOctets` to the management station `clifford`.

   Alternately, if the network link between the management station and collection station is of sufficient capacity, you may make the files accessible via NFS.
4. If the file *install_dir*\conf\collectProxyMap (`$OV_CONF/collectProxyMap` on UNIX systems) exists on the collection station, copy it to the management station as well, either in the temporary data directory or another directory.
5. Make sure the temporary directory on the management station has `drwxrwxrwx` permission and that each data file has `-rw-rw-rw` permission.

6. On the management station, run the `ovcoltosql` command as `root`, to export the data into the RDBMS. Specify the temporary data directory as the data source by using the `-s` option. For example, (from a command window on systems running a Windows operating system):

   `ovcoltosql -Ro -s /tmp/collsta5`

   This example exports data from the directory `/tmp/collsta5` into an Oracle® database.

   If there is a `collectProxyMap` file for the collection station, use the `-p` option of `ovcoltosql` to specify the proxy map file location.

   Refer to the *ovcoltosql* reference page in NNM online help (or manpage on UNIX systems), and to the *Reporting and Data Analysis with HP OpenView Network Node Manager* manual for options that you can specify with this command.

If two or more collection stations collect on the same instance of the same MIB variable on the same node, they are said to have an overlapping collection. If data from an overlapping collection is merged into a single RDBMS, some of the data will not be entered into the RDBMS. `ovcoltosql` exports the data that is more recent than that already in the RDBMS. The collection station from which data is exported first will have data from its overlapping collection entered into the RDBMS, while most, if not all of the overlapping data from the second collection station will not be entered.

NOTE

You can execute `ovcoltosql` on the collection stations themselves to a common RDBMS server, but it is not recommended. This can cause potential performance problems when slow network links are traversed. Also, this approach results in the `topo_id` column of the `snmp_trend_dataset` table, which consists of topology IDs, to be assigned by different collection stations. Hence, the column will be useless for performing SQL joins across topology and trend tables.

Refer to the *Reporting and Data Analysis with HP OpenView Network Node Manager* online manual for information on RDBMS querying and administering operations.

A The Filter Definition Language

This appendix discusses in detail the language used to define filters in the filter definition file, giving you the information you need to modify existing filters, or define new ones.

This appendix is a reference to the contents of the filter definition file, and to the grammar of the filter definition language.

NOTE

This appendix assumes that you are familiar with two topics normally covered in the study of computer science: Boolean expressions and Backus-Naur Form (BNF) grammars. This appendix makes no attempt to teach the fundamentals of either Boolean expressions or BNF grammars. If you are unfamiliar with these topics, you may find it helpful to review these ideas in a computer science text book before you feel entirely comfortable defining new filters.

The Filter Definition File

Filters are defined in an editable file of ASCII characters; this is called the filter definition file. The format and contents of this file make up the subject of this appendix.

The filter definition file for all the filters you use with HP OpenView Network Node Manager (discovery, topology, map, and persistence) is *install_dir*\conf\C\filters on Windows operating systems and $OV_CONF/$LANG/filters on UNIX systems.

After you install NNM, the filter definition file contains a selection of useful definitions that you may be able to use as is, or with some modifications.

The filter definition file has several key components, and they appear in the file in the following order:

- Set definitions.
- Filter definitions, which each include one or more Attribute Value Assertions.
- Filter-Expression definitions, which each refer to one or more filters.

Each of these components is described in considerable detail in later sections of this appendix.

The filter definition file contains many instances of the above elements, always in order. That is, all set definitions are together at the top of the file, all filter definitions follow next, and all filter-expression definitions are at the end of the file.

As you examine the filter and filter-expression definitions in the sample filter definition file (“Default Filter File” on page 199), you will notice that some are intended for use as map filters, others are for discovery filtering. This distinction helps clarify the purpose of each filter and its components, and simplifies your troubleshooting of new or modified filters. Also, remember which types of filters are intended to *exclude* items and which are to *include*. Refer to Table 2-1 on page 38 for more information.

Filters and Filter-Expressions

The whole purpose of the filter definition file is to define the filters that are used with the scalability features of NNM. You have two basic types of filters: “simple” filters (which are simple only in that they do not refer to other filters), and filter-expressions.

Filters and filter-expressions are closely related and used identically. Everywhere else, they are referred to broadly as “filters.” Within the filter definition file, however, there is a distinction between them:

- A **filter** is a combination (or “expression”) of one or more Attribute Value Assertions.[1]
- A **filter-expression** is a combination (or “expression”) of one or more previously defined filters. In other words, a filter-expression is a “high- level filter,” created by combining lower-level filters that are already defined.

In every other respect, a filter-expression is exactly like a filter. You use filters and filter-expressions in exactly the same way. That is, anywhere a filter can be used, a filter-expression can also be used. Although most of the examples in this section uses filters, filter-expressions would be equally valid in those same examples.

You use logical, also called Boolean, operators (that is, And (&&), Or (||), and Not (!)), and normal precedence grouping via parentheses, to combine multiple filters in a filter-expression, or to combine multiple Attribute Value Assertions (AVAs) in a filter.

Defining filters and filter-expressions has two important rules:

1. A filter definition can include *only* AVAs. A filter definition can *not* include filters or filter-expressions.
2. A filter-expression definition can include *only* filters or filter-expressions that are defined earlier in the filter file. A filter-expression definition can *not* include AVAs.

1. See “Attribute Value Assertions” on page 174 for more details.

Sets

String-valued attributes can be tested for membership in a set. A **set** is simply a list of strings. The only operation available on sets is the test for membership.

You can enumerate the members of a set in two ways:

- Within the filter file itself, according the filter file grammar.[1]
- In a separate file, where set members are listed, one per line. You cannot use wildcards or ranges. If the set member begins with a slash (“/”), it is used as the absolute pathname of such a file. Relative pathnames and environment variables are invalid. In the Windows operating system, filter files must use the forward slash and should include a drive specification.

Files and member strings can be mixed in the same set definition. Set definitions can not be nested.

1. See “Filter File Grammar” on page 189 for details

Attribute Value Assertions

An **attribute value assertion** (**AVA**) is a statement about the value of an attribute. For example, here is an AVA about the `"IP Hostname"` attribute:

```
"IP Hostname" == "server7"
```

An object can be compared against an AVA to determine whether that assertion is true or false for the object. In other words, an AVA is a condition that an object either does or does not meet. When an AVA is tested against a particular object, one of three results can occur:

- The object does not have that attribute.
- The object has the attribute, but the value of the attribute does not match the value in the filter.
- The object has the attribute, and the value of the attribute matches the value in the filter.

The above AVA will be false when the object has an IP hostname attribute, but the IP hostname of the object is *not* “server7”. It will *also* be false when the IP hostname attribute is not present in an object.

The above AVA will only be true when the object has an IP hostname attribute, and the IP hostname of the object is “server7”. Similarly the AVA `isRouter==TRUE` is only true for routers.

AVAs are defined in terms the of fields of an object and possible values of those fields. The fields used in creating AVAs for filters are a subset of the fields contained in the object database (`ovwdb`).[1]

NOTE Many field names in the object database have embedded white space. Any field name or field value containing white space must be enclosed in double quotes (for example `"Selection Name"`).

In the object database, attributes can have four types:

1. See “Filterable Objects and Attributes” on page 183 for the specific fields you can use in filters.

- Boolean
- Integer
- String
- Enumerated

Attribute value assertions fall into one of these four categories, depending on the attribute type.

After a short section that discusses the operators that are valid in AVAs, AVAs for each of the above types of attributes are covered in more detail.

