

# NNM 7.51 Extended Topology and APA Deployment Handbook

Version 1.3

30 August 2007

NNM-Team, Bangalore and Fort Collins

Please send feedback and questions to  
[nnm.docs@hp.com](mailto:nnm.docs@hp.com)

## Table of Contents

1	What's new in this revision .....	4
2	Introduction .....	4
3	Conventions .....	4
4	Acronyms and definitions .....	4
5	Machine Sizing .....	5
6	Database .....	5
7	Background on ET .....	6
8	Enabling ET .....	6
9	Tuning ET Discovery .....	6
9.1	Zones .....	6
9.2	Autozone Tool .....	7
9.3	Manual Zones .....	8
9.4	Testing All Zones .....	11
9.5	Running ET Discovery .....	13
9.6	Incremental Node Discovery .....	13
9.7	Cisco Discovery Configuration .....	14
9.8	Interface Filtering .....	16
10	Tuning APA .....	18
10.1	APA and Netmon .....	18
10.2	APA Architecture .....	18
10.2.1	Collapsing Stations .....	19
10.2.2	Using OVO as the Manager of Managers (MOM) .....	20
10.2.3	Using NNM MS with netmon and NNM CS with APA .....	20
10.2.4	Using two NNM stations in a "warm standby" scenario .....	21
10.3	Enabling APA .....	21
10.4	Measuring APA performance .....	24
10.5	Concepts of Tuning .....	24
10.6	The simple way out .....	26
10.7	Understanding APA polling policy matching .....	26
10.8	How does APA "poll" .....	28
10.9	Out of the box defaults .....	28
10.10	What if a node is both a switch and a router? .....	28
10.11	Some HP supplied policies .....	29
10.12	A quick lesson on reading the Class Specification (polling policy) in paConfig.xml .....	29
10.13	Example using User Configurable Filters .....	31
10.14	A new valuable addition to ovet_demandpoll.ovpl .....	38
10.15	Another suggested polling policy .....	39
10.16	Example of changing polling policies for specific node types .....	39
10.17	Improving APA startup time .....	41
10.17.1	First Improvement .....	41
10.17.2	Second Improvement .....	42
10.17.3	Third Improvement .....	42
10.18	Changing the thread count in APA .....	42
10.19	Reducing the memory footprint in APA .....	43

10.20	Side effects of islands of connectivity .....	44
11	Reducing Traps and Events .....	45
12	Backup Strategies.....	50
13	GUI.....	51
13.1	Simple Changes .....	51
13.1.1	Changes to web.xml .....	51
13.1.2	Changing the java heap size.....	52
13.1.3	Changing Topology Cache .....	52
13.2	Container Views.....	53
13.3	Advanced Container View Tasks.....	55
13.3.1	Delete Container.....	55
13.3.2	Rename Container .....	56
13.3.3	Scaling Background Graphics.....	56
13.3.4	Resetting Containers.....	59
13.4	Adding Authentication to HomeBase .....	59
14	Filters in ET .....	61
14.1	Introduction to filters.....	61
14.2	Structure of TopoFilters.xml .....	61
14.3	Creating, Modifying and Verifying Filters.....	62
14.4	Assertion Types .....	62
14.5	Filter Examples .....	64
15	Appendix A – A new and improved Swouter filter .....	67

# 1 What's new in this revision

While there are many small changes throughout the text, here are the major changes in revision 1.3 in no particular order:

- A better “swouter” solution.
- Additional APA Architecture
- Additional ways to improve APA startup time
- chatr commands for HP Itanium Platform
- Dealing with "topology islands" in APA
- Additional GUI configuration changes

## 2 Introduction

This document is intended as a handbook to help customers properly deploy the Extended Topology and APA features of NNM 7.5x.

This document is a collection of "Best Practices" we have gathered while working with customers. It is not meant to be a complete deployment guide but customers and integrators should find it useful.

This paper assumes you know how to get “classic NNM” (including OVW, netmon, snmpCollect, etc.) up and running. All emphasis will be on the configuration and deployment of the more modern features including Extended Topology (ET) and Active Problem Analyzer (APA). We will also be specifically addressing NNM 7.51. As of this writing, this is the latest version of NNM. NNM 7.51 released many new features that are documented via whitepapers. We recommend you to read those as well.

We won't be covering specific technology in APA and ET. We won't describe the technical way they work. Some very good presentations have been given on the technology behind these features at OpenView forums and HP Software Universe. We can get you a copy of these presentations if you are interested. This is more of a document about deploying and tuning rather than understanding the technology.

## 3 Conventions

Throughout this paper, we will give commands and files in their UNIX style like \$OV\_BIN/ovstop. You can easily make the translation for Windows to %OV\_BIN%\ovstop.

## 4 Acronyms and definitions

- NNM = Network Node Manager
- ET = Extended Topology
- APA = Active Problem Analyzer
- NNM Classic = Refers to original NNM topology (netmon discovery, loadhosts, autodiscovery, netmon polling, etc.)
- NNM SE = Same as NNM Classic. No ET. No APA.
- DIM = Distributed Internet Monitoring

- MS = Management Station
- CS = Collection Station
- MOM = Manager of Managers
- OVO = OpenView Operations
- ovet\_poll = the actual process name for APA
- ovas = the NNM Application Server for the User Interface
- BES = Binary Event Store

## 5 Machine Sizing

A common question we get asked is, “How much computing power is required for ET and APA?” We don’t have a formula but we can give a suggestion for large environments. What do we consider a “large environment” for one NNM station?

We would say:

- more than 2500 switches and routers
- more than 100,000 interfaces

For a large environment, we recommend the following but stress that this is not a requirement, but just a recommendation.

- 4-CPU box
- CPU speed is very important
  - 3 GHz CISC
  - 1 GHz RISC
- RAM also important
  - at least 2 Gig RAM

Another suggestion, on Windows especially, is to not install NNM on the same partition as the Operating System. If you have a C drive and a D drive, put NNM on the D drive.

## 6 Database

Most large scale NNM solutions are using the built in database called Solid. You can use Oracle but it is not required. The NNM solution has more database hints for Solid so performance should be good on Solid.

When using Solid, we also recommend the following steps. We typically see at least a 10% performance improvement.

- Always move the Solid database off the OS drive
- Use RAID 0 or some other RAID technology for the Solid drive

The final and probably most important step is to follow the guidelines in the 7.51 whitepaper called Solid Performance Improvement. Not all of the settings in that document are setup automatically when you install 7.51 so check to make sure they are all set. The setting ForceThreadsToSystemScope is specific to Solaris and must be added manually after a 7.51 installation.

Note: the settings in that paper can be applied to Solid on 7.5 so even if you haven't installed 7.51, get a copy of the whitepaper and follow the steps by hand.

## 7 Background on ET

We are now going to assume you already have an “NNM Classic” topology in place. This means that you have all your nodes discovered either by auto-discovery, loadhosts, or added to map and these nodes are monitored by netmon. If you need further information about setting up classic NNM, see the *Managing Your Networks* manual.

Let us give a little bit of background on what ET does. ET does not discover new nodes. It discovers the “extended topology”. This includes layer 2 connectivity, HSRP, VRRP, VLANs, and other information. This topology is then used by APA to monitor the network. ET does discrete discoveries which tend to last anywhere from 4 hours to a full day.

What does ET get from classic NNM? Each ET discovery gets a list of managed nodes and a mapping of MAC addresses to IP addresses for nodes in the NNM topology. Unmanaged nodes are not brought into ET. But “netmon unmanaged” interfaces are not respected in APA and may or may not be monitored by APA.

## 8 Enabling ET

It is simple to enable ET. Run `$OV_BIN/setupExtTopo.ovpl` from the command line. It will ask you some licensing questions. Depending on the size of your network, the script may prompt you to automatically determine and configure zones. We suggest answering “Yes” to this if prompted.

## 9 Tuning ET Discovery

### 9.1 Zones

The concept of zones is discussed in the ET manual provided by HP (*Using\_Extended\_Topology.pdf*). Please take the time to go read Chapter 2 and Appendix A in this document before going further in this handbook.

We'll give a simple summary of zones here. “Zones” is not a network term. This is an NNM ET term. The way ET discovery works is that as nodes are interrogated, their information is stored in an “in-memory” database. At the completion of discovery, the in-memory database is deleted. The final ET topology is stored into Solid database. In order to make ET discovery more scalable, it uses a divide-and-conquer approach to segment the network into smaller pieces. Zones are used represent a logical division in the network. ET will discover each zone separately and then merge the information together at the end. If a node is in multiple zones, it will be interrogated in each zone discovery.

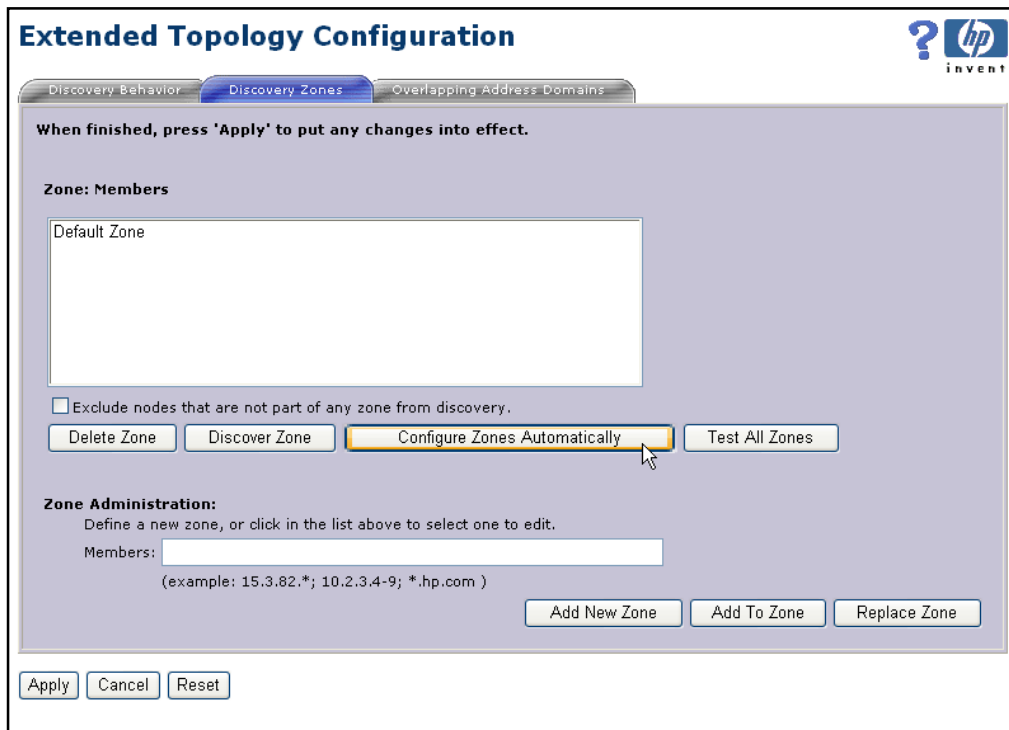
Zones can be created manually by the user or automatically by NNM. There are pluses and minuses to each approach. We will list some here. We recommend creating manual

zones as we feel these typically lead to a faster discovery with better accuracy in the connectivity.

## 9.2 **Autozone Tool**

You can have NNM create zones for you. These zones are strictly based on Layer 3 information from the classic NNM topology. The advantage to using this tool is that you don't have to think about it much. The disadvantage is that it can create less than ideal zones and break up key node connectivity. A lot of that is because on many networks, the IP address of a switch doesn't imply a lot of information about the layer 2 connectivity. A second disadvantage to using autozone is that it must be re-run before each discovery if you have added new nodes since the previous discovery. Most customers don't realize this. Since autozone is very specific about putting nodes into zones, if you add additional equipment since the last run of autozone, the new equipment will be put into the "default zone". That's usually a poor place for nodes to go. And if you are not careful, you'll end up with too many nodes in the default zone. The third disadvantage to the autozone tool is that it tends to put more nodes than necessary into multiple zones thus causing slower discoveries.

Autozone can be run from the User Interface or from the command line. To run it from the User Interface, go to the "Extended Topology Configuration" page and click on "Configure Zones Automatically".



To run it from the command line, simply execute `$OV_BIN/autozone.ovpl`. No arguments are required. In both cases, the log file `$OV_PRIV_LOG/autozone.log` is

generated. This log file is written during the autozone execution so you can “tail” the file if you wish to monitor progress. The autozone tool can take a long time to execute.

You might consider setting up autozone.ovpl to run as a cron job or scheduled task so that it executes before running an ET discovery. If you do this, you should restart the ovas process after running autozone.ovpl.

```
ovstop ovas
ovstart ovas
```

The ovas process is the GUI Application Server for NNM. You need to restart it after running autozone.ovpl via command line because the GUI doesn't re-read the autozone results when autozone.ovpl is run from the command line. If you don't restart ovas, it can cause some issues showing accurate progress in the Discovery Status bar and the Test All Zones button. So if you run autozone.ovpl via command line, restart ovas as well. This is not true for when you click the “Configure Zones Automatically” button. When you autozone this way, the GUI is aware of the change and adjusts accordingly.