NOTE Attribute Value Assertions can only be tested against literal constant values. The attributes of two different objects cannot be tested against each other.

Valid Operators for AVAs

These are the operators that you can use to express AVAs:

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
~	Like, which used for pattern matching.[1]
!~	Not like, also used for pattern matching.
IN	Is a member of the set

1. See “String AVAs” on page 177 for more information.

TIP Filters are "short-circuit" evaluated. This means NNM only reads far enough to get an answer, then doesn't read the rest of the filter definition. For best performance, put likely TRUE clauses first when OR-in, and put likely FALSE clauses first when AND-ing.

Table A-1, Operators Allowed in AVAs for Each Field Type, shows which of these operators are valid in particular AVAs, depending on the field type of the attribute.

Table A-1 Operators Allowed in AVAs for Each Field Type

Operator	ovwStringValue	ovwEnumValue	ovwIntValue	ovwBooleanValue
==	Yes	Yes	Yes	Yes
!=	Yes	Yes	Yes	Yes
>	Yes	No	Yes	No
>=	Yes	No	Yes	No
<	Yes	No	Yes	No
<=	Yes	No	Yes	No
~	Yes	No	No	No
!~	Yes	No	No	No
IN	Yes[1]	No	No	No

1. Valid only if the value of the expression is the name of a defined set. See "Sets" on page 173 for details.

Boolean AVAs

Boolean AVAs can be expressed with just an attribute name (with an optional, leading `NOT` operator), or can be explicitly tested against the keyword values `TRUE` and `FALSE`. For example, testing for a object being a router can be expressed in either of the two following ways:

```
(isRouter)

(isRouter == TRUE)
```

The inverse expression, testing for a non-router, could be expressed in either of following ways:

```
(! isRouter)
```

```
(isRouter == FALSE)
```

Integer AVAs

Integer AVAs can be tested for equality, inequality, greater than, greater than or equal to, less than, and less than and equal to (==, !=, >, >=, <, <=). For example, to test for a multi-homed node, an expression using an integer AVA might look like this:

```
(numInterfaces > 1)
```

Enumerated (Enum) AVAs

In the object database, enum valued attributes have their enumerated values expressed explicitly. In enumerated AVAs these same enumerated values are used. For example, to test whether an object is from a particular vendor, the following AVA might be used:

```
(vendor == "HP")
```

Enum AVAs can only test for equality or inequality. Because the integer value of a given enum value can change, you must use enum strings in enum AVAs, and not their integer equivalents.

All enumerated values must be delimited (enclosed) in double quotes (”), as in the above example.

String AVAs

String AVAs can be tested using any lexical operator (==, !=, >, >=, <, <=). For example, if you were testing for a particular host, you might use the following AVA:

```
("IP Hostname" == "gomez")
```

Strings are tested using the `strcoll` function, and thus take advantage of any natural language semantics provided by that function.[1]

1. See the manpage on UNIX systems for *strcoll*(3).

In addition to standard lexical comparison, extended regular-expression matching (as defined in *regcomp*(3))[1] is also available via the "like" (`~`) operator. When you use this operator, the string is treated as a `regcomp`-style regular expression, in which certain characters have special meaning.

The AVA below matches *any* hostname with the sub-string "morticia" in it:

```
("IP Hostname" ~ "morticia")
```

while the next AVA (because it tests for "equality" rather than "like", and thus does not use regular-expression matching) matches only and exactly the hostname composed of the string "morticia":

```
("IP Hostname" == "morticia")
```

All regular expressions must be delimited (enclosed) in double quotes (`"`), as in the above example.

CAUTION

When negating the "is like" operator (~), place the negation *outside* of the expression. This is necessary due to NNM discovering all level 2 interfaces, including objects like serial cards which have no IP address. When no IP address is present, the object is assigned 0.0.0.0. If you negate the operator, as in `"IP Address" !~ 1.2.3.4`, you will inadvertently *include* all non-IP level 2 interfaces. By negating the expression, as in `!("IP Address" ~ 1.2.3.4)`, you get the set of all objects that *have* an IP address, and it is not like the one in the expression.

NOTE

See the "Caution" block in "Set Specifications in SNMP OIDs" on page 181 before using regular-expression matching with SNMP object identifiers or with IP addresses.

1. See the manpage on UNIX systems for *regcomp*(3).

Specialized Pattern Matching in NNM Filters

There are two special types of string-valued AVAs for which NNM provides specialized pattern matching on the associated attributes:

SNMP Object IDs (SNMP OIDs)	In a filter, an SNMP OID is identified as a leading period followed by decimal octets separated by periods. For example: `.1.3.6.34.2.209`
IP addresses	An IP address is identified as a string of exactly four (4) decimal octets separated by periods, but *without* a leading period: `23.26.11.14`

Three types of specialized pattern matching are available—wildcards, ranges, and sets. Set patterns are available only for SNMP OIDs. You can replace any decimal octet in an IP address or SNMP OID with a special pattern matching string.

NOTE Do not overlook the “Caution” block in “Set Specifications in SNMP OIDs” on page 181.

Wildcard Characters in SNMP OIDs and IP Addresses For IP address and SNMP OID pattern matching, the wildcard operator is the asterisk (`*`). An asterisk in an IP address indicates that any value is acceptable in the given decimal octet.

For example, to pass any address in the Class A IP network “23”, you would use the following in your AVA:

```
"IP Address" ~ 2 3.*.*.*
```

The wildcard character acts slightly differently for SNMP OIDs. In this case, a wildcard indicates that any value *for zero or more* decimal octets is acceptable. For example, the following pattern:

```
.1.4.5.1.*.13.2.*
```

matches any OID with ".1.4.5.1" followed by any number of decimal octets of any value (including no octets at all), followed by "13.2" followed by zero or more trailing decimal octets. Therefore, the above pattern would match the following OIDs:

- .1.4.5.1.13.2
- .1.4.5.1.22.33.13.2
- .1.4.5.1.13.2.14.55.232
- .1.4.5.1.22.44.33.13.2.23.32.1

but not:

- .1.4.5.1.13 (no 2 after the 13)
- .1.4.5.2.2.13.2 (no 1 after the 5)
- .1.4.5.1.33.22.2 (missing the 13)

Range Specifications in SNMP OIDs and IP Addresses A range is expressed by a decimal integer, a hyphen (-), and another decimal integer that is greater than the first. Decimal octet values between 2 and 68 would be specified as follows:

```
2-68
```

A range indicates that any value in the range (inclusive) is acceptable in the given decimal octet. Ranges are significant for the given decimal octet only, in both SNMP OIDs and IP addresses. The values specified in a range must be in their natural order, from smallest to largest; a reversed range (like, for example, 68-2) is not permitted.

The following are more examples of valid SNMP OIDs and IP address AVAs, using sets, wildcards, and ranges in combinations:

```
("IP Address" ~ 23.2.*.*)

("IP Address" ~ 23.2.3-9.*)

("IP Address" ~ 23.2.16-22.0-1)

("SNMP sysObjectID" ~ .1.3.6.*.9.5)

("SNMP sysObjectID" ~ .1.3.6.8-9.*)

("SNMP sysObjectID" ~ .1.3.6.7-8.12-13)
```

Set Specifications in SNMP OIDs In SNMP OIDs (but not IP addresses), you can specify a set of decimal octets in any position by separating the members by a comma. For example, in the following AVA:

```
("SNMP sysObjectID" ~ .1.3.6.34.8,9,11,23)
```

would match the following SNMP OIDs:

```
("SNMP sysObjectID" ~ .1.3.6.34.8)
("SNMP sysObjectID" ~ .1.3.6.34.9)
("SNMP sysObjectID" ~ .1.3.6.34.11)
("SNMP sysObjectID" ~ .1.3.6.34.23)
```

CAUTION

To invoke special pattern matching on IP address and SNMP OID strings, either the "like" operator (`~`) or "not like (`!~`) *must* be used.