There is one additional change you can make with autozone but use this one very cautiously. If you have a very powerful computer, you might consider allowing more nodes and interfaces to go into a single zone. There is a file called \$OV\_CONF/nnmet/recVals.conf. In this file, there is ONLY ONE VALUE that you should ever change. That is the numManagedObjects value. DO NOT change the numManagedNodes file. DO NOT change any of the other values either. The only value you can consider changing is the numManagedObjects. This number represents the maximum number of objects that are put into a single zone. If you have a very high end computer, you might consider increasing this value to 10000. We don't recommend going higher than that. After you change this value, you can re-run autozone.ovpl or rerun it via the GUI.

```
# Note: numManagedObjects (below) refers to the maximum number of
# managed nodes + managed interfaces that can be handled safely in a
# single zone. It is calculated when setupExtTopo.ovpl is run, based
# upon system resources of the management station. The value here can
# be adjusted. Note that it will be recalculated and overwritten the
# next time setupExtTopo.ovpl is run. Note that making the number too
# large will result in the Automatically Configure Zones option in the
# Extended Topology Configuration GUI creating larger zones, which may
# be too large for a system to handle or degrade performance. The
# creation of fewer, large zones may be easier than many smaller zones.
# Making the number too small will require more zones, and will mean that
# zone creation (either by hand or automatic) may take longer and involve more
# decisions about splitting a switch fabric into different overlapping zones.
# This number is also used by the "Test All Zones" button to generate
# warnings when a configured zone or candidate zone exceeds this size.
numManagedObjects : 6000
```

### 9.3 *Manual Zones*

As mentioned earlier, this is our preferred method. But it's only realistic to use this method if you adhere to a naming scheme that includes location information or if you can



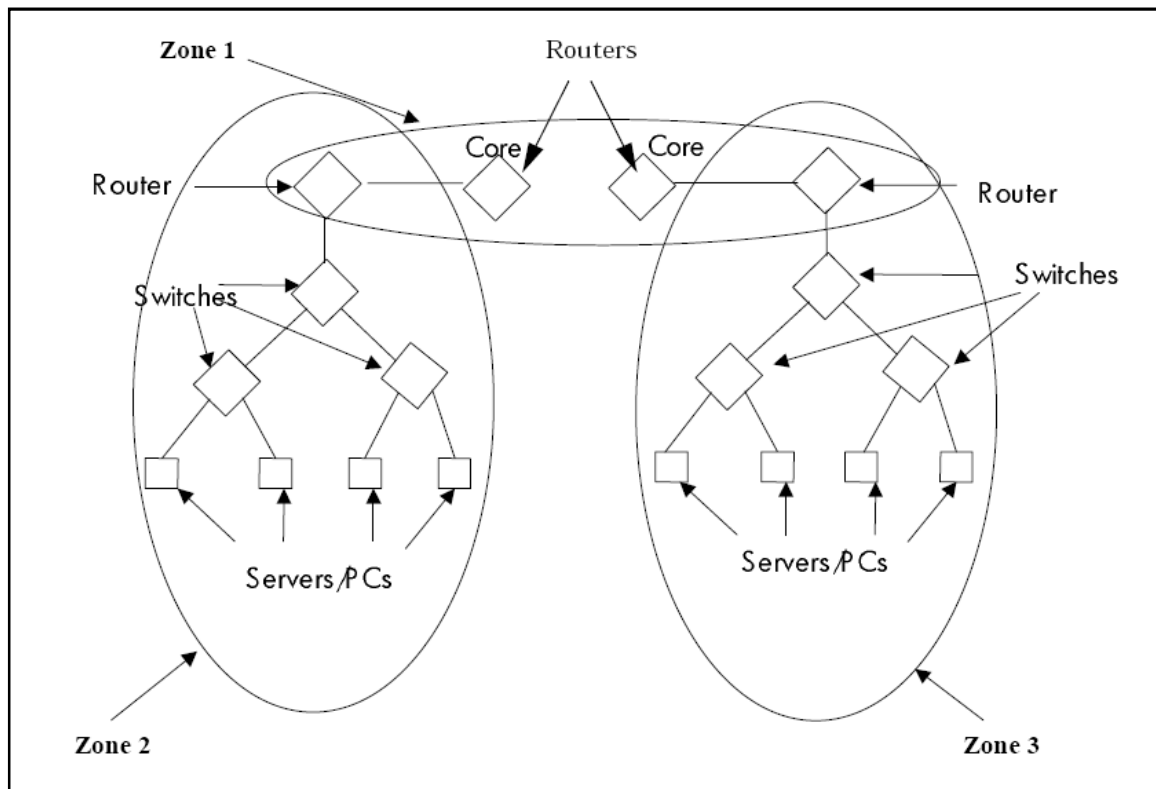
separate your geographies based on IP ranges. Let us explain more about zones and then we'll cover how to create them but again, this is covered more thoroughly in the Using Extended Topology Manual.

An ideal zone is a bunch of equipment that is connected via Layer 2 and Layer 3 and “stands alone” meaning it contains all the Layer 2 gear that is directly connected. At most customer sites, the best mapping of a zone is a building. Many customers “route” between buildings. You should separate zones at layer 3 boundaries. You also should connect up the zones by having at least one router from each zone in another zone.

You should not separate the following equipment into different zones:

1. Directly connected switches
2. Switches connected to routers
3. Switches in the same VLAN (that is, having the same global VLAN names or IDs)

An example from the ET manual is shown here. In this sample there are three zones. Zone 2 and Zone 3 would typically be a building. Notice how this example has one router from Zone 2 and 3 that is also in Zone 1. We know this is very simplistic but it should help you understand the concepts.



Now let us show an approach to specifying these zones. Many customers use a naming convention to specify location and also device role. Maybe the naming convention is

wwxyyzzz where ww is city, xx is building number, yy is device type like switch or router and zzz is device role like core, distribution or access. So as an example, let's say you have two locations, Fort Collins (FC) and Loveland (LV). Let's also suppose we support a few device types like Switch (SW), Router (RT), Server (SV), Firewall (FW), and Load Balancers (LB). Of course the list could go on and on. And last, let's suppose you have some roles like Core (COR), Distribution (DIS), and Access (ACC).

With this information, we could decide that we want to build the zones using the following expression. Let me assume that the router in zone 2 and 3 above that is shared between zones is identified as a distribution router.

FC01\* (this will be zone 2 in the picture above)  
 FC02\* (this will be zone 3 in the picture above)  
 FC????DIS, \*COR (this will be zone 1 in the picture above)

The good news is that manual zones support limited pattern matching expressions. It's not a full regular expression, so do some experimenting with it before you go too far. Here is the usage sheet on the pattern matching. Notice the use of the NOT "!" in the brackets and also notice that you can match a "-". A common mistake happens with the asterisk. The asterisk only matches up to the first period. So in my example above, FC01\* would match FC01SWACC but not FC01SWACC.corp.com. You might need to make the wildcard show FC01\*.corp.com instead.

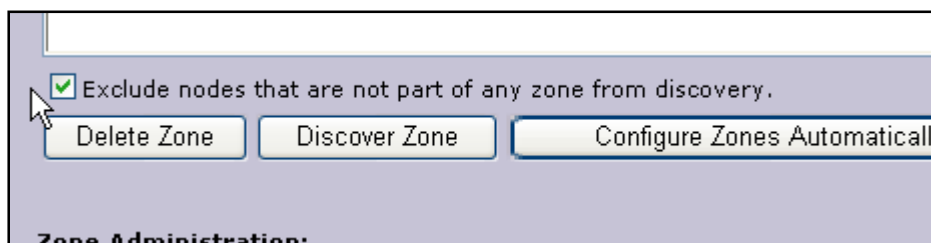
NOTE: We do some validation of the wildcarding in the GUI when you hit Add New Zone, Add To Zone, and Replace Zone. Sometimes the GUI is more restrictive than really necessary. If you feel brave, you could try editing the XML file directly as that is a little more lax on the wildcarding rules.

Character	Usage
Asterisk *	Use an asterisk to represent any number of characters up to the next period: <ul style="list-style-type: none"> <li>• *.corp.com matches pc.corp.com or ws.corp.com.</li> <li>• pc.*.com matches pc.corp.com or pc.location.com.</li> <li>• *.* matches corp.com or pc.com, but not pc.corp.com. The * in 10.*.1.3 matches any number.</li> </ul>
Question mark ?	Use to match a single character: <ul style="list-style-type: none"> <li>• pc.c?.com matches pc.ca.com, pc.cb.com, or pc.cc.com, but does not match pc.cal.com or pc.c.com.</li> <li>• pc.???.com matches pc.abc.com, pc.bcd.com, or pc.cde.com, but does not match pc.ab.com or pc.abcd.com.</li> </ul>
Brackets [...]	Use to match single characters, characters within a range, or characters not within a range:

	<ul style="list-style-type: none"> <li>• <code>[bcf]an.corp.com</code> matches <code>ban.corp.com</code>, <code>can.corp.com</code>, or <code>fan.corp.com</code>, but does not match <code>dan.corp.com</code> or <code>lan.corp.com</code>.</li> <li>• <code>[b-d]an.corp.com</code> matches <code>ban.corp.com</code>, <code>can.corp.com</code>, or <code>dan.corp.com</code>, but does not match <code>fan.corp.com</code>, <code>an.corp.com</code>, or <code>clan.copr.com</code>.</li> <li>• <code>[!d-z]an.corp.com</code> restricts the selection to <code>aan.corp.com</code>, <code>ban.corp.com</code>, or <code>can.corp.com</code>.</li> </ul>
Dash n-n	<p>Specify a range of IP addresses.</p> <ul style="list-style-type: none"> <li>• <code>10.2.1-3.1</code> represents <code>10.2.1.1</code>, <code>10.2.2.1</code>, or <code>10.2.3.1</code></li> </ul>

If you prefer to edit XML directly, you can edit the file `$OV_CONF/nnet/etconfig.xml`. Make sure you create a backup before editing the file. Also note that the GUI server (ovas) does not know if you have edited the file by hand. In order to see the changes, you need to restart ovas. There is no command line for testing zones so you must restart ovas and test the zones via the GUI.

Let me give a final recommendation about working with manual zones. I suggest that you try discovering a couple of zones that are fairly representative of the network. Here's a way to do that. Make a backup copy of `$OV_CONF/nnet/etconfig.xml`. Then use the ET Configuration page to delete all but a couple of zones. Next be sure to select the checkbox "Exclude nodes that are not part of any zone from discovery". Because we deleted most of the zones, the majority of nodes will be in the default zone. By checking this box, it tells ET to only discover nodes that are in the few defined zones.

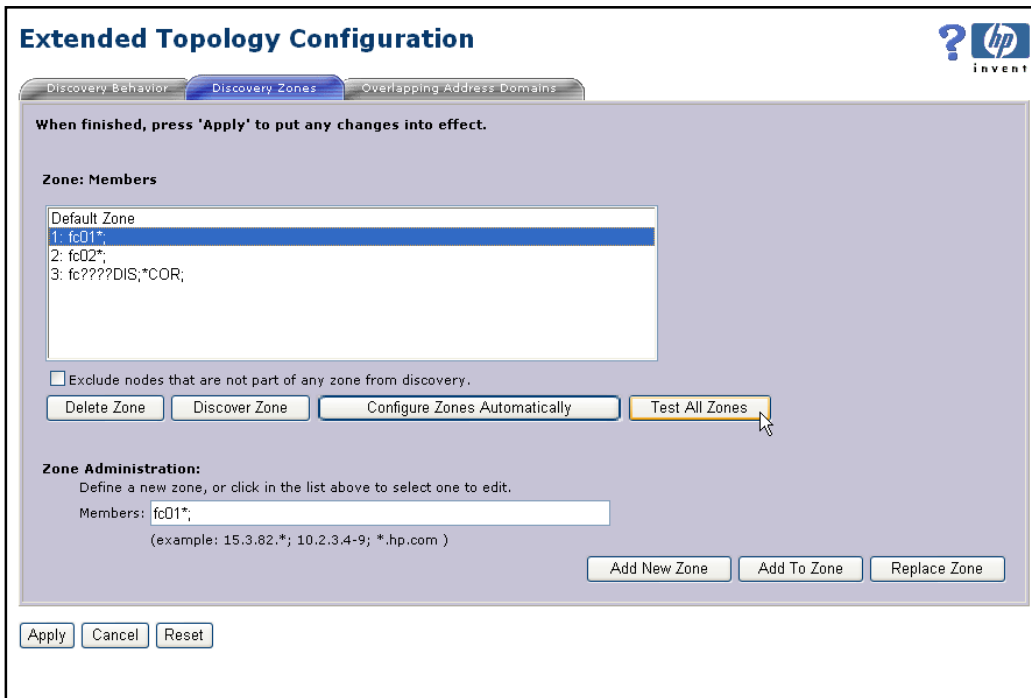


Get a feel for how each time each zone takes. Then you can project this across the entire discovery. There is additional time at the end of a large discovery but this will at least let you have a sense of how fast your zones will go. Ignore this backend time while doing this experiment and just focus on the time per zone.