Furthermore, to use NNM's specialized pattern matching facility, the IP address or SNMP OID string *must not* be enclosed in quotation marks. The presence of enclosing quotation marks means that the string will be treated as a regular expression, and *not* given specialized pattern matching; the absence of such quotation marks means the comparison *will* use NNM's specialized pattern matching facility.

The difference is important. Consider this AVA, where the value is quoted:

```
"IP Address" ~ "23.18.3.*"
```

With the quotation marks around the value (`"23.18.3.*"`), it is treated as a regular expression. The result is that *many* IP addresses beginning with "`23.18.3`" would cause the AVA to evaluate to `TRUE`, including, for example, `23.18.34.7`, which was probably not intended. In fact, even some strange and improbable IP address strings (like, for example, `23518.3711.1011.23033)` would pass this AVA. The reason is that, for regular expressions, the period character (`.`) matches *any* character in that location of the comparison string (including, confusingly, the period character itself).

However, if we use the same AVA, but *without* the quotation marks around the value (`"IP Address" ~ 23.18.3.*`), NNM's special pattern matching facility guarantees that only IP addresses in the apparently intended range would cause the AVA to evaluate to `TRUE`.

Excluding a Node

For the purposes of building and using filters, a node can be thought of as a “super-object”, which comprises the node and all its interfaces.

Thus, if a node object passes a filter, the filter will also pass all interfaces of the node regardless of whether they would, considered in isolation, pass the filter.

Similarly, if an interface passes a filter, the node object it belongs to, and all other interfaces of that node, also pass the filter.

Consequently, if you want a filter to exclude a node, the filter must block the node object itself and all of its interfaces.

Relationships Between Nodes and Interfaces

You can create a filter that combines nodes and interfaces, as shown in the following example.

```
nodeNinterface "" {(vendor = "SUN") && (IP Address ~ 15.2.112.*)}
```

Valid Operators for Logical Combinations of AVAs or Filters

AVAs, filter names, and filter-expression names are combined to create filter and filter-expression definitions (as noted before, *only* AVAs are allowed in a filter definition, and AVAs are *never* allowed in a filter-expression definition).

You use the following operators to create logical combinations (or “expressions”) of these components. Likewise, you combine filters or filter-expressions in a filter-expression definition with these operators:

Operator	Meaning
`&&`	Logical “and”, used to combine multiple AVAs and filter names into logical expressions.
`||`	Logical “or”, used to combine multiple AVAs and filter names into logical expressions.
`!`	Logical negation, used to negate stand-alone field names and other logical expressions. Fields that are already part of an expression cannot be negated; rather the entire expression should be negated.

Use parentheses—“(” and “)”—to specify explicit precedence in a group of operations. For example, a filter definition might look like this:

```
Filtr1 "" { isRouter && (numInterfaces > 1) }
```

Or a filter-expression definition might look like this:

```
FiltrExpr1 "" { (Filtr1 || Filtr2) && Filtr3 }
```

Filterable Objects and Attributes

The following four tables specify, for each type of object, the attributes (fields) for which filters can be defined. This is a subset of the objects and attributes in the `ovwdb` database.

Table A-2 Filterable Fields in Node Object

Field	Field Type
`IP Hostname`	String
`IP Status`	Enumerated
`isATM` *	Boolean
`isBGP4` *	Boolean
`isBridge`	Boolean
`isCDP` *	Boolean
`isCollectionStationNode`	Boolean
`isConnector`	Boolean
`isDS1` *	Boolean

Table A-2 Filterable Fields in Node Object (Continued)

Field	Field Type
isDS3 *	Boolean
isFrameRelay *	Boolean
isHSRP *	Boolean
isHub	Boolean
isIP	Boolean
isIPRouter	Boolean
isIPV6 *	Boolean
isIPX	Boolean
isIPXRouter	Boolean
isMcClusterMember	Boolean
isMPLS *	Boolean
isNetWareServer	Boolean
isNode	Boolean
isOSPF *	Boolean
isRMON *	Boolean
isRMON2 *	Boolean
isRouter	Boolean
isSONET *	Boolean
isSNMPSupported	Boolean
isSTP *	Boolean
isVRRP *	Boolean
isWireless *	Boolean
NetWare Server Name	String

Table A-2 Filterable Fields in Node Object (Continued)

Field	Field Type
Selection Name	String
SNMPAgent	Enumerated
SNMP sysContact	String
SNMP sysDescr	String
SNMP sysLocation	String
SNMP sysName	String
SNMP sysObjectID	String
TopM Interface Count	Integer
vendor	Enumerated

IMPORTANT * Fields marked with an asterisk have no effect on discovery filters.

Table A-3 Filterable Fields in Interface Object

Field	Field Type
isInterface	Boolean
isIP	Boolean
isIPX	Boolean
IPX Address	String
IP Address	String
IP Status	Enumerated
IP Subnet Mask	String
Selection Name	String

Table A-3 Filterable Fields in Interface Object

Field	Field Type
SNMP ifDescr	String
SNMP ifName	String
SNMP ifPhysAddr	String
SNMP ifType	Enumerated
TopM Network ID	Integer
TopM Node ID	Integer
TopM Segment ID	Integer

Table A-4 Filterable Fields in Segment Object

Field	Field Type
IP Segment Name	String
IP Status	Enumerated
isBusSegment	Boolean
isFDDIRingSegment	Boolean
isSegment	Boolean
isSerialSegment	Boolean
isStarSegment	Boolean
isTokenRingSegment	Boolean
Selection Name	String
TopM Interface Count	Integer
TopM Network ID	Integer

Table A-5 Filterable Fields in Network Object

Field	Field Type
IP Address	String
IP Network Name	String
IP Status	Enumerated
IP Subnet Mask	String
IPX Address	String
IPX Network Hop Count	Integer
isIP	Boolean
isIPX	Boolean
isNetwork	Boolean
Selection Name	String
TopM Default Seg ID	Integer
TopM Interface Count	Integer
TopM Segment Count	Integer

Valid values for the enumerated fields can be found in the files listed in Table A-6, Value Definitions for Enumerated Fields,

Table A-6 Value Definitions for Enumerated Fields

For the field named:	The enumerated values are defined in:
IP Status	On Windows operating systems: *install_dir*\fields\C\ip_fields On UNIX systems: $OV_FIELDS/$LANG/ip_fields
SNMP ifType	On Windows operating systems: *install_dir*\fields\C\snmp_fields On UNIX systems: $OV_FIELDS/$LANG/snmp_fields
SNMPAgent	On Windows operating systems: *install_dir*\fields\C\snmp_fields On UNIX systems: $OV_FIELDS/$LANG/snmp_fields
vendor	On Windows operating systems: *install_dir*\fields\C\ovw_fields On UNIX systems: $OV_FIELDS/$LANG/ovw_fields