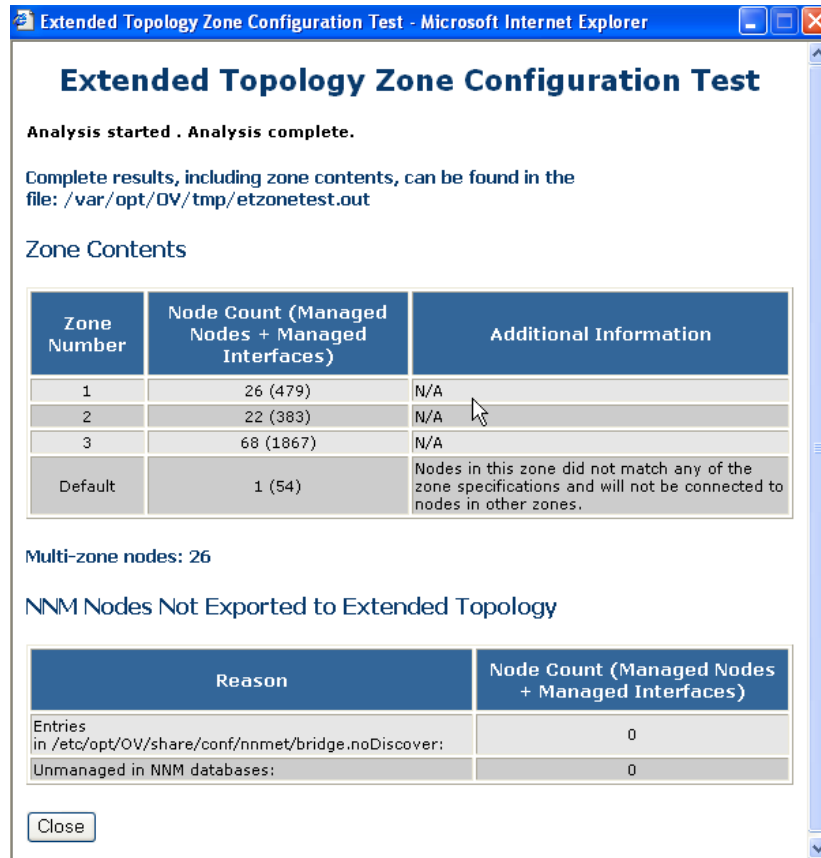
When you are finished, you should replace the backup `etconfig.xml` file. Then restart ovas.

## 9.4 Testing All Zones

After configuring the zones either via manual or autozone, you should run the Test All Zones button.



Look over your results. Your goal is to have the Additional Information say N/A for each row. You also probably don't want anything in the default zone though you might be using this zone as a catch-all. If so, make sure it adheres to the guidelines we listed above.



You should also look at the log file `/var/opt/OV/tmp/etzonetest.out` that is generated. In that file you'll find a nice listing of the nodes in each zone.

We recommend that you re-test your zones occasionally to make sure nodes aren't inadvertently falling into the default zone.

## 9.5 *Running ET Discovery*

Now you are ready to run a discovery. You can either do this through the ET Configuration GUI or by running the command:

```
$OV_BIN/etrestart.ovpl -verbose -disco
```

I like the `-verbose` option though it is not required. You can monitor the progress of the discovery via the Discovery Status page. You can also tail the file `$OV_PRIV_LOG/ovet_disco.log`.

## 9.6 *Incremental Node Discovery*

Incremental Node Discovery is one of the new features released in NNM 7.51. This feature allows the users to have "Incremental node discovery" in ET. As of NNM 7.50, only netmon (classic NNM) was able to discovery the nodes dynamically. In ET, it was

not possible to add the single node dynamically and discover it. ET always required full/zonal ET (batched) discovery to incorporate newly added/managed nodes in NNM. With Incremental Node Discovery coming in NNM 7.51, users will have the benefits of managing the newly added nodes in ET and they will also have the capability to monitor these newly added ET nodes via APA. However, there are few limitations to Incremental Node Discovery. For example, a node added in ET via Incremental Node Discovery is the representation of classic NNM topology node. Since netmon (classic NNM) is not able to discover VLAN, HSRP and etc specific interface information; Incremental Node Discovery will not have the VLAN, HSRP interfaces discovered and monitored as well. Nor will these nodes have any addresses beyond the management address. There is no (L2/L3) connectivity information calculated if the node is added in ET via Incremental Node Discovery. All such nodes are seen as island nodes in Dynamic views. All the interfaces of such nodes are also “Unconnected” interfaces. Incremental Node Discovery can be enabled by adding the “-single\_node\_discovery” option in the \$OV\_CONF/ovet\_bridge.lrf file.

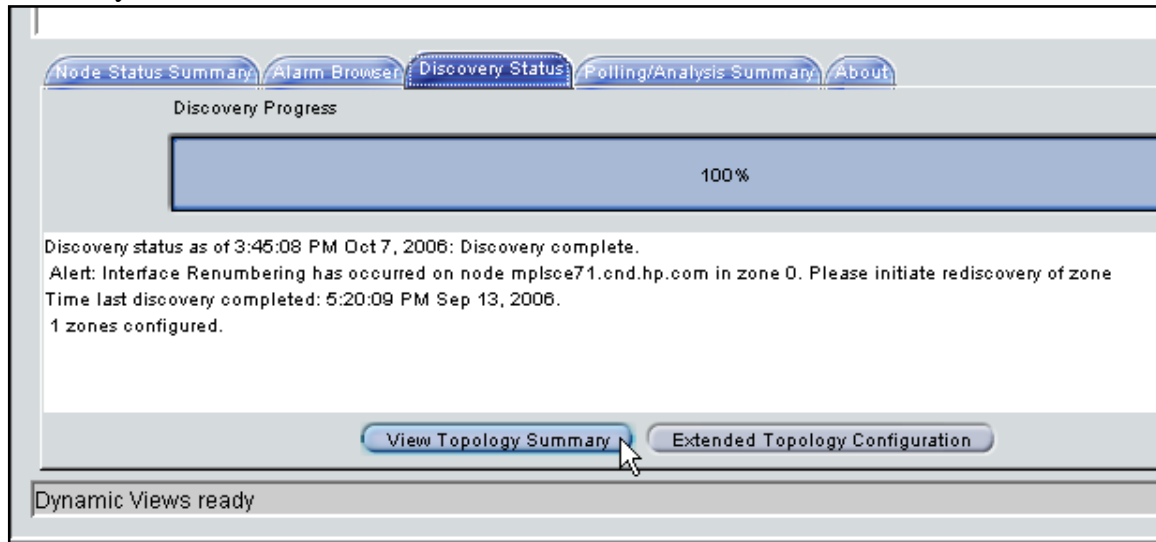
With all these limitations, you may be asking, “Why should I use this feature?” The main reason is to be able to monitor these nodes more quickly via APA. We present a lot of details about APA in other sections in this document. The APA behavior for polling new incremental nodes is defined by the isNewNode and isNewInterface polling policies (paConfig.xml) until a new ET discovery is run at which point these nodes are no longer considered “new”. Many customers like to at least “ping” these new nodes just to make sure that they are alive until the next discovery.

## 9.7 **Cisco Discovery Configuration**

A new feature that was released in 7.51 is Cisco Discovery Configuration. This feature is strictly for reducing your ET discovery time though it could also improve your connectivity if done right. There is a good whitepaper about this functionality, namely \$OV\_DOC/WhitePapers/CiscoDiscoveryConfiguration.pdf. Please read through this whitepaper as we won't duplicate the information here. If you run CDP in your network, we strongly suggest that you try disabling Forwarding Database Tables (FDB) on strategic devices. You need to be careful with this feature but it can be powerful. If you disable FDB, then you rely on CDP for connectivity. Obviously you won't discover connectivity to non-CDP devices from Cisco devices. For example, if you have Cisco switch connected to a Windows Server, you won't discover the connection to the server. (Assumption here is that the server is monitored by NNM.) So you may not want to use this feature on certain switches connected to servers. But you might want to go with “pure CDP” discovery on your Wireless Access Points as servers typically won't be connected to them.

Do some thinking about this and try out some experiments. We've seen substantial improvement in discovery times with this feature. But you must be careful to not accidentally lose important connections. We recommend that between experiments, you should track the number of layer 2 connection to make sure you aren't losing a lot.

To check this, click on the Discovery Status tab. Then click on View Topology Summary.



Then look for the section listing “Number of L2 Links”. Make sure that number doesn’t drop unexpectedly low. But note that sometimes it drops for legitimate reasons due to improved discovery accuracy. You should also launch a Neighbor View on nodes of the type you’ve been modifying and see how the connectivity looks.

### Extended Topology

**Network Node Manager Extended Topology Information:**

- State: **Topology State = READY**
- Last Discovery Completed (Cache Timestamp): **Sep 13, 2006 5:20:09 PM**
- Length of last discovery cycle: **13 Minutes, 4 Seconds**
- Number of Licensed Node Limit : **Unlimited**
- Number of Nodes: **70**
  - IPV4 Nodes: **70 (100%)**
  - IPV6 Nodes: **0**
  - [Doesn't respond to SNMP](#): **4 (6%)**
- Number of Interfaces: **1789**
  - IPV4 Interfaces: **1789 (100%)**
  - IPV6 Interfaces: **0**
- **Number of L2 Links: 124**
- Number of VLANs: **26**
- Number of HSRP Routing Groups: **3**
- Number of VRRP Routing Groups: **0**
- Number of Meshes: **15**
- Number of IPV6 PrefixGroups: **0**
- Number of IPV4 Subnets: **111**
- Number of Aggregate Ports: **12**
- Number of Boards: **51**
- Average Number of Interfaces/Node: **25.56**
- Number of Addresses: **371**
  - IPV4 Addresses: **371 (100%)**
  - IPV6 Addresses: **0**
- Total Number of Topology Objects: **2385**

## 9.8 Interface Filtering

Another feature that was introduced as part of 7.51 is Interface Filtering. Again there is a whitepaper describing this feature in \$OV\_DOC/WhitePapers/InterfaceFiltering.pdf.

This feature was primarily introduced to deal with routers that have a huge number of virtual interfaces that you don't care to manage. Some customers have routers with over 10,000 virtual interfaces. Devices this large can be very challenging for ET. With this feature, you can eliminate many interfaces that you don't care about.

We have also found this feature to be useful to speed up discovery by eliminating blocks of interfaces you never intend to monitor. For example, you might want to only monitor the uplink on your Wireless Access Points. With this feature, you could identify your WAPs by a naming convention and then eliminate all interfaces except those with an IF Name of "Gi\*" or a specific IF type. You can find a lot of information about interfaces by opening the Node Details page of a node. From here you can easily see the Name, Description and Alias of an interface. For more details, click on the Entity Name link for an interface.

**Node Details for ntc6k04**

General Capabilities Addresses Interfaces Boards VLANs Layer 2

Total Number of Interfaces: 22

Interfaces										
Entity Name	Name	Description	Alias	Status	Index	Board	Port	DNS Name	Address	
<a href="#">ntc6k04.cnd.hp.comf.0[1.1]</a>	Gi1/1	GigabitEthernet1/1	LACP Port-Channel 1 to ntc6kgw1 Gig2/3	Normal	1	1	1			
<a href="#">ntc6k04.cnd.hp.comf.0[2.1]</a>	Gi1/2	GigabitEthernet1/2	LACP Port-Channel 1 to ntc6kgw2 gig3/3	Normal	2	1	2			
<a href="#">ntc6k04.cnd.hp.comf.0[3.1]</a>	Gi2/1	GigabitEthernet2/1	LACP Port-Channel 1 to ntc6kgw2 Gig2/3	Normal	3	2	1			
<a href="#">ntc6k04.cnd.hp.comf.0[4.1]</a>	Gi2/2	GigabitEthernet2/2	LACP Port-Channel 1 to ntc6kgw1 Gig3/3	Normal	4	2	2			
<a href="#">ntc6k04.cnd.hp.comf.0[5.1]</a>	Gi2/3	GigabitEthernet2/3	-	Not Monitored	5	2	3			
<a href="#">ntc6k04.cnd.hp.comf.0[6.1]</a>	Gi2/4	GigabitEthernet2/4	-	Not Monitored	6	2	4			
<a href="#">ntc6k04.cnd.hp.comf.0[7.1]</a>	Gi2/5	GigabitEthernet2/5	-	Not Monitored	7	2	5			
<a href="#">ntc6k04.cnd.hp.comf.0[8.1]</a>	Gi2/6	GigabitEthernet2/6	-	Not Monitored	8	2	6			

From the Interface Details page, you can see the value for IF type. You can find the IF type definitions at <http://www.iana.org/assignments/ianaiftype-mib> or you can also see this MIB on your installation under \$OV\_SNMP\_MIBS/Standard/IANAifType-MIB.




### Interface Details for ntc6k04.cnd.hp.com[ 0 [ 12 ] ]

General Capabilities Addresses HSRP

**General**

ntc6k04.cnd.hp.com[ 0 [ 12 ] ]

**Status:**  Normal

**Description:** EOBC0/0


**Extended Topology ID:** 79deacf0-3d13-71db-0bae-0f0278ed0000

**Interface Last Updated:** Sep 13, 2006 5:17:59 PM

**Interface Created:** Sep 13, 2006 5:17:59 PM

**Interface Name:** EO0/0

**Interface Alias:** -

**Interface Type:** 53 

**Interface Index:** 12

**Port Number:** 0

**Vendor Port Number:** 0

**Interface Speed:** 100 MBits/sec

**Interface Physical Address:** 00:00:21:00:00:00

**Interface Administration Status:** up

**Interface Operational Status:** Up

At one customer site, we were able to eliminate all the ports on User Access switches (not server switches) except for uplinks. This eliminated 100,000 interfaces and significantly improved the discovery speed. If you can get the interface count down, you'll be able to potentially combine zones together, possibly improving your connectivity accuracy. You'll also see the entire system speed up by simply removing the volume of interfaces that NNM has to deal with. Another nice benefit of this is that your APA poller will have faster start up times.

NOTE: We have noticed one side-effect that you need to be aware of. When you eliminate huge number of interfaces via this feature, they will not be discovered as part of ET but they may have already been discovered as part of classic NNM (netmon). After you setup the netmon.interfaceNoDiscover file, netmon will begin removing interfaces from the classic topology when a "config-poll" is performed on a node. There is usually a high overhead when removing huge numbers of interfaces (like 100,000). Expect to see some processes running high CPU like ovwdb and netmon for the next 24 hours after making this change. When running this high, it makes ET discovery slow down. So you'll need to strike a balance between testing your improvements in ET and waiting for netmon to demand-poll all the nodes. It is recommended to shut off netmon during the ET interface filtering experimenting. After you get all the interfaces that you wanted to discover, re-enable netmon and let it run 24 hours.

## 10 Tuning APA

Now that you've finished your discovery, your next step is probably to enable the APA poller. Ironically, we are going to now suggest that you might want to tune APA with a smaller topology. It'll make for faster turnarounds when you are tuning. You might consider going back to just one or two zones as mentioned previously for making the APA work while tuning. But eventually you'll need to tune APA against the entire topology.

We won't be going into the technology behind APA in this paper but we'll point out a few things here. APA only works on ET objects. It performs state-of-the-art polling and root cause analysis. Much of its analysis is based on "neighbor analysis". That's why it's important to get a good ET discovery. APA is extremely flexible and can be tuned to do almost anything. But this flexibility introduces complexity.

You can go with the out-of-the-box settings but in general, you'll get better results by doing some tuning. It's not trivial and you'll need to get comfortable editing XML. But if you are willing to get your hands dirty, your network management will be that much better.

### 10.1 *APA and Netmon*

APA replaced the netmon status polling. Netmon continues to run even when APA is enabled. Netmon continues to do regular "configuration polling" and auto-discovery of new nodes if you have it configured this way. So you will see both processes running simultaneously. But after you enable APA, it does all the status polling.

Most important to note is that none of the netmon status polling configuration carries over to APA. If you had previously marked many interfaces as unmanaged for netmon (via either the OVW GUI or using the ovautoifmgr tool, none of these changes will apply to APA.

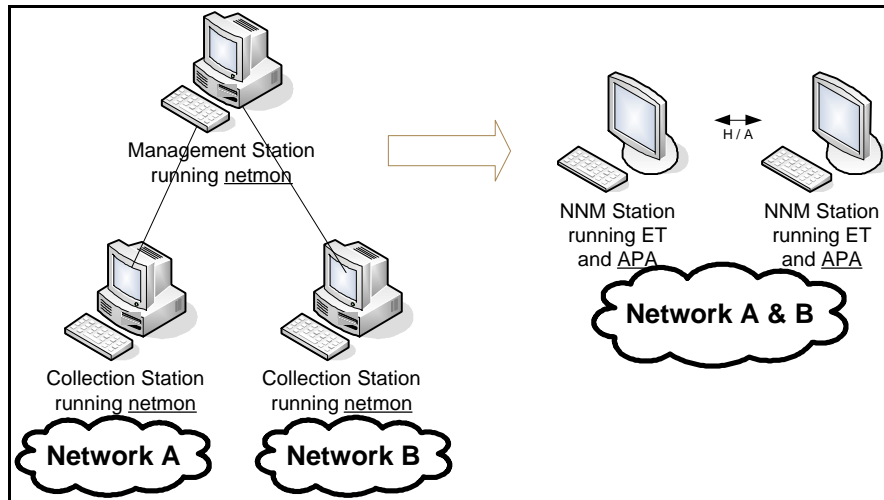
You must configure and tune APA using a separate, new approach as described in the next few sections.

### 10.2 *APA Architecture*

APA is not designed to work with traditional NNM distribution (Distributed Internet Monitoring or DIM) involving a Management Station (MS) and multiple Collection Stations (CS). That is because the ET database is not replicated.

## 10.2.1 Collapsing Stations

Because APA can scale to larger numbers than netmon was able to, some customers have had success “collapsing” multiple collection stations into a single NNM station. This is shown here:

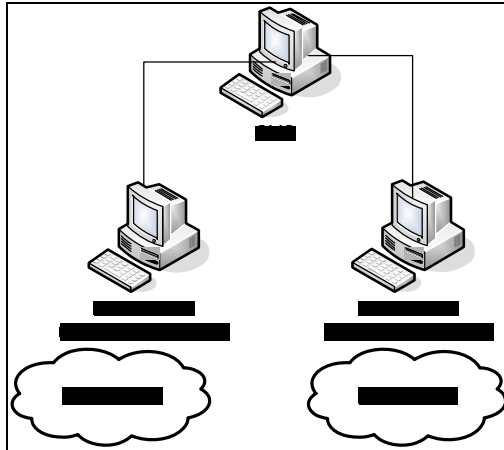


This picture shows a High Availability (H/A) clustering solution being used for failover like HP Service Guard. If you don't require failover, you could just use a single station and do regular backups. If a station then fails, you could restore your latest backup and continue monitoring from there.

If you choose this approach, be sure to run some initial tests. At one customer site, we found that the numbers from the old netmon stations were a little misleading. Because the customer had limited their discovery with netmon options, not all of the interfaces were being discovered and monitored. When we ran “ovtopodump -l”, it showed a low interface count on the collection stations. But when we discovered the same network with NNM 7.5, it had a much higher interface count because all the L2 interfaces were now being discovered. This made us realize we couldn't collapse as many collection stations together as we had initially hoped.

## 10.2.2 Using OVO as the Manager of Managers (MOM)

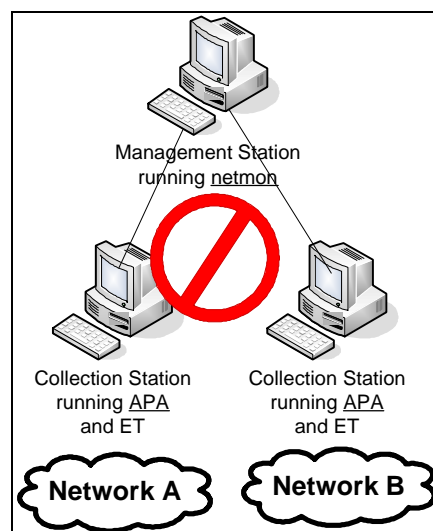
If you decide to go with separate ET stations for monitoring portions of your network, you can collect all the alarms at a top level OVO stations. You'll need to launch to specific HomeBase instances for each ET station for dynamic views like Neighbor View.



## 10.2.3 Using NNM MS with netmon and NNM CS with APA

Some customers have tried using the failover capabilities of DIM by using netmon at the MS level and APA at the CS level. We recommend against this solution because APA and netmon model status differently and are configured differently. You will end up with inconsistencies. You also will end up with more alarms than expected. It's almost impossible to make netmon and APA poll in the exact same way.

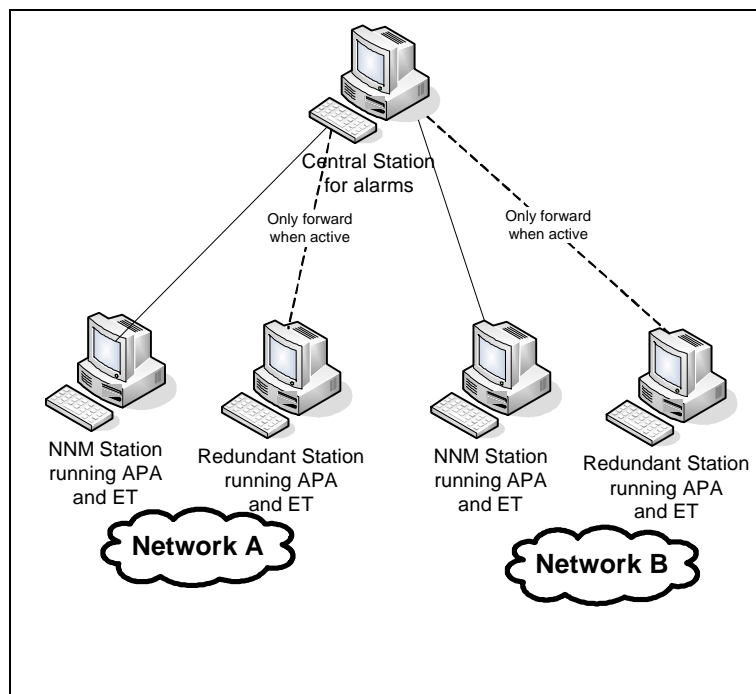
You could use this setup if you didn't need failover and just want a central management station similar to the OVO solution described above.



## 10.2.4 Using two NNM stations in a “warm standby” scenario

Some customers have had success using two NNM stations acting as a pair to simultaneously monitor the same network in order to have failover. Usually the alarms are forwarded to a central station. In this setup, when one machine fails, the redundant machine is configured by the user to forward events to the central station only during the failover period.

In this setup, each NNM station is running its own independent ET discovery and APA. There is no formal synchronization between the stations. Only the configuration files are kept in synch. This can be done via a shared drive or by deploying all configuration files to the NNM stations from a central location. There aren't many configuration files (4 or 5 files) to keep in synch so this can be a fairly easy task. There is some risk that the ET databases may be different between the primary NNM station and the redundant station but this risk is quite low if ET discovery is done near the same time and that any ACL's allow the same node access. Of course, traps must be sent to both stations in the pairing.



## 10.3 Enabling APA

With your architecture in place, the next step is to enable APA. Sometimes we call this, “throwing the big switch.” By default, APA monitors only HSRP and OAD (Overlapping Address Domain) devices. To make APA poll all nodes, you need to enable APA (Throw a Big Switch). When you enable APA, netmon will no longer be doing status polling. That is now the job of APA. The netmon poller will continue to do new-node discovery if have run auto-discovery and it will do configuration polls on nodes. APA also does an abbreviated form of a configuration poll but it does not full replace the netmon version.

To throw the big switch, run

```
$OV_BIN/ovet_apaConfig.ovpl -enable APAPolling
```

This command automatically configures netmon to no longer perform status polls. Be prepared for a high number of alarms. APA polls things differently than netmon and you'll see new alarms. APA will send alarms on nodes that netmon has already alarmed you about. After some time, it will get to a steady-state. We'll get these alarms under control via tuning but you will see a high number at the beginning. You could put in place some the suggested tuning steps before throwing the big switch but we usually just enable APA and then start working from there.

You'll also see alarms about addresses for the first time. APA models interfaces and addresses separately whereas netmon treated addresses as an attribute of an interface. The APA model is more accurate. You can have an interface reporting as "up" via SNMP status and yet not be reachable via ICMP (ping).

The basic APA Model is a parent->child relationship with top most entity as a Node.

#### **Model is:**

Node (Node contains boards/cards)

- Board (Board contains interfaces) [if Boards/cards are not present on the device, interfaces are children of nodes]
  - Interface (Interface contains address(es))
    - § Address (Lowest level entity)

Status propagation for different objects happens like:

Node  $\beta$  Board  $\beta$  Interface  $\beta$  Address

#### **Different Statuses marked by APA are:**

Unknown (Blue), Minor (Yellow), Critical (Red), Normal (Green) and Disabled (Brown - only for interfaces which are having ifAdminStatus as "Down" during the ET discovery).

#### **Flags set by APA are:**

Not Monitored (for interfaces/nodes)

Responding, Not Responding, Unreachable (for addresses)

#### **Default behaviour of APA:**

- On detecting a failure, root cause (primary failure) of a failure is marked as Critical (Red). Parents, if any, are marked as Minor (Yellow). Children, if any, are marked as Unknown (Blue) to indicate secondary failure.

- For example, if an address is not responding using ICMP/Ping, address is marked as “Not Responding”; the interface (Parent) which is associated with address is marked as Minor (Yellow); Board, if any, (Parent of an interface) will be marked as Minor (Yellow); Node (Parent of a board if the board exists Or parent of an interface if no boards exists) will also be marked Minor (Yellow).
- For example, if an interface goes down (ifOperStatus=down, in case of linkDown, marked as Critical/Red); addresses (Children) are marked as Unknown (Blue); Board (if any) and Node are marked as Minor (Yellow) being the parents of interfaces.
- Disabled and NotMonitored statuses are not propagated to top/parent level at all.
- In case of a Node Down (Node stops responding to SNMP and ICMP/ Node is shut down/ Power cable of a node is pulled out), it is marked as Critical (Red) and all the underlying boards, interfaces and addresses are marked as Unknown (Blue) to indicate the secondary failure.
- In case, where a main IP address of a node does not respond to SNMP (but responds to ICMP/ping), then the Node is marked Critical with an event in the Alarm browser saying “SNMP Agent is not responding”. In this case, all the underlying interfaces of the node will be marked “Unknown”.
- In case, where the node partially responds to SNMP for few interfaces (including main IP address interface), but it does not respond to SNMP (due to an SNMP timeout) for other interfaces then, the responding interfaces are “Normal”, the not responding interfaces (the timed out ones) are marked “Unknown” and the Node turns “Minor” !! This is because there is something genuinely wrong with the node as few interfaces timed out on SNMP query.
- In case of a primary/secondary failure scenario, where if router A goes down and we are not able to reach any nodes behind router A, all these unreachable nodes are marked as Unknown (Blue) to indicate the secondary failure. Root cause of this problem will have an event associated with it and the entity (Node/Interface) will be marked Critical/Red with parents being marked as Yellow (Minor).

## 10.4 *Measuring APA performance*

You can see how well APA is performing by going to HomeBase and clicking on the Polling/Analysis Summary page. You typically want to watch the Waiting Poller Tasks and Waiting Analyzer Tasks. You want these to be near zero though they can be as high as 100 without a real problem. They will be artificially high when you first start ovet\_poll so let it have a little time to get caught up. If the numbers are staying high, then APA is falling behind in polling and analysis. You can do two things to help improve this. The first, and most important, is to change polling policies to reducing the amount of polling and analysis required. This is described in the upcoming sections. The second thing you can do is to increase the number of threads for polling and analysis. Again that is described ahead.

### Network Node Manager Home Base

You are currently running with a temporary license that expires on Dec 12, 2006 9:11 AM. The system administrator run \$OV\_BIN/ovnnmPassword on the server system, ksm

View

Container View

Description

A graphical representation of a container of devices

Node Status Summary Alarm Browser Discovery Status **Polling/Analysis Summary** About

Most recent statistics update from ovet\_poll received at Oct 17, 2006 4:07:43 PM.