Filter File Grammar

The following is the formal Backus-Naur Form (BNF) grammar for a filter file. See “Filters” on page 37 for details about where such a file is used.

```
Filter_File          ::=    Filters
                            | Filters FilterExprs
                            | Sets Filters
                            | Sets FiltersFilterExprs

Sets                 ::=    "Sets" "{" Set_Stmts"}
                            | "Sets" "{" "}"

Set_Stmts            ::=    Set_Stmt
                            | Set_Stmts Set_Stmt

Set_Stmt             ::=    SetName Description "{" Memberlist"}"

Memberlist           ::=    Member_Stmt
                            | Memberlist "," Member_Stmt

Member_Stmt          ::=    StringOrIdentifier
                            | <absolute_file_pathname>

Filters              ::=    Filters" "{" Filter_Stmts"}"
                            | "Filters" "{" "}"

Filter_Stmts         ::=    Filter_Stmt
                            | Filter_Stmts Filter_Stmt

Filter_Stmt          ::=    FilterName Description "{" Expression"}"

Expression           ::=    Expression "&&" Expression
                            | Expression"||" Expression
                            | "!" Expression
                            | "(" Expression ")""
                            | BoolExpr
                            | SetExpr

BoolExpr             ::=    FieldName
                            | FieldName ExprOperator <string>
                            | FieldName ExprOperator <integer>
                            | FieldName PatternOperator <string>
```

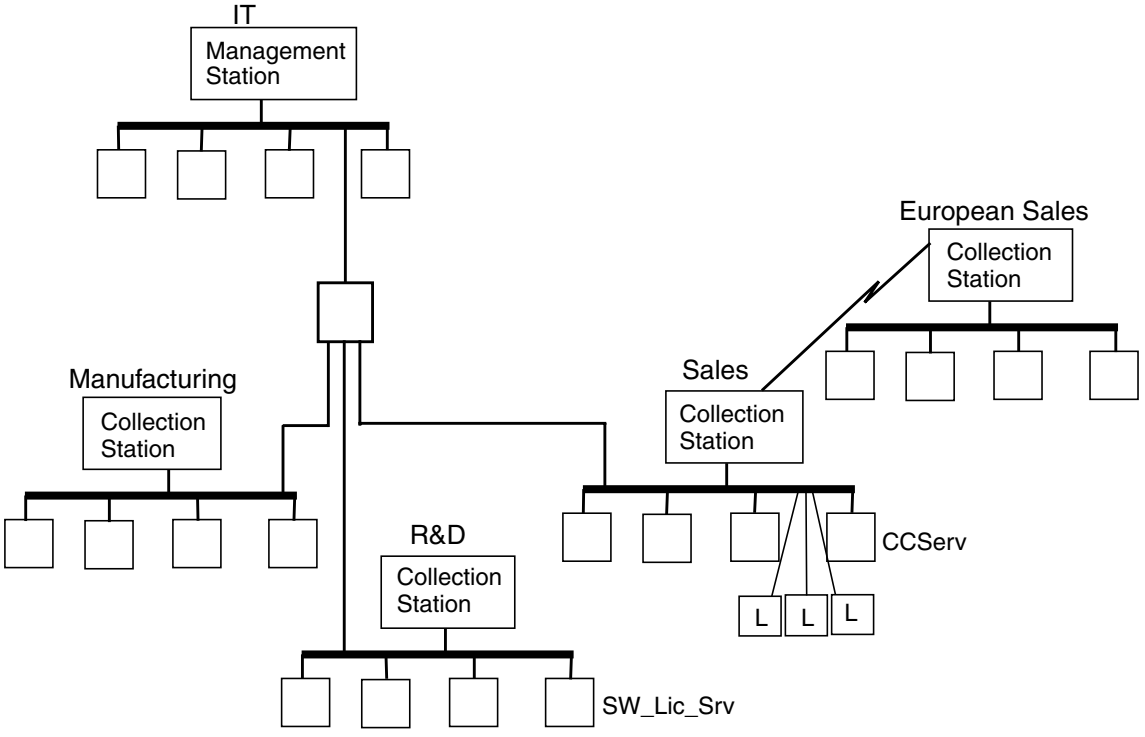
```
SetExpr              ::=    FieldName "IN" SetName

FilterExprs          ::=    "FilterExpressions" "{" FilterExpr_Stmts "}"
                            | "FilterExpressions" "{" "}"

FilterExpr_Stmts     ::=    FilterExpr_Stmt
                            | FilterExpr_Stmts FilterExpr_Stmt

FilterExpr_Stmt      ::=    FilterExprName Description "{" FiltExpr "}"

FiltExpr             ::=  FiltExpr "&&" FiltExpr
                          | FiltExpr "||" FiltExpr
                          | "!" FiltExpr
                          | "(" FiltExpr ")"
                          | FilterName

FieldName            ::=  StringOrIdentifier

ExprOperator         ::=  "==" | "!=" | "<" | "<=" | ">" | ">="

PatternOperator      ::=  "~" | "!~"

FilterExprName       ::= <unquoted_string>

FilterName           ::= <unquoted_string>

SetName              ::= <unquoted_string>

StringOrIdentifier   ::= <quoted_string> | <unquoted_string>

Description            ::=  <quoted_string>
```

Filter File Example

This example describes how these rules translate into usable filters for a sample network management solution. Hemispheres-Petroleum Company has decided to implement HP OpenView Network Node Manager's distribution capabilities to gain control over their overall network. They have a typical network organization that parallels their personnel organization -- local area networks for the Research and Development, Manufacturing, and Sales departments, with an additional network in place in Paris for European Sales. The network architecture is diagrammed in Figure A-1.

The IT department will be responsible for managing these networks, along with their own. They have decided to have a collection station for each department report to the management station in IT.

Figure A-1 Hemispheres-Petroleum Company Network Layout



The default filter file shipped with NNM has several useful building blocks already in place. (The default filter file appears on page 199.) Simple filters appear in the second section of the files, in the format `FilterName "Description" { AVA }`. For example, look at the line that says `Hubs "Any multi-port repeater" { isHub }`. The name of the filter, as it would later be used from the `ovw` user interface or in building filter expressions, is Hubs. `netmon` compares every node's SNMP response, looking for the `isHub` property. Any object possessing this property fits the filter `Hubs`. Similarly useful simple filters are in place for Networks, Segments, Nodes, IPRouters, Bridges, SNMPNode, HPNodes, and NonHPNodes.

Setting Up Collection Domains

Collection domains are specified through discovery filters. Setting these up demonstrates several tips that will help you manage filters for your network.

TIP Use a naming convention to designate different types of filters in your filter file.

In this example, all discovery filters have a suffix of _DF. This helps in two ways. First, while building filters, it reminds you that "`isSegment || isNetwork`" is not required on discovery filters, but must be present for topology and map filters. Second, when you're configuring filters to apply through NNM's user interface, the suffix will clue you in as to which filters to select for discovery application.

TIP To build accurate discovery filters, first allow NNM to discover everything and place it in the topology database. Then create your filter and set it up as a topology and map filter (_TMF). By applying this to the map, you'll see exactly what your proposed filter will be screening out.

Your alternative is to create a discovery filter, apply it at the first discovery, and hope you didn't screen out anything important by accident. You have no way to see what's happening if you do this.

First set up the collection domain for the IT department's own management station. Using the pattern filter named `EngrLan` in the default filter file, add a filter named `ITLan_DF`, which includes the IP address range for IT nodes (15.2.107-117):

```
ITLan_DF "IT's IP subnet" { ("IP Address" ~ 15.2.107-117.*) }
```

Reading that back, the name of the filter is `ITLan_DF` and can be referred to by filter expressions and user interface fields that ask for a filter name. It passes objects whose IP Address field in their SNMP data is between (is like) 15.2.107.* and 15.2.117.*.

Now build a filter expression (combination of filters) to turn that into a topology and map filter:

```
ITLan_TMF "IT's IP subnet" {("IP Address" ~ 15.2.107-117.*)|| NetsNSegs}
```

The addition of `NetsNSegs`, a standard NNM filter, passes all objects with SNMP data matching `isSegment` or `isNetwork`. A discovery filter includes these by default, but topology and map filters must add them explicitly. Networks and segments are required so that node objects will have containers in which to reside. Without them, your map database becomes corrupted.