Statistic	Current	Max
Active Analyzer Tasks	0	0
Waiting Poller Tasks	1	1
Interfaces Polled (SNMP)	96003	96003
Addresses Polled (ICMP)	2715	2715
Boards Polled (SNMP)	1444	1444
HSRP Groups Polled	256	256
Waiting Analyzer Tasks	0	0

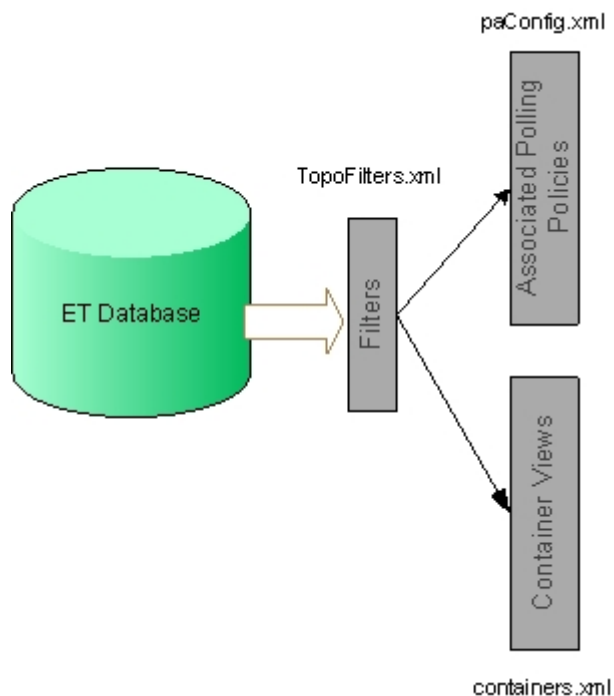
Dynamic Views ready

## 10.5 *Concepts of Tuning*

APA behavior can be changed. APA works with collections of objects. These objects can be nodes, interfaces, addresses, HSRP groups, and others. Collections of objects are defined via filters and some inherent attributes assigned by ET discovery. For instance, you might have a collection of Cisco Switches that is defined by their System Object ID value. You could also have a collection of all interfaces that are on a router and have an associated address within a specific range. Collections can be very simple or very complex.

These collections of objects then have polling policies associated with them. If an object passes multiple filters, an ordering is applied such that only one polling policy is assigned to the object.





To change APA, you can change existing filters and you can also define new filters. You can also define the associated polling policies. The three main parameters on polling policies include interval, snmpEnable and pingEnable.

The principle files that you will work with include

- \$OV\_CONF/nnmet/paConfig.xml
- \$OV\_CONF/nnmet/topology/filter/TopoFilters.xml
- \$OV\_CONF/nnmet/topology/filter/APANoPollNodes.xml
- \$OV\_CONF/nnmet/topology/filter/MyHostID.xml
- /opt/OV/bin/ovet\_topodump.ovpl
- /opt/OV/support/NM/checkpollcfg
- /opt/OV/bin/ovet\_demandpoll.ovpl

The main files are paConfig.xml and TopoFilters.xml. TopoFilters.xml is where the filters are defined. paConfig.xml is where the associated polling policies are defined. The other two xml files are support files that are referenced at times. ovet\_topodump.ovpl, ovet\_demandpoll.ovpl and checkpollcfg are tools that will be used as part of the tuning process.

There are two XSD schema definition files that you might wish to read through but these should not be modified.

- \$OV\_CONF/nnmet/topology/filter/TopoFilter.xsd
- \$OV\_CONF/nnmet/topology/filter/TopoFilterCommonTypeDef.xsd

To tune APA effectively, you need to understand both filters and polling policies. I have a separate section devoted to filters in ET. You may wish to read the section before proceeding here.

The usual tuning steps include:

- Identify problem or area of concern
- Find matching filter or create a new one in TopoFilters.xml
- Adjust polling behavior in paConfig.xml
- Validate XML edits
- Confirm polling settings with various tools including checkpollcfg, ovet\_demandpoll.ovpl, and the APA process itself (ovet\_poll).
- Restart APA (ovet\_poll) though this may need to be done before the previous step depending on which tool you are using. The checkpollcfg tool does not require a restart of APA.

## 10.6 *The simple way out*

If you aren't interested in learning the details of tuning APA, let me suggest a few common solutions that generally work well for most customers. You could decide to just go with these.

1. If you have Switch/Routers (we like to call them swouters) like Cisco 6500 and you run these primarily as a switch, then follow the instructions the Appendix of this document. NOTE: This is a new and improved version of the swouter document. This has changed from the previous document so look over the changes. The changes are not mandatory but provide a little more flexibility.
2. Use the predefined class specifications that are commented out in paConfig.xml. Simply edit paConfig.xml and follow the instructions that are in the file near each of these class specifications.
  - a. unmanage interfaces of a type that you know you don't want to manage (class specification is called IfTypeFilter)
  - b. Disable ping for ranges of addresses that you know are unreachable (class specification is called NoPingAddresses)
3. Make the changes to improve APA startup time detailed in Section titled Improving APA startup time.

You could skip over the rest of this APA section if that's all you want to do.

## 10.7 *Understanding APA polling policy matching*

Figuring out why APA is polling something in the way it is can be a little tricky. Polling policies are labeled as Class Specifications in the paConfig.xml file.

The three primary "parameters" that are user configurable are snmpEnable, pingEnable, and interval. The values for snmpEnable and pingEnable are either true or false. The values for interval are any positive integer greater than zero.

For each object (like an address, interface or node), APA begins parsing the paConfig.xml from the top down looking for a polling policy match for the object and the parameter setting. It also looks for match for its parent objects and applies logic to make a decision about how to poll the object. For snmpEnable and pingEnable parameters, this is a logical AND. For interval, its more of an if/else statement as detail below.

If it can't find an explicit policy match for the object, it reverts to the default setting. Note that default settings are "typeless", meaning there isn't a separate default for nodes, interfaces and addresses. They all share the same default settings. It then does the same algorithm for the parent object and performs a logical AND of all the results.

In APA, Node is the parent of Interface which is the parent of Address.

Node  $\beta$  Interface  $\beta$  Address

So if you are trying to find a polling policy for an address in paConfig.xml, it will look for explicit polling policies for the address, the parent interface and the parent node. It would look something like this:

(Address Setting) AND (Interface Setting) AND (Node Setting)

where each setting may either be explicit or default.

Let me walk you through an example. Supposed we have an address 10.1.1.1 that is assigned to an unconnected interface on a router and we are trying to see whether it is pinged or not. APA begins looking through the paConfig.xml file for a policy match for the address. Let's suppose it doesn't find one. So it reverts to the default setting which is PingEnable=true. Now it goes to the parent object which is an interface. It now parses the paConfig.xml file again from the top down looking for a match on the interface. It now finds a match for "unconnected interfaces on routers" and the configuration is set to PingEnable=false. Now it goes to the parent object which is a node. Again it parses from the top down looking for a match. It finds one for routers and the configuration is set to PingEnable=true. Now we AND all these values together and you'll get:

True AND False AND True = False

This isn't as hard as it may appear at first but it comes in handy to understand it.

The other thing we need to cover just slightly is when the parameter is an "interval". It's similar logic but rather than ANDing the values together, it just takes the first value it matches and doesn't try the default until all parents are exhausted. So the logic is:

(Explicit Addr Interval) else (Explicit Intf Interval) else (Explicit Node Interval) else (default interval)

Let's do another simple example here. If you had set Routers to have a polling interval of 2 minutes and hadn't set any polling interval for any Interfaces and hadn't set any interval for addresses. Let's also suppose the default polling cycle is set to 5 minutes. Then it would evaluate to 2 minutes as shown here:

(undefined) else (undefined) else (120 seconds) else (300 seconds) = 120 seconds

## 10.8 **How does APA “poll”**

APA can gather status either perform an SNMP poll that reads status from a MIB or by issuing an ICMP poll (usually called “ping”) and basing status on the response of the address. We will explore tuning both of these types of polls. If pingEnable and snmpEnable both are false for a particular object, that object is marked as “Not Monitored”.

There are also limitations based on the object type. While we speak of pinging a node or setting pingEnable=true on a node, that’s really the parent object. The ping is actually performed against an address that is on a interface that is on a node. Similar logic is applied to SNMP polling.

## 10.9 **Out of the box defaults**

Before you make changes to APA, it’s good to understand the out-of-the-box settings. The best way to do this is to study the paConfig.xml file. The following list is simplified and not comprehensive but gives a good starting point.

Device Type	SNMP Polling	Ping Polling
Default	•	•
Router	•	•
Switch	•	
End Node		•
Connected Admin Up Switch IF	•	
Connected Admin Up Router IF	•	•
Unconnected Admin Up Router IF	•	•
Unconnected Admin Up Switch IF		
Unconnected End Node		•

## 10.10 **What if a node is both a switch and a router?**

If a node is both a switch and a router, then APA polls it as a router. That is due to the fact that isRouter is above isSwitch in the paConfig.xml file.

Based on the previous table, you can see that would mean that a switch/router is both SNMP polled and Pinged out-of-the-box. It also means that Connected and Unconnected Admin UP interfaces would be both SNMP polled and pinged. If you are using these devices more like a switch, you might not like getting an alarm every time a computer is shutoff at the end of the day on a user access port.

You could change the polling policies on routers to only poll connected admin up interfaces on routers (and also change the node level) but that might not be ideal as you might think of switch/routers differently that pure routers.

We suggest making a new category called “swouters” and defining polling policies on nodes, interfaces and addresses for swouters. See Appendix A – A new and improved Swouter for complete details on this solution.

### 10.11 **Some HP supplied policies**

HP included some commented out Class Specifications in paConfig.xml. These include

- APANoPollNodes
- WanIF
- IfTypeFilter
- AllCards
- NoPingAddresses

This list may not be complete. See the paConfig.xml file for the complete list. We found that these were common policies that customers might want to use yet they usually require some modification by individual customers.

To use these policies, simply edit paConfig.xml and follow the instructions that are in the file near each of these class specifications. Full instructions are included as part of the comments in the file. We’ll use the IfTypeFilter in the next example.

### 10.12 **A quick lesson on reading the Class Specification (polling policy) in paConfig.xml**

Let us show you a snippet of XML from the paConfig.xml file that defines a polling policy. You can see that there are three parameters namely interval, snmpEnable and pingEnable. These are set to 300, true and true respectively. You should only modify these values. But on some polling policies, all three of these may not be present. Remember in a previous section we described how a setting may be explicit or may revert to the default. Each of these would be an explicit setting. If you setup or modify a policy and it doesn’t have an explicit setting, feel free to add one. Just copy it from this snippet.

```
<classSpecification>
  <filterName>isRouter</filterName>

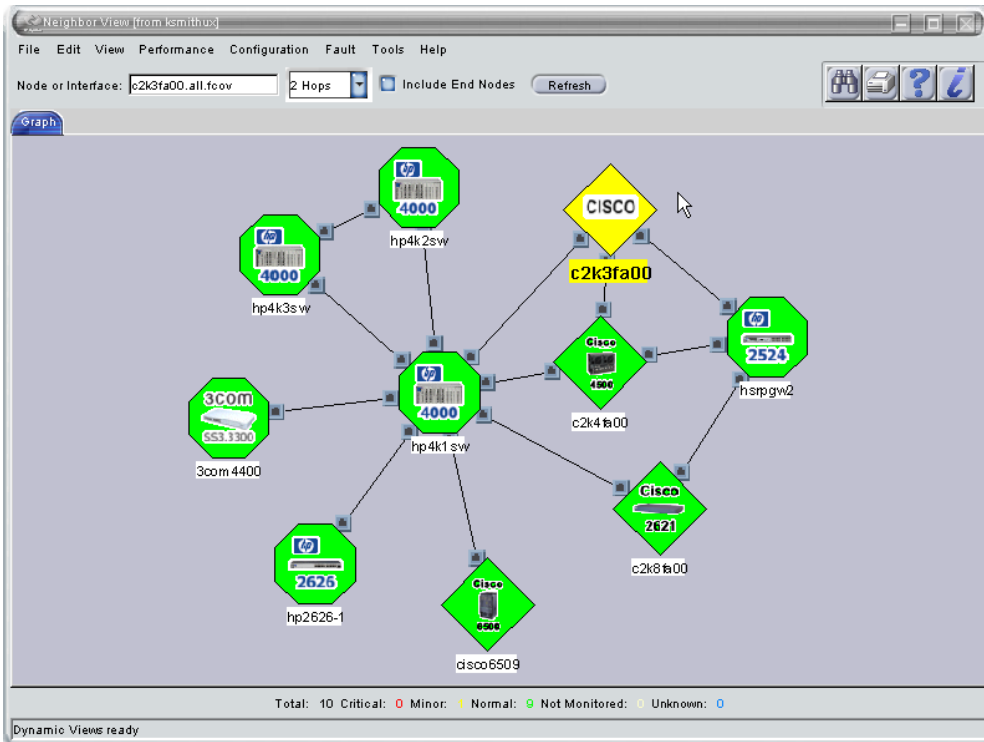
  <parameterList>
    <parameter>
      <name>interval</name>
      <title>Interval to Poll Device</title>
      <description>The interval which the device will be polled in
                    seconds.</description>
      <varValue>
        <varType>Integer</varType>
        <value>300</value>
      </varValue>
    </parameter>

    <parameter>
      <name>snmpEnable</name>
      <title>Enable polling via SNMP</title>
      <description>Enable/Disable polling of a device via SNMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>true</value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>
```

```
<parameter>
  <name>pingEnable</name>
  <title>Enable polling via ICMP</title>
  <description>Enable/Disable polling of a device via ICMP.</description>
  <varValue>
    <varType>Bool</varType>
    <value>true</value>
  </varValue>
</parameter>
</parameterList>
</classSpecification>
```

## 10.13 Example using User Configurable Filters

Let us guide you through a simple example. The values in this example are not practical but it's a good sample. Let's suppose that you have a number of nodes that are in what you perceive as a "normal" state yet dynamic views are showing them as non-normal (yellow). You would like APA to consider the node as normal. The way to achieve this is to identify what is making it non-normal and modify the polling policies to change this state.



In this example, you can see the yellow node we are interested in c2k3fa00. Launch the Node Details page by double clicking on the node. You can see that one of the interfaces has a status of Minor.

**Node Details for c2k3fa00**

General Capabilities Addresses **Interfaces** VLANs Layer 2

- Name: pimMib - Value: Str:
- Name: mcMib - Value: Str:IPMROUTE-MIB

**Interfaces**

Total Number of Interfaces: 7

Entity Name	Name	Description	Alias	Status	Index	Board	Port	DNS Name	Address
<a href="#">c2k3fa00.all.fcovf_0 [ 1 1 ]</a>	Et0/0	Ethernet0/0	Connect to hp4k1sw port H1	Normal	1	-	0		IPv4: 15.2.131.63
<a href="#">c2k3fa00.all.fcovf_0 [ 2 1 ]</a>	Se0/0	Serial0/0	Connect to cisco2k5 Serial0/0	Minor	2	-	0		IPv4: 172.18.0.2
<a href="#">c2k3fa00.all.fcovf_0 [ 3 1 ]</a>	Etl1/0	Ethernet1/0	Connect to hrsr900t port 1	Normal	3	-	0		IPv4: 15.2.132.3
<a href="#">c2k3fa00.all.fcovf_0 [ 4 1 ]</a>	Etl1/1	Ethernet1/1	Connect HSRP to cisco4k3 Ethernet2	Normal	4	-	0		IPv4: 15.2.129.2
<a href="#">c2k3fa00.all.fcovf_0 [ 5 1 ]</a>	Etl1/2	Ethernet1/2	-	Not Monitored	5	-	0		
<a href="#">c2k3fa00.all.fcovf_0 [ 6 1 ]</a>	Etl1/3	Ethernet1/3	-	Not Monitored	6	-	0		
<a href="#">c2k3fa00.all.fcovf_0 [ 7 1 ]</a>	Nu0	Null0	-	Normal	7	-	0		

You can drill down further down further by clicking on the interface hyperlink.

**Interface Name:** Se0/0

**Interface Alias:** Connect to cisco2k5 Serial0/0

**Interface Type:** 32

**Interface Index:** 2

**Port Number:** 0

**Vendor Port Number:** 0

**Interface Speed:** 2 MBits/sec

**Interface Administration Status:** up

**Interface Operational Status:** Up

**Capabilities**

- isCDP
- isFrameRelay

**Addresses**

**IP Level:** IPv4

**IPv4 Address:** 172.18.0.2

**IPv4 Ping State:** not responding

IPv4 Address Information			
Address	IP Version	Type	Ping State
172.18.0.2	IPv4	-	not responding

You can see that this address is not responding to ping. You have a few choices now. You could use the NoPingAddresses class specification in paConfig.xml (this also requires modifying the TopoFilters.xml file as per the instructions in the file). But let's say instead that you notice that this interface is of type 32. You decide that you never want to ping any address on an interface of type 32 in your environment. You decide to



use the HP supplied IfTypeFilter section in paConfig.xml to disable ping of any address on an interface of type 32.

You should now edit the paConfig.xml file. Find the section on IfTypeFilter. Read the instructions in the top part of the comment. It tells you to uncomment this section. It also states that this class specification disables ICMP. Notice that there isn't an explicit setting for snmpEnable. You could add one if you want or just let it use the default as this would do.

```
<!-- Uncomment this class specification to disable ICMP (ping)
      for IfTypeFilter. Users should edit the TopoFilters.xml
      file to specify what IFTypes to include in the filter.
<classSpecification>
  <filterName> IfTypeFilter </filterName>
  <parameterList>
    <parameter>
      <name> pingEnable </name>
      <title> Enable polling via ICMP</title>
      <description>
        Enable/Disable polling of a device via ICMP.
      </description>
      <varValue>
        <varType> Bool </varType>
        <value> false </value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>
remove this line also to uncomment above class specification -->
```

Remove the bottom line of text and add --> at the top to close the comment on top.

The new text should look as shown here.

```
!-- Uncomment this class specification to disable ICMP (ping)
      for IfTypeFilter. Users should edit the TopoFilters.xml
      file to specify what IFTypes to include in the filter. -->
<classSpecification>
  <filterName>IfTypeFilter</filterName>

  <parameterList>
    <parameter>
      <name>pingEnable</name>
      <title>Enable polling via ICMP</title>
      <description>Enable/Disable polling of a device via ICMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>>false</value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>
```

Now the next step, as per the instructions, is to edit TopoFilters.xml and specify the IfType that you want. Edit the file \$OV\_CONF/nnet/topology/filer/TopoFilters.xml and search for IfTypeFilter.

```
<!-- Users can modify this filter to include IFTypes to filter on.
      The filter is setup to match interfaces with particular IFTypes.
      Types can be added to the filter by adding a new attribute with
      a particular IFType. -->
<interfaceAssertion name="IfTypeFilter" title="IfTypeFilter" description="Interface are
      of a particular ifType">
  <operator oper="OR">
    <attribute>
```

```

        <ifType>28</ifType>
    </attribute>

    <attribute>
        <ifType>20</ifType>
    </attribute>
</operator>
</interfaceAssertion>

```

Notice that the comment gives further instruction. It happens to be currently set to IF types of 28 and 20. These are just placeholders. They weren't hurting anything previous to enabling the policy in paConfig.xml. Now that the policy is live, we must set the IF type values properly in TopoFilters.xml. You can see that it is performing an OR of the ifType 28 and 20. For our example, we want to change this to ifType 32. Since there is an OR condition, it requires two values but we only want one (32). So, we two choices. We could either change the operator from OR to NOOP (meaning No Operation) and eliminate one of the attributes or you could also just put 32 in for both ifType values. It won't hurt any and makes it easier to add more interfaces. We'll choose the former as that's a little cleaner.

Here's the new XML:

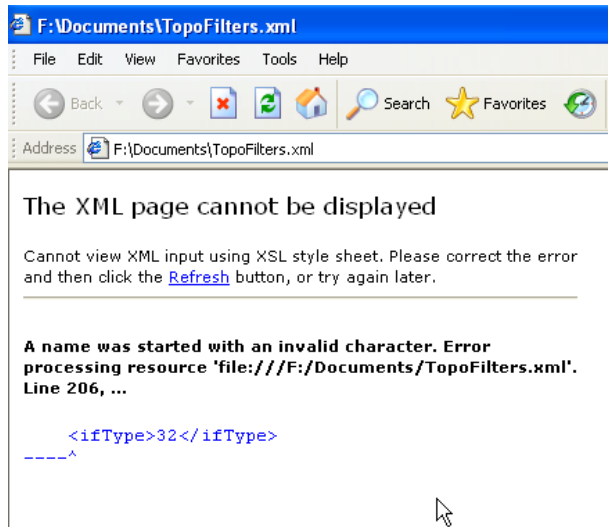
```

<!-- Users can modify this filter to include IFTypes to filter on.
     The filter is setup to match interfaces with particular IFTypes.
     Types can be added to the filter by adding a new attribute with
     a particular IFtype. -->
<interfaceAssertion name="IfTypeFilter" title="IfTypeFilter" description="Interface are
of a particular ifType">
    <operator oper="NOOP">
        <attribute>
            <ifType>32</ifType>
        </attribute>
    </operator>
</interfaceAssertion>

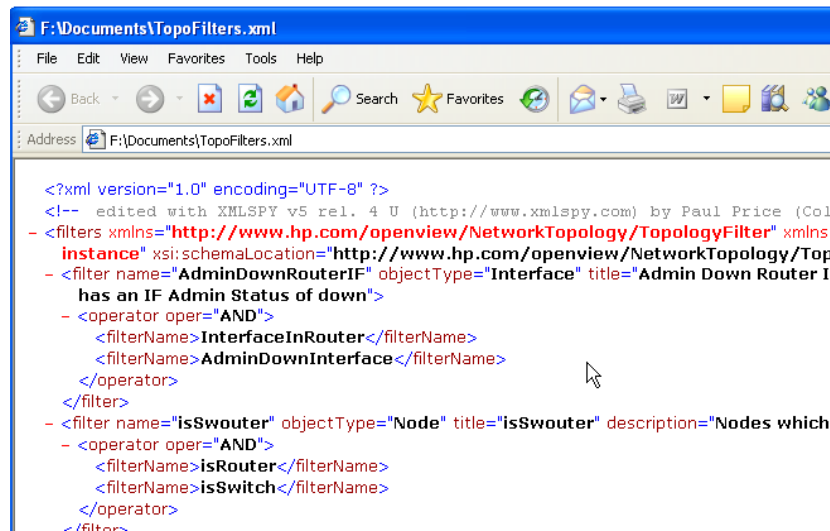
```

You now should test your changes. First, we recommend that you validate the XML checking for syntax errors. One easy way to do this is with Internet Explorer. All you need to do is load the files (TopoFilters.xml and paConfig.xml) into Internet Explorer and it will notify you of any syntax errors. In case you haven't used this, you can open local files by using File->Open and then Browse to a local copy of the files. You could also use an XML editor to validate the syntax.

If there is an error, you'll usually see an error message like:



If there are no errors, you'll simply see the file without any error messages.



The next step is to test the change. Since we made a change in TopoFilters.xml, we'll test that first. We'll use the tool \$OV\_BIN/ovet\_topodump.ovpl. This tool has a man page.

First run "ovet\_topodump.ovpl -l filt" to list the filters. If you corrupted the file in some way, this command will fail. Then look for the IfTypeFilter filter just to make sure it's there.

Next try out the filter. Run "ovet\_topodump.ovpl -if -filt IfTypeFilter" to get a list of all the interfaces that match this filter. Make sure that the list matches your expectations.

Now let's test that the polling policies have been applied correctly. As you recall, we set it to not ping any addresses on interfaces of type 32. As with most things, there are

multiple ways to test the polling policy application. The first one is to use the tool `/opt/OV/support/NM/checkpollcfg`. WARNING: This tool can take a very long time to execute if you have a large database. At some sites, it can take 20 or 30 minutes to run.

We don't think there is a man page but you can give the tool a "-?" option for a usage page. We just use the "-o" option along with a node name that you want to check. This tool will print a summary of how interfaces and addresses are being polled based on the latest version of `paConfig.xml` and `TopoFilters.xml`. It's important to notice that this tool runs independent of the poller process `ovet_poll`. The poller can be stopped during this effort. It's fine if you leave the poller running as well but know that the changes to `paConfig.xml` and `TopoFilters.xml` are not picked up by the poller until you restart it.

Let's run the command now:

```
/opt/OV/support/NM/checkpollcfg -o c2k3fa00
```

```
node| ifName| Board| address      IndexNum  pollInterval  isPolled  pollDisabled  snmp?  ping?
```

c2k3fa00.all.fcov	-				300	1	0	1	1
15.2.129.2	-				300	1	0	0	1
Connect to hp4k1sw port H1	IF	[ 1]			300	1	0	1	1
15.2.131.63	-				300	1	0	0	1
Connect to hsrp800t port 1	IF	[ 3]			300	1	0	1	1
15.2.132.3	-				300	1	0	0	1
Connect HSRP to cisco4k3	IF	[ 4]			300	1	0	1	1
15.2.129.2	-				300	1	0	0	1
Connect to cisco2k5 Serial10/0	IF	[ 2]			300	1	0	1	0
172.18.0.2	-				300	0	0	0	0
-	IF	[ 5]			21600	0	0	0	0
-	IF	[ 6]			21600	0	0	0	0
-	IF	[ 7]			300	1	0	1	1

Let me explain a little about this output. First, notice that the "node" and its management address are listed as the first two lines. These indicate the polling policies set on the node level. Then we list the interfaces on the node. These are not in any particular order so watch out for that. Notice that the address on an interface is listed as a separate line below the interface.

The poll interval is given in seconds. The other columns are mostly Boolean values. The "isPolled" value is simply an AND of the SNMP and ping columns.

Getting back to our example now, you can see that ping values on our interface are 0 and thus we won't ping the address on that interface.