Now is the time to use the tip above. First allow NNM to discover everything and place it in the topology database. Then apply the TMF filter to the map using `ovtopodump -f`, showing exactly what will be screened out.

The IT network management administrator sets up similar filters for `SalesLan`, `MfgLan`, `RDLan`, and `ESalesLan`. Here is the filter file so far (without all the comment lines):

```
Filters {
Networks "Any network" { isNetwork }
Segments "Any segment" { isSegment }
NetsNSegs "All networks & segments" { isNetwork || isSegment }
Nodes "Any node" { isNode }
IPRouters "Any IP Router" { isIPRouter }
Bridges "Any bridge" { isBridge }
Hubs "Any multi-port repeater" { isHub }
SNMPNode "Any node supporting SNMP" { isNode && isSNMPSupported }
HPNodes "Hewlett-Packard nodes" { isNode && ( vendor == "Hewlett-Packard" ) }
UBNodes "Ungermann-Bass Nodes"  { isNode && vendor == "Ungermann-Bass" }
NonHPNodes "Non-HP nodes" { isNode && ( vendor != "Hewlett-Packard" ) }
ITLan_DF "IT's IP subnet" { ("IP Address" ~ 15.2.107-117."*) }
SalesLan_DF "Sales's IP subnet" { ("IP Address" ~ 15.2.125-140.*) }
MfgLan_DF "Manufacturing's IP subnet" { ("IP Address" ~ 15.2.145-160.*) }
```

```
RDLan_DF "R&D's IP subnet" { ("IP Address" ~ 15.2.175-250.*) }
ESalesLan_DF "European Sales's IP subnet" { ("IP Address" ~ 192.5.32-50.*) }
}

FilterExpressions {
ITLan_TMF "IT's IP subnet map filter" { ITLan_DF || NetsNSegs }
SalesLan_TMF "Sales's IP subnet map filter" { SalesLan_DF || NetsNSegs }
MfgLan_TMF "Manufacturing's IP subnet map filter" { MfgLan_DF || NetsNSegs }
RDLan_TMF "R&D's IP subnet map filter" { RDLan_DF || NetsNSegs }
ESalesLan_TMF "European Sale's IP subnet map filter" { ESalesLan_DF || NetsNSegs
}
}
```

With these filters, each collection station can specify its own domain of interest with no overlap.

TIP

Build your overall filter file once and propagate it from your management station to all the collection stations. Use RCS for version control on the file and `rcp` to copy it to the collection stations. This way you know exactly what each collection station operator is looking at. You retain control over the file contents. And you can see a collection station's view of the network from the management station by applying the collection station's topology filter as a map filter on the management station.

Excluding Nodes

Take another look at the Sales LAN. It includes three laptop computers for sales people who are often on the road. The network operator really doesn't need to track whether those nodes are up or down, so write a DHCP filter that can exclude them:

```
SalesLaptops "Laptop units often offline in Sales
department" {"IP Address" ~ 15.2.135-140.*}
```

Now in a filter expression you can refer to `{SalesLan && !(SalesLaptops)}`. This is rather long, but it allows you to identify those laptops separately sometimes. Certainly, you could use this in a topology filter on the Sales collection station that would disable the status of these nodes being sent to the management station continuously.

You could also have specified `&& !("IP Address" ~ 15.2.135-140.*)` on the overall `SalesLan` filter if you knew you would never want the laptops included.

CAUTION

When negating the "is like" operator (~), place the negation *outside* of the expression. This is necessary due to NNM discovering all level 2 interfaces, including objects like serial cards which have no IP address. When no IP address is present, the object is assigned 0.0.0.0. If you negate the operator, as in `"IP Address" !~ 1.2.3.4`, you will inadvertently *include* all non-IP level 2 interfaces. By negating the expression, as in `!("IP Address" ~ 1.2.3.4)`, you get the set of all objects that *have* an IP address, and it is not like the one in the expression.

Failover Filters

What would happen if the Sales collection station failed? Assuming you have the management station configured for failover, all the nodes that the Sales collection station has in its topology database are polled by the management station. A failover filter would specify only the most critical nodes for which the management station would actually pick up status polling.

Sales is currently a fairly small department, so no filter is needed yet. When the Sales department grows, IT will probably want to limit which ones qualify for management failover.

Important Node Filters

Suppose the hub between the management station and the Sales collection station (actually, all the collection stations in this example) goes down. The management station will log an event that the hub is down, with secondary information that each of the nodes behind the hub is also unreachable. Getting that hub back online is a top priority, and most of the other stations are indeed secondary -- except that credit card authorization server. Access to that server is critical to business and the network operator needs to know about that immediately. By setting up an important node filter, you can assure that this node always qualifies as a primary event in the alarm browser.

```
SalesImptNodes "Critical nodes to have primary alarms if
Sales becomes unreachable" { "IP Hostname" == "ccserv" }
```

Similarly, the network management administrator creates important node filters for the other departments and adds them to the file.

Topology Filters

Looking at performance on the management station, IT realizes that they still see every node in the network -- it's as if they had never bothered with collection stations at all. Every bit of information is getting passed right on up to the management station.

Reviewing the Manufacturing LAN, they realize that it contains over 100 actual nodes, mostly PCs automating fabrication processes. At this point, PCs are not critical to your success, but the production servers from which they run their inventory and instruction data are vital. The collection station operator can monitor the availability of the PCs. All you need to monitor at the management station are the servers.

Use the set construct in the filter file to build your topology filter that only forwards information about the servers to the management station.

```
Sets{
MfgServersSet = { "parts", "prefab", "finalfinish",
"inventory" }
}
```

The previous filter expression for Manufacturing was `MfgLan_TMF "Manufacturing's IP subnet map filter" { MfgLan_DF || NetsNSegs }`. You still only want the Manufacturing domain, and you do want all the networks and segments. Use the set of servers to limit what gets forwarded:

```
MfgLan_TMF "Manufacturing's IP subnet map filter"
      { (MfgLan_DF && MfgServersSet) || NetsNSegs }
```

The collection station in Manufacturing uses `MfgLan_DF` to discover all nodes in its domain. The topology database on the collection station contains all these nodes. When the collection station starts to forward information to the management station, it applies `MfgLan_TMF` to the data stream. Only the servers in the `MfgServersSet` and the networks and segments pass the filter and get sent to the management station. The management station's topology database contains only this information and not the rest of the nodes on the Manufacturing LAN.

The network administrator in IT creates similar topology filters to reduce clutter coming in to the management station from the other collection stations.

Map Filters

The management station serves various operators in the IT department. Connie has received extensive training in troubleshooting connector devices, and specializes in them throughout the network. Sergai only cares about the servers for the various departments. Look at each of their map filters.

The filter expression from which Connie works looks like this:

```
Connectors "All connector devices"
      { Bridges || IPRouters || Hubs }
```

But there is a problem. IT's topology filter from Manufacturing, for example, only passes servers, networks, and segments. They'll have to modify the topology filter to give you the connector information first:

```
MfgLan_TMF "Manufacturing's IP subnet map filter"
      { (MfgLan_DF && MfgServersSet)
        || Connectors || NetsNSegs }
```

Now Connie needs a map filter which includes `NetsNSegs` for her map database to work:

```
Connectors_MF "All connector devices"
      { Bridges || IPRouters || Hubs || NetsNSegs }
```

With this map, Connie will see all the connecting devices from all the LANs.

Using the Network Presenter

With so many PCs on the site, IT hires Paul to keep them all up and running. But Paul's map on the management station has a problem. It says there are no PCs for him to manage. You have specifically filtered out the PCs before they reach the management station.