At this point, we'll restart the poller.

```
ovstop -c ovet_poll
ovstart -c ovet_poll
```

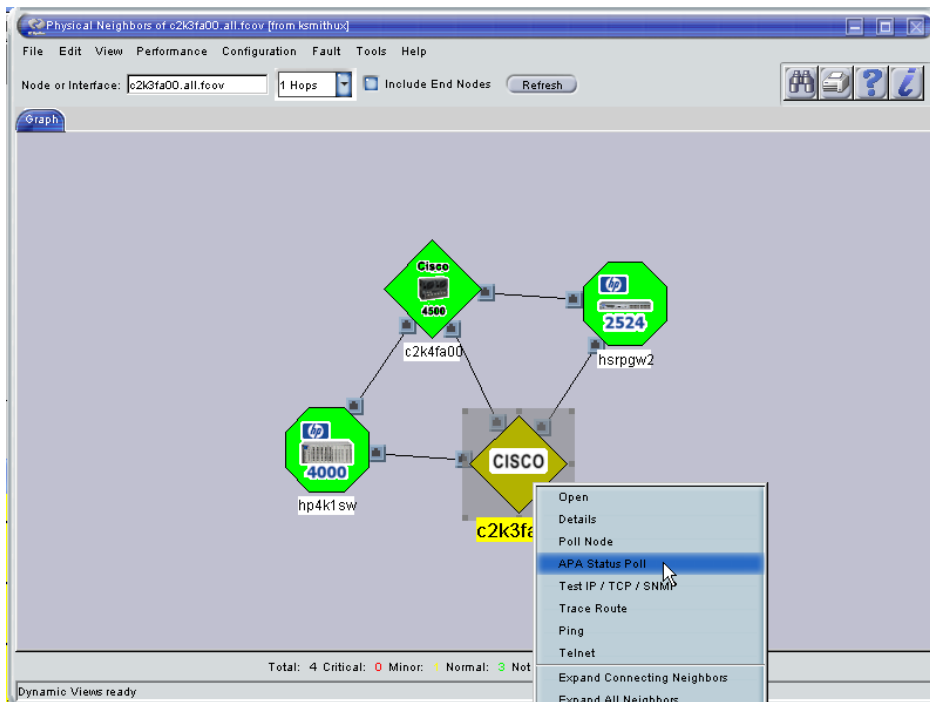
You should now wait for the poller to finish loading everything and begin polling. If you have a large database, this can take a while. You can see when it transitions by running `ovstatus` and checking the Last Message to read "Polling Devices".

```

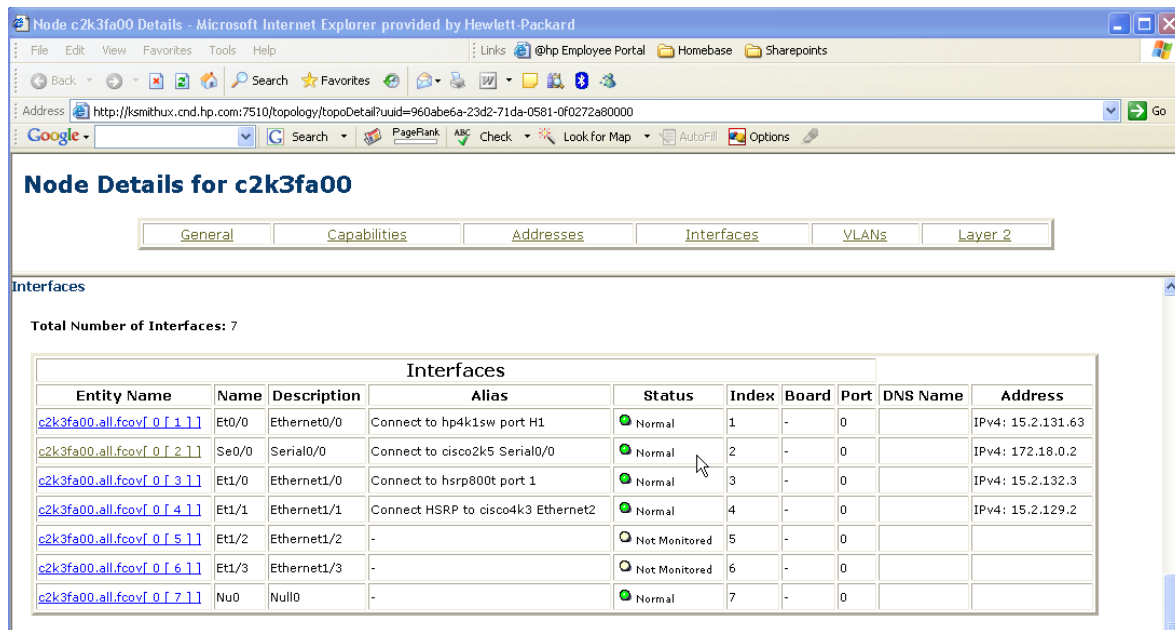
ovstatus -c ovet_poll
Name          PID  State          Last Message(s)
ovet_poll     2926 RUNNING        (ovet_poll-81) Polling Devices.

```

After the poller has begun polling devices, return to the GUI and check the status of the node. It may still be yellow. You might need to run a APA demand poll on the node. To do this, right click on the node and select APA Status Poll. The status of the node should now be up-to-date. In our example, the node should turn green.



Double click on the node to launch the Node Details page. You should see that the interface is now Normal.



If you go down the Addresses section of this page, you'll see that the address is no longer monitored.

IPv4 Address Information			
Address	IP Version	Type	Ping State
15.2.132.3	IPv4	-	responding
172.18.0.2	IPv4	-	not monitored
15.2.129.2	IPv4	-	responding
15.2.131.63	IPv4	-	responding

We used to suggest using the interval value to help diagnose polling settings. I recommend against that now. Instead use the new feature “-d” option in `ovet_demandpoll.ovpl`.

### 10.14 A new valuable addition to `ovet_demandpoll.ovpl`

A valuable feature has been added to `ovet_demandpoll.ovpl`. If you use the `-d` option, it will give you all kinds of important information. Feedback has been added about what polling policy / filter is applied to the current object. Let's go through an example here.

Begin by running the command `ovet_demandpoll.ovpl` with a `-d` option. We are only showing a snippet of the output here showing an address 10.28.16.33. You see the three polling policies namely `snmpEnable`, `pingEnable` and `interval`. You can see the inheritance of the policies. The “Composite” line shows the reason for the composite value rather than a filter. If you look closely at the `snmpEnable` portion, you can see that it will be always false for addresses. That's because we never SNMP poll an address, we only ping addresses.

You can see on `pingEnable` that the composite is false due to the `ET_FILTER` (really a polling policy) `My_Unmonitored_VLAN_Interfaces` having a value of false.

With this output, you'll be able to tell exactly why any object is being polled the way it is.

```
ovet_demandpoll.ovpl -d testnode.corp.com
```

```
=====
ADDRESS 10.28.16.33
=====
OBJ_TYPE  SHORT_OBJECT_NAME  snmpEnable  ET_FILTER
-----
all-types      *                true        DEFAULT
NODE          hp3sw3           true        isSwitch
INTERFACE     hp3sw3[0[53]]   false       My_Unmonitored_VLAN_Interfaces
ADDRESS       10.28.16.33     true        DEFAULT
Composite     10.28.16.33     false       Always false for Addresses

OBJ_TYPE  SHORT_OBJECT_NAME  pingEnable  ET_FILTER
-----
all-types      *                true        DEFAULT
NODE          hp3sw3           true        isSwitch
INTERFACE     hp3sw3[0[53]]   false       My_Unmonitored_VLAN_Interfaces
ADDRESS       10.28.16.33     true        DEFAULT
Composite     10.28.16.33     false       NodeVal && InterfaceVal && AddressVal

OBJ_TYPE  SHORT_OBJECT_NAME  interval  ET_FILTER
-----
all-types      *                900        DEFAULT
NODE          hp3sw3           900        isSwitch
INTERFACE     hp3sw3[0[53]]   300        UnconnectedAdminUpOrTestSwitchIF
Composite     hp3sw3[0[53]]   300        Interface Value
```

*Special note to windows users on ovet\_demandpoll.ovpl:* This perl script calls an executable called /opt/OV/support/NM/ovet\_demandpoll.exe. Due to the way this is structured, you usually can't redirect "standard out" from the perl script. In order to fix this, simply copy the ovet\_demandpoll.exe into the \$OV\_BIN directory and then run ovet\_demandpoll.exe instead of ovet\_demandpoll.ovpl. The arguments are identical.

### 10.15 **Another suggested polling policy**

APA will poll all interfaces on routers by default. You may not wish to poll ISDN interfaces on your routers. We suggest using the IFTYPE policy shown previously to disable polling of ISDN interfaces.

### 10.16 **Example of changing polling policies for specific node types**

In this example we'll skim over a few details that you learned in the previous example. Let's say you have a class of nodes that you don't ever want to ping. You can identify these nodes by their System OID. For this example, we'll use Blue Coat Systems devices. To modify the polling policy for these devices, you'll need to do the following:

- Edit TopoFilters.xml
  - Add node filter based on sysOID
  - Add interface filter defining interface on that type of node
- Edit paConfig.xml

- Add class specification for the nodes based on these filters and define polling parameters
- Add a class specification for interfaces on these nodes and define polling parameters

Let me give a few details. First edit TopoFilters.xml and add the node filter:

```
<nodeAssertion name="isBlueCoat" title="isBlueCoat" description="BlueCoat devices">
  <operator oper="NOOP">
    <attribute>
      <sysOID>1.3.6.1.4.1.3417.1.1.23</sysOID>
    </attribute>
  </operator>
</nodeAssertion>
```

Next add an interface assertion filter to match all interfaces on these nodes. These can be a little more challenging so we highlighted some important parts.

```
<interfaceAssertion name="IFInBlueCoat" title="IFInBlueCoat" description="Interfaces in
BlueCoat devices">
  <operator oper="NOOP">
    <interfaceAssociation ascType="inNode">isBlueCoat</interfaceAssociation>
  </operator>
</interfaceAssertion>
```

You should now test the filter with ovet\_topodump.ovpl. Next you need to put in a polling policy into paConfig.xml. You must put in a policy for the node and then a separate one for the interfaces on the node. Remember that this file is parsed from the top down so you need to be careful where you place this policy. We like to place new policies fairly high in the file so other matches don't take precedence. Shown here is the node specification. (We left out parts of the XML that are represented with an ellipsis "...").

```
<classSpecification>
  <filterName>isBlueCoat</filterName>
  <parameterList>
    <parameter>
      <name>snmpEnable</name>
      ...
      <value>>true</value>
      ...
    </parameter>

    <parameter>
      <name>pingEnable</name>
      ...
      <value>>false</value>
      ...
    </parameter>
  </parameterList>
</classSpecification>
```

Next put in the interface specification.

```
<classSpecification>
  <filterName>IFInBlueCoat</filterName>
  <parameterList>
    <parameter>
      <name>snmpEnable</name>
      ...
      <value>>true</value>
```



```

    ...
  </parameter>

  <parameter>
    <name>pingEnable</name>
    ...
    <value>>false</value>
    ...
  </parameter>
</parameterList>
</classSpecification>

```

After making these changes, verify their correctness and restart APA. Now you have a polling policy in place for these type of nodes and interfaces on these nodes. Now try running “ovet\_demandpoll.ovpl -d <node>” to see if this node and its interfaces are being managed via this new polling policy.

While these examples are fairly simple, I think you get the idea of how to make more complex modifications.

## 10.17 *Improving APA startup time*

(Note: These changes have already been done automatically if you have installed the latest patches, so double check the current definition before changing it.)

### 10.17.1 First Improvement

This change involves changing the logic for the NotConnectedIF filter. This can make a huge difference in startup time.

- Edit \$OV\_CONF/nmet/topology/filter/TopoFilters.xml
- Locate the definition for **NotConnectedIF** and comment it out.

```

<!--
  <filter name="NotConnectedIF" objectType="Interface"
    title="NotConnectedInterfaces" description="Interfaces
    without Layer 2 Connection">
    <operator oper="NOT">
      <filterName>ConnectedIF</filterName>
    </operator>
  </filter>
-->

```

Then find the definition for ConnectedIF and place the following new interface assertion above it (in blue below). You can use the existing ConnectedIF as a sample but don't forget to change the oper to NOT.

```

<!-- This is the new one -->
  <interfaceAssertion name="NotConnectedIF" title="ConnectedInterfaces"
description="Interface without L2 Connection">
    <operator oper="NOT">
      <attribute>
        <capability>isL2Connected</capability>
      </attribute>
    </operator>
  </interfaceAssertion>

<!-- This one stays the same -->
  <interfaceAssertion name="ConnectedIF" title="ConnectedInterfaces"
description="Interface with L2 Connection">

```

```

        <operator oper="NOOP">
            <attribute>
                <capability>isL2Connected</capability>
            </attribute>
        </operator>
    </interfaceAssertion>

```

- Run `ovet_topodump.ovpl -lfilter` to make sure you didn't make any mistakes.

## 10.17.2 Second Improvement

- Edit `$OV_CONF/nnmet/topology/filter/TopoFilters.xml`
- Locate the definition for **AdminUpOrTestInterface**
- Remove or comment out the existing definition

```

<!--
<interfaceAssertion name="AdminUpOrTestInterface" title="AdminUpOrTestInterface"
description="Interface with administrative state up or testing">
    <operator oper="OR">
        <attribute>
            <ifAdminState>up</ifAdminState>
        </attribute>
        <attribute>
            <ifAdminState>testing</ifAdminState>
        </attribute>
    </operator>
</interfaceAssertion>
-->

```

- Add the definition as shown below

```

<interfaceAssertion name="AdminUpOrTestInterface" title="AdminUpOrTestInterface"
description="Interface with administrative state up or testing">
    <operator oper="NOT">
        <attribute>
            <ifAdminState>down</ifAdminState>
        </attribute>
    </operator>
</interfaceAssertion>

```

## 10.17.3 Third Improvement

HP shipped a filter that is slow and should be removed and replaced with simply pinging all switches. Here are the steps that you should follow.

- Backup original `paConfig.xml`
- Edit `paConfig.xml`
- Comment out **IFInNotConnectedSwitch** filter `<!-- -->`
- Change **isSwitch** ping setting to true (`pingEnable=true`)
- Change **UnconnectedAdminUpOrTestSwitchIF** ping setting to true (`pingEnable=true`)
- Confirm new settings with `checkpollcfg` (switch address is pinged)
- restart `ovet_poll`

## 10.18 Changing the thread count in APA

If you see polling or analysis getting behind and you've done your best to reduce the number of polled objects with the techniques described previously, then you might try increasing the number of threads for polling and analysis. Be cautious when increasing

these values. It will use more system resources. We've had good luck by doubling the sizes but we know of customers that have gone even larger. The values don't need to be a multiple of 2.