You could add all the PCs back in for all the topology filters. Paul would have one view with all the PCs and could do combined trend analysis. However, the management station topology database would get bogged down and everyone would have to filter them back out with their map filters.

You might think to purchase a separate management station for Paul and set up the collection station topology filters to only send PC information. However, topology filters are defined only once per collection station; a collection station cannot pass different data to different management stations.

As an alternative, Paul can use the Network Presenter through his web browser to act as a console on each of the collection stations. He would have more windows to watch, but network traffic would be minimized. Paul would be able to see the maps, but he would not be able to do any NNM administration through the Network Presenter since it is read-only.

This is an example of the trade-offs your organization will consider during your network management planning phase.

Default Filter File

The following is the default filter file, based on the above grammar:

```
//
// @(#)$OV_CONF/$LANG/filters
// @(#)HP OpenView NNM Pre-Release X.06.20 (Pre-Preview)  Jan 17 2001
// @(#)Copyright (c) 1990-2001 Hewlett-Packard Company
// $Revision: /main/PULSAR/PULSAR_REP/NEPTUNE/4 $  $Date: 2001/01/11 23:58 UTC $
//
// This is the default filter file.  These filters are examples
// which may be useful in your environment.  Feel free to modify this
// file and/or add your own.
// See OVfilterIntro(5) for more information on this file.
//

// Sets are a simple way to list string values to test
// against in a filter.  The "IN" operator tests a field value
// for membership in a set defined here.
//

Sets {

        // These are simple examples of sets.
        //
        // servers "Set of Servers" { "sv1", "sv2", "sv3" }
        // gateways "Backbone gateways " { "gw1", "gw2", "gw3" }
        //
        // sets can include a filename which contains the entries.
        // On Windows, the set file must be on the partition where NNM
        // is installed.

        // routers "Set of Routers" { /openview/conf/routers.set }
}

//
// Filters and FilterExpressions are used as Topology, Discovery,
// Map or Persistence filters.  Applications which use filters
// (like ipmap(1), ovtopmd(1m), ovtopodump(1), and ovtopofix(1m))
// can take the name of a Filter or a FilterExpression.
//

Filters {

        // The following filters are potentially useful as either Discovery
```

```
// or Topology filters, or components for building Map filters.
//
// NOTES CONCERNING DISCOVERY FILTERS:
// Networks and Segments are not directly testing with Discovery
// filters (and so there is no need to include
// "isNetwork || isSegment" in these filters).
// However, it does not hurt to include networks or segments in
// a discovery filter, so long as you realize that they will not
// be acted on.
//
// Since nodes are only discovered in managed networks, the network
// must already exist and be managed before the filtering test
// will be applied.  New networks connected to discovered nodes will
// be discovered unmanaged.   For discovery filters,
// a node and all its interfaces can be thought of
// as a single object. If any interface or the node itself passes the
// filter, the node and all of its interfaces will pass the filter.
//
// ovtopofix -f follows the discovery filter semantics and only
// applies to locally monitored objects.  However, if a network or
// segment is empty after the processing of the nodes with the -f
// option, that segment or network will also be removed.
//
// ovtopodump(1) also follows the discovery filter semantic, and
// the filter only applies to nodes/interfaces.
//
// NOTES CONCERNING MAP AND TOPOLOGY FILTERS:
// For these filters, all objects are subject to filtering.
// One must specify the networks and segments to be passed.
// Furthermore, objects are displayed/included only if all their
// parent objects (submaps) pass the map/topology filter.  For example,
// if no networks would pass the filter ("isNetwork" is not included),
// then all segments and non-gateways nodes would be filtered out.
// Like Discovery filters, if any interface or
// node passes a Map filter, the node and all of its interfaces
// also pass the filter.
//
// See the FilterExpressions section for map and topology
// filter examples.
//

Networks "Any network" { isNetwork }
Segments "Any segment" { isSegment }
Nodes "Any node" { isNode }
```

```
// The following filters are primarily for example purposes, though
// they might be useful as discovery filters.  They do not generally
// make sense as map or topology filters because they do not specify
// the inclusion of networks and segments.  They also make sense as
// parts of a filter expression.

Routers "Any Router" { isRouter }
IPRouters "Any IP Router" { isIPRouter }
Bridges "Any bridge" { isBridge }
Hubs "Any multi-port repeater" { isHub }
SNMPNode "Any node supporting SNMP" { isNode && isSNMPSupported }

// The next filter is useful for defining map and topology filters
// (see the FilterExpresssions section).  It is not
// particularly useful as a discovery filter
// since all networks and segments automatically pass Discovery
// filters.  Nor is it useful as a standalone
// map filter, since it doesn't pass any nodes.

NetsNSegs "All networks & segments" { isNetwork || isSegment }

// This filter specifies that only objects that support IP should
// be included.  This filter should be used as a topology filter on
// any collection station discovering non-IP information (Level 2
// or IPX only nodes or networks) that is forwarding to a
// NNM 4.X management station.  This will not completely remove
// all non-IP information (e.g. IPX interfaces on a node that also
// supports IP will come through), but will limit the impact on the
// management station, and make migration to NNM 5.X easier.

IPOnlyObjects "Only objects that support IP"
        { isIP || isSegment }

// The following filters are examples that you can customize to
// match your environment.

// LocalLAN "Nodes with interfaces on the 15.2.112-119 subnet"
//      { "IP Address" ~ 15.2.112-119.* }
// NeighborLANs "Node with interfaces on the neighboring LANs on site"
//      { "IP Address" ~ 192.1.2-10.* }
// HPNodes "Hewlett-Packard nodes"
//      { isNode && ( vendor == "Hewlett-Packard" ) }
// NonHPNodes "Non-HP nodes"
//      { isNode && ( vendor != "Hewlett-Packard" ) }

// These are example filters using the sets defined above.
```

```
// GatewaysSet "Any designated Gateway node"
//      { "IP Hostname" in gateways }
// ServersSet "Any designated Server node"
//      { "IP Hostname" in servers }

// The next filter is useful for defining map and topology filters
// (see the FilterExpresssions section).  It is not
// particularly useful as a discovery filter
// since all networks and segments automatically pass Discovery
// filters.  Nor is it useful as a standalone
// map filter, since it doesn't pass any nodes.
//
// The network name below assumes an entry in /etc/networks.
// If no such entry exists, the actual network name will be
// something like "15.2.112".
// MyNet "The network in which I'm interested"
//      { isNetwork && "IP Network Name" == "yellow-lan" }

// This filter would be an example of a way to pass all objects
// contained in the subnet 15.2.112.0 with a subnet mask
// of 255.255.248.0 (so the subnet range is 15.2.112.0 to
// 15.2.119.255).  Note that the inclusion of "isSegment" is
// not a concern, because all segments outside of this network
// would be filtered out because the networks containing them
// were filtered out.  We must have some specification of segments
// if we want the segment contents to be included.

// EngrLan "The 15.2.112 subnet used by engineering"
//      { ("IP Address" ~ 15.2.112-119.* ) || isSegment }

// This filter accomplishes the same thing, but is more restrictive
// in the specification of segments to only include segments in that
// engineering network.  This assumes there is an entry in /etc/networks
// such that the subnet gets the name "engr-LAN", and hence the segments
// in that network all have a name of form "engr-LAN.<something>"
// Generally, the above will accomplish what you want, but this
// may be necessary in some situations if you want specific segments.
// In particular, it works well if you want to use this to EXCLUDE
// the objects in this network.  See the example in the Filter
// expressions below.

// EngrLan2 "The 15.2.112 subnet used by engineering"
//      { ( "IP Address" ~ 15.2.112-119.* ) ||
//        ( "IP Segment Name" ~ "engr-LAN.*" ) }
```

```
        // The following are some sample DHCPFilters.
        // They can be specified by using the Network Polling Configuration
        // dialog application, xnmpolling, on the Status Polling page.

        // DHCPSubnet "Addresses in subnet 15.70.177 are DHCP allocated"
        //      { "IP Address" ~ 15.70.177.* }

        // DHCPNode  "A particular address is DHCP allocated"
        //      { "IP Address" ~ 15.70.177.9 }

        // DHCPRange "IP Addresses from 15.70.177.5 through 15.70.177.10"
        //      { "IP Address" ~ 15.70.177.5-10 }

        //
        // Access to various nodes based on SNMP MIBs that have
        // been discovered as supporting a particular MIB
        //

        CDPnodes "Nodes supporting Cisco Discovery Protocol" { isCDP }
        ATMnodes "Nodes supporting ATM" { ("SNMP ifType" == "ATM")  ||
            ("SNMP ifType" == "atmLogical") ||
            ("SNMP ifType" == "atmDxi") ||
            ("SNMP ifType" == "atmFuni") || ("SNMP ifType" == "atmIma") }
        DS1nodes "Nodes supporting DS1" { "SNMP ifType" == "T-1 Carrier (ds1)" }
        DS3nodes "Nodes supporting DS3" { "SNMP ifType" == "T-3 Carrier (ds3)" }
        SONETnodes "Nodes supporting SONET" { "SNMP ifType" == "SONET" }
        FrameRelayNodes "Nodes supporting Frame-Relay" {
            ("SNMP ifType" == "Frame Relay") ||
            ("SNMP ifType" == "Frame Relay Service") ||
            ("SNMP ifType" == "frameRelayInterconnect") }
        RMON1nodes "Nodes supporting RMON-1" { isRMON }
        RMON2nodes "Nodes supporting RMON-2" { isRMON2 }
        // Short-hand for any RMON device
        RMONnodes "Nodes supporting RMON-1 or RMON-2" { isRMON || isRMON2 }
}

// FilterExpressions are simply combinations of filters defined in the
// same filter file (above).  FilterExpressions make it simple to
// combine filters without reproducing the expressions of each
// filter again

FilterExpressions {

        // One can turn the filters defined above into viable map or
        // topology filters by simply adding "|| NetsNSegs".  (Doing so
        // does not invalidate the filters as discovery
        // filters.  It just adds a superfluous test.)
```

```
NetInfrastructure "Any network connecting device and what they connect"
        { Routers || Bridges || Hubs || NetsNSegs }
NetBackbone "Networks and gateways/routers"
        { Routers || Networks }

// The following combines the two set filters
// defined above into one FilterExpression.
// It works unmodified as a discovery filter.
// To work as a map filter, network and segment filtering
// must be added (see below).

// VitalNodes "All Gateways and Servers"
//      { GatewaysSet || ServersSet }
// One can turn the filters defined above into viable map or
// topology filters by simply adding "|| NetsNSegs".  (Doing so
// does not invalidate the filters as discovery
// filters.  It just adds a superfluous test.)

// VitalNodesMap "All nets & segs, but only gateway and server nodes"
//      { GatewaysSet || ServersSet || NetsNSegs}
// LocalLANView "All nets & segs, but only local nodes"
//      { LocalLAN || NetsNSegs }

// Using the filters defined above that include only a specific
// network, we can also exclude the specific network like this
// Note the use of the more specific form to exclude only the segments
// in the engineering lan.  This could have been specified directly
// as a negation in the filter part, but this form works well if you
// have several networks to manipulate in this manner.
// EverythingButEngr "Everything but the engineering LAN"
//      { !EngrLan2 }

// Of course the above filter expressions, when used as
// map filters, pass all networks and segments.  You
// may wish to see only a particular network.  The following map
// filters accomplish this.  Note that though segments
// and nodes from other networks will pass the filters, IP Map
// will ignore them because their parent networks will not pass.
// NOTE:  These filters will not work as Discovery
// filters because all network and segments automatically pass
// Discovery and Topology filters.

// MyNetMap "Only the network of interest and all its constituent parts"
//      { MyNet || Segments || Nodes}
// MyVitalNodesMap "Gateways, servers and segments in net of interest"
//      { MyNet || Segments ||  GatewaysSet || ServersSet }
```

```
        // This is a map persistence filter which ensures that
        // all Ungermann-Bass are kept in memory and up to date.
        // Note that this will also keep any containing submaps in memory.

        // PersFilter "Objects to keep in map memory"
        //      { HPNodes }
}
```

B Using NNM Under Mixed and non-Mixed Codeset Environments

This chapter covers requirements for using NNM's alarm browser, `xnmevents`, in a mixed or non-mixed codeset environment. These steps are not required if you are using the web interface to alarms, `jalarms`.

Using Identical Codesets

Event forwarding in NNM's distribution and scalability model expects that the management station and the collection station use the same codeset. If this is the case in your environment, no further action is required.

For example, suppose a management station is operating under the ISO8859-1 codeset. If a collection station forwards events to this management station using the ISO8859-1 codeset, no action is required for event forwarding.

Using Mixed Codesets

The shared codeset restriction is minimal for European languages, Korean, and T-Chinese, because NNM is not localized to those languages. However, this restriction becomes a problem for the Japanese localized NNM users that may need to use both Japanese EUC and Shift-JIS codesets or S-Chinese localized NNM users that may need to use both GBK and GB2312 codesets. The following examples show how this problem can occur.

Example B-1 Japanese EUC and Japanese Shift-JIS Mixed Codeset

Suppose a management station is running under Japanese EUC, but one of its collection stations is running under Japanese Shift-JIS, the management station's GUI will fail to display the Shift-JIS data obtained from a Shift-JIS collection station. This is because a management station assumes that the data from the collection stations are using the same codeset as itself, and parses and displays all data (including the Shift-JIS data) as Japanese EUC data. This results in a data corruption of all Shift-JIS characters.

Example B-2 S-Chinese GBK and GB2312 Mixed Codeset

If you are using S-Chinese codesets, the GB2312 codeset is a subset of the GBK codeset. If you are using GBK codeset on a collection station, only the characters in the GB2312 ranges of the GBK codeset will be visible without corruption on the management station. For best results, send characters from a collection station using the GB2312 codeset to a management station using the GBK codeset.

If you are using S-Chinese codesets in a Windows environment, remember that the Windows environment uses the GBK codeset exclusively. In this scenario, you have two options to choose from:

1. If you are running an NNM collection station in a Windows environment, the management station can be a UNIX system running the GBK codeset.
2. If you are running an NNM collection station in a Windows environment, the management station can be a system running a Windows operating system.

NOTE For best results, do not send characters from a collection station using the GBK codeset to a management station using the GB2312 codeset.

To summarize, in this mixed codeset environment, the collection stations must be configured to forward event data to the management station using ASCII only, and furthermore, "user events" must not contain non-ASCII characters. Since ASCII is a subset of all other codesets, data from the collection station will not be corrupted by the management stations' GUI.

Although SNMP traps do not prevent you from using non-ASCII characters in order to support mixed codeset environments, it is recommended that you do not forward events that contain non-ASCII characters. Avoid generating user-sendable events containing non-ASCII characters, if these events are forwarded from a collection station to a management station. For example, do not send events, such as the "`OV_Popup_Message`" or "`OV_Message`," with non-ASCII characters, using the `snmptrap` command from a collection station to a management station.

Japanese system owners should refer to Table B-1 on page 213 and S-Chinese system owners should refer to Table B-2 on page 214 when configuring a collection station to operate under a mixed-codeset environment. To configure a collection station to operate under a mixed-codeset environment, take the following steps:

1. Create a new file called `$OV_CONF/ovlang.conf` (for example, `C:`*`install_dir`*`\conf\ovlang.conf`) that contains the following two entries, each on a separate line:

   ```
   ovtopmd

   netmon
   ```

   If you need to forward threshold events, add `snmpCollect` to `ovlang.conf`.

2. Restart the `ovw` processes by executing, as `Administrator` or `root`, the following commands:

   ```
   ovstop

   ovstart
   ```

NOTE There is no need to remove or clear the `ovtrapd.log` file of a management station, but if it contains data from this collection station prior to this step, that data may be corrupted.

The first two steps will force the collection station to generate data forwarded to the management station with ASCII characters only. However, data that is not forwarded to a management station will continue to operate using the codeset specified by you or by the environment.

Table B-1 Mixed Codeset Environment for Japanese Systems

Collection Station's System and Codeset	Management Station's System and Codeset			
	Windows SJIS	HPUX SJIS	HPUX EUC	Solaris EUC
Windows SJIS	(1)	(1)	(2)	(2)
HP-UX SJIS	(1)	(1)	(2)	(2)
HP-UX EUC	(2)	(2)	(1)	(1)
Solaris EUC	(2)	(2)	(1)	(1)

(1) -- this is not a mixed codeset environment; no configuration necessary.

(2) -- this a mixed codeset environment; configure the collection station to forward ASCII data to the management station.

Table B-2 Mixed Codeset Environment for S-Chinese Systems

Collection Station's System and Codeset	Management Station's System and Codeset		
	Windows GBK	Solaris GBK	Solaris GB2312
Windows GBK	(1)	(1)	(2)
Solaris GBK	(1)	(1)	(2)
Solaris GB2312	(1)	(1)	(1)

(1) -- this is not a mixed codeset environment; no configuration necessary.

(2) -- this a mixed codeset environment; configure the collection station to forward ASCII data to the management station.

Glossary

A

agent In the network management model, the agent is the component that "instruments" or provides management information about a particular network device, such as a file server, to a network management system. *See also: proxy agent*.

C

client A computer system, on a network, that accesses a service from another computer (server). A client may run any arbitrary operating system.

client/server A concept that functionally divides the execution of a unit of work between activities initiated by an end user or program (client) and resource responses (server) to the activity request.

collection domain The set of nodes and other objects for which trend data collection and threshold checking is occurring on a collection station. Frequently the discovery domain and the collection domain include the same set of nodes and objects. Discussions in this document assume that is the case, and use this term to refer to both. *See also: management domain, discovery domain.*

collection station A management station that acts as a collection point for information being gathered on managed elements. *See also: management station.*

correlated events Multiple events analyzed by NNM and presented under a single event in the browser.

D

discovery domain The set of nodes and other objects actively being discovered and monitored by an NNM station.

discovery filter Specifies which devices an NNM station is actively discovering and monitoring. *See also: filtering.*

F

failover Configuring a management station to pick up status polling duties for a collection station that goes into critical status.

failover filter List of specific nodes the management station should poll, leaving the status of the rest as they were when the collection station went critical.

filtering Refers to the process of limiting the object the user is interested in having discovered, monitored, displayed, etc., by determining certain attributes that are of interest to the user. Filtering generally is used to limit processing to a subset of information already available, to reduce the cognitive load on the user, align presentation with the users needs, and generally reduce system load by filtering the amount of data handled as soon as possible. *See also: discovery filter, failover filter, map filter, persistence filter, and topology filter.*

I

IP address The Internet Protocol address is a network level address assigned to each system is a TCP/IP network. It is 4 bytes long. An example IP address is 192.215.15.46.

M

management application Software that obtains and presents information from agents, or proxy agents. The information is usually presented in a graphical user interface.

management console An instance of the user interface from which you can control applications that do network and system management. The console may be on the system that contains the management software or it may be on another system in the management domain.

management domain The set of all nodes and other objects in the network that are of interest to the user.

management station The computer system used in the management of other objects in the network, system, and software environment. Management stations run network or system management services, daemons, processes, and/or management user interfaces. Management clients, management servers, management consoles, and stand-alone management systems are all examples of management stations. *See also: collection station.*

map A set of related objects, symbols, and submaps that provides a graphical and hierarchical presentation of your network and its systems. Because a map is defined by the collections of submaps it comprises, you never view a map directly, but rather one or more of its submaps. Maps are derived from the HP OpenView Windows object database, and multiple maps can be created from the same object database. *See also: submap.*

map filter Defines what objects in the HP OpenView Windows object database will be permitted on the map. Objects that are rejected by a map filter remain in the management station's topology database, and are still subject to status polling. Map filtering occurs on a per-map basis, not a per-display basis. *See also: filtering.*

N

node A termination point for two or more communication links - or more simply, a device on a network. The node can serve as the control location for forwarding data among the elements of a network, as well as perform other networking, and in some cases local processing functions.

O

object A computer system, interface card, printer, or any other device or object that appears in the network.

P

persistence filter Specifies one or more objects which are required to be in memory at all times, and therefore are not permitted to be on a transient submap. *See also: filtering, persistent submap, persistent object.*

persistent object An object which has passed a persistence filter, and is therefore in memory whenever the map on which it resides is open. Any submap that contains a persistent object is automatically a persistent submap. *See also: persistent submap, persistence filter.*

persistent submap A submap that is always in memory, either because it is above the configured persistence level, because it

contains a persistent object, or because it is the child of a persistent object. *See also: transient submap.*

primary collection station When applied to a particular object, this refers to the collection station which has been chosen as having primary responsibility for monitoring the object. In particular, it is used to determine the status and other attributes of the object. A given object may be reported and monitored by multiple collection stations, but only one station will be the primary station. *See also: secondary collection station.*

proxy agent A proxy agent performs information preparation and exchange on behalf of a device it is representing. For example, an agent is a proxy if it is located on the file server but is representing a client. *See also: agent*.

R

root submap A standard, top-level submap for every map.

router A router uses a packet's destination address to determine which network or network segment is its destination.

S

server A computer system that provides services to other computer systems on the network. A server could be a file server and provide data storage, application, routing, and print services.

secondary collection station When applied to a particular object, refers to a collection station that is known to be monitoring the object, but which has not been chosen as the primary collection station.

site A group of nodes connected completely by a Local Area Network (LAN), i.e. all at the same geographic location.

SNMP Simple Network Management Protocol. A protocol running above TCP/IP used to exchange network management information. SNMPv2C has extended functionality over the original protocol.

station *See: workstation, collection station, or management station.*

submap A particular view of the network environment, consisting of related symbols that are displayed in a single window. Multiple submaps can be displayed simultaneously. Submaps are typically (though not necessarily) organized in a hierarchy, with the root submap at the top of the hierarchy. *See also: map.*

T

TCP/IP Transmission Control Protocol/Internet Protocol. A set of protocols for Layers 3 and 4 of the seven-layered Open Systems Interconnect network model

topology filter A topology filter at a collection station defines the subset of local topology data that management stations can see. *See also: filtering.*

transient submap A submap that is not necessarily in memory at all times, but which is available "on-demand" when the operator opens it. Any submap that can be recreated at any time based wholly on topological information is potentially a

transient submap, depending on how and whether the on- demand submap feature is enabled. *See also: persistent submap.*

W

workstation A computer or terminal that is connected to the network.

Index


invent


Java™
COMPATIBLE