To change the number of threads, edit paConfig.xml and look for the sections shown below. Increase the values and restart APA. We recommend trying 32 for PollingEngineThreadPoolSize and 20 for statusAnalyzerThreadPoolSize.

```
<parameter>
  <name>PollingEngineThreadPoolSize</name>
  <title>Polling Engine Thread Pool Size</title>
  <description>This parameter specifies how many threads are in the polling engine thread pool. Increasing the value of this parameter, increases the amount of pooling parallelism and the amount of system resources consumed (e.g. number of file descriptors). Increasing the value of this parameter may or may not increase polling engine performance.</description>
  <varValue>
    <varType>Integer</varType>
    <value>32</value>
  </varValue>
</parameter>

<parameter>
  <name>statusAnalyzerThreadPoolSize</name>
  <title>Status Analyzer Thread Pool Size</title>
  <description>This parameter specifies how many threads are in the status analyzer thread pool. Increasing the value of this parameter, increases the amount of analysis parallelism and the amount of system resources consumed (e.g. number of file descriptors). Increasing the value of this parameter may or may not increase status engine performance.</description>
  <varValue>
    <varType>Integer</varType>
    <value>20</value>
  </varValue>
</parameter>
```

## 10.19 *Reducing the memory footprint in APA*

APA (ovet\_poll) can have a very large memory footprint. At some customer sites, this has been as grown to over 1 Gig. With the current design, APA loads all the polled objects from the topology into RAM.

There is a setting that you can modify to help reduce this footprint. Edit paConfig.xml and modify value for the parameter loadOnlyPolledObjectsIntoMemory. You can change this from false to true. This instructs APA to only loaded the “polled” objects into RAM thus reducing the footprint.

But there is a side-effect that you need to be aware of. If you discover a new connection as part of modifying something in ET discovery, that new connection may not be polled until you restart APA. We recommend only setting this value if you are up against memory limitations. We also recommend that you wait until you are finished tuning APA to set it true.

```
<parameter>
  <name>loadOnlyPolledObjectsIntoMemory</name>
  <title>Load Only Polled Objects Into Memory</title>
  <description>Flag that controls loading of unpolled nodes and interfaces into memory. If loadOnlyPolledObjectsIntoMemory = true, then nodes and interfaces that are not polled will not be loaded into APA memory.This results in faster startup performance and a
```

```

smaller memory footprint.It is recommended to use this setting only for large
environments.</description>
  <varValue>
    <varType>Bool</varType>
    <value>true</value>
  </varValue>
</parameter>

```

On HPUX PA-RISC, there are challenges allowing a process to grow larger than 1 Gig RAM. Sometimes we had to increase the maxsize to 2GB and run the following chatr commands on ovet\_poll executable. This is “sticky” and only needs to be done once or at least until there is a change to the ovet\_poll binary like possibly after installing a new patch or upgrade.

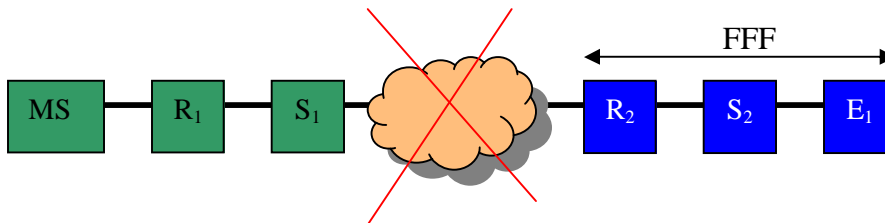
```
chatr +q3p enable /opt/OV/bin/ovet_poll
```

On Itanium platforms, you need to do things slightly different. The binary must be compiled to change the default memory model before you can execute this command. We have recompiled ovet\_poll and ovwdb on Intermediate Patch 16 (IP16) to allow this chatr command. Other binaries like ovet\_disco and ovas will be compiled this way in IP17. If you need these sooner, contact support.

```
/usr/bin/chatr +as mpas /opt/OV/bin/ovet_poll
```

## 10.20 *Side effects of islands of connectivity*

If you have “islands of connectivity” (2 or more nodes that are disconnected, per ET, from other nodes) and the island becomes unreachable via ping and SNMP, APA will not issue alarms for this island since it considers these faults as secondary. For example, in the illustration below, a failure in the cloud prevents packets from flowing from the MS to R<sub>2</sub>, S<sub>2</sub> or E<sub>1</sub>. APA Analysis will conclude that these nodes are all Far-From-Fault and turn them all blue w/o generating any alarms.



There are workarounds that can help:

- Mark nodes in the island as “important nodes” via the file MyHostID.xml, paConfig.xml and TopoFilters.xml. Search for “Important” in these files for more details. APA will then issue alarms on secondary failures on these nodes
- Mark nodes as “critical nodes” via CriticalNodes.xml, paConfig.xml and TopoFilters.xml.
- Connect up “islands to mainland” by adding manual connections via connection editor

- Currently NNM 7.53 is planning to issue new alarms for “Remote Site Unreachable”. This has been well received in beta testing. When released, you’ll probably need to enable this feature in paConfig.xml.

## 11 Reducing Traps and Events

At many customer sites, NNM is receiving thousands of traps that it doesn’t use. Unless you really need to see these traps, we recommend that you filter them out of the system. This is the number one cause of PMD process instability.

As of this writing, the only traps that APA uses (provided ReceiveEvents parameter in \$OV\_CONF/nmnet/paConfig.xml is set to “true”) are shown here:

CiscoColdStart	1.3.6.1.6.3.1.1.5.1
CiscoLinkDown	1.3.6.1.6.3.1.1.5.3
CiscoLinkUp	1.3.6.1.6.3.1.1.5.4
CiscoWarmStart	1.3.6.1.6.3.1.1.5.2
ColdStart	1.3.6.1.6.3.1.1.5.1
HSRPState	1.3.6.1.4.1.9.9.106.2.0.1
LinkDown	1.3.6.1.6.3.1.1.5.3
LinkUp	1.3.6.1.6.3.1.1.5.4
chassisChangeNotifOID	1.3.6.1.4.1.9.5.11.2.0.2
StackMIBModuleDown	1.3.6.1.4.1.9.5.0.4
StackMIBModuleUp	1.3.6.1.4.1.9.5.0.3
WarmStart	1.3.6.1.6.3.1.1.5.2

All others are simply passed through the event pipeline and written into the Binary Event Store (BES). With a high trap reception rate, you’ll probably be rolling your BES at a high rate (like every 4 hours). You don’t want the BES to roll over too often otherwise you can’t dump a nice history of events for diagnosis.

Even if a trap is set to log only, it will still pass through the event pipeline and bog down the system. It is best to “nip it in the bud” and remove the trap at the moment it is received by NNM.

We have a new feature called ovtrapd.conf that can be used to filter traps. There is a nice whitepaper called /opt/OV/doc/WhitePapers/EventReduction.pdf that is shipped with NNM 7.51. So we won’t detail the mechanics of using the feature but instead show a good process.

In order to see what are most frequent traps that you are receiving, you want to run the command “\$OV\_BIN/ovdumpevents” to get an readable format of the current BES.

```
ovdumpevents > /tmp/dumpevents.txt
```

Then you can evaluate the data looking for trends. This file isn’t the easiest thing to read. There is a support tool (/opt/OV/support/processEvents) that can help but it isn’t well

maintained so we don't guarantee anything. You pass the dumpevents output file as generated in the previous step. You also provide it an empty file called the summaryfile where it writes results. We get a number of errors when we run it but you can ignore them. *Plus note that the script processEvents has a problem running on Solaris.* To fix this, you'll need to edit the file and change all of the instances of the work "awk" to "nawk".

```
/opt/OV/support/processEvents dumpevents.txt summaryfile
```

The summary file will contain information about the number of trap occurrences in the BES. Lines will look like:

```
Total Number for trapId .1.3.6.1.4.1.9.9.43.2.0.1 = 5274  
.1.3.6.1.4.1.9.9.43.2.0.1 is not an OV_event
```

The trap OID is highlighted above. If you wanted to find out what node this trap is coming from, simply grep for the OID in the dumpevents.txt file.

Let's suppose that you decide this trap is not of interest to you. A lot of times you won't have a MIB defining the trap. We find that you can search the internet for most OIDs and get good results. We googled the OID above and found that it is a ciscoConfigManEvent which is a notification of a configuration management event.

To remove this trap, you follow the instructions for ovtrapd.conf and place the trap in the file as shown here. You can add additional traps as well. You can also restrict this to a set of IP addresses but for this example we'll suppose we want to throw away that trap from any source IP address. To do that, put the following entry into the file. Note that you can add multiple traps to the same line. You can also use wildcarding. But you can't put the same IP on multiple lines so you must put them on the same line with commas separating them. Try to make the line as short as possible using wildcards.

```
<*. *.* , .1.3.6.1.4.1.9.9.43.2.0.1 , .1.3.6.1.4.1.9.5.0.*>
```

In a future, we hope to develop the ability to specify what traps you want to keep rather than what to exclude. That way you could simplify this file.

We also have implemented the ovtrapd filtering in such a way that you can specify the number of seconds as criteria for calculating the event rate. That is, now a new option called "-c" has been added to ovtrapd to handle the number of seconds criteria for trap filtering. For example, if we want to block an IP address/interface from which we get 10 events every minute (60 seconds) then we need to create an entry like "OVs\_YES\_START:pmd:-b 10 -c 60:OVs\_WELL\_BEHAVED::" in ovtrapd.lrf file and here -b is number of events and -c is number of seconds for which we want to monitor the event rate.

Please note that this event rate will get calculated on receipt of every <n> number of events specified with -b option. In example mentioned above, we will block the device/IP (all the events) if 100 events are generated from it in less than 60 seconds and then we'll

monitor the device again for next 100 events and see if we have these events generated in less than 60 sec and if not, then we'll unblock the device/IP and events will resume to enter in NNM from these IPs.

Also note that any option specified in ovtrapd.lrf does not work along with ovtrapd.conf options. All the IPs specified in ovtrapd.conf are blocked forever from sending traps to NNM station.

Following is an example to enable tracing for ovtrapd to get a list of blocked devices. Please change the \$OV\_LRF/ovtrapd.lrf file as follows:

```
"OVs_YES_START:pmd:-b 10 -c 60:OVs_WELL_BEHAVED::"
```

We recommend that you do this process a number of times and try to cut down on the number of traps. Try to get it down to one trap per minute.

You can do a little bit of cutting down of NNM generated events by changing the category of them from LOGONLY to IGNORE. This is done by editing trapd.conf or in the Event Configuration GUI. In the GUI, IGNORE is called "Don't log or display" But use caution when doing this. Some of these alarms are used by other parts of the system.

An example of an alarm you might choose to ignore is the OV\_Bad\_Subnet\_Mask. Maybe you don't ever care about those. So rather than making them LOGONLY, make them IGNORE instead. You can do this by editing trapd.conf directly. Make sure you create a backup first. After making the change, run "xnmevents -event" and "xnmtrap -event" to read the change in.

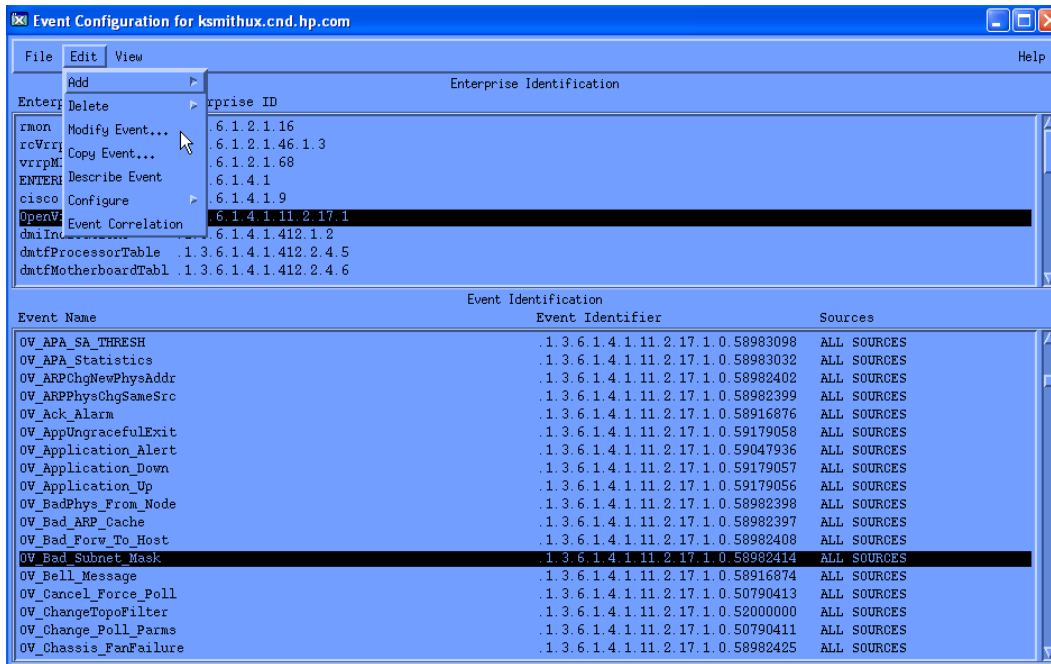
#### Before

```
EVENT OV_Bad_Subnet_Mask .1.3.6.1.4.1.11.2.17.1.0.58982414 "Configuration Alarms" Minor
```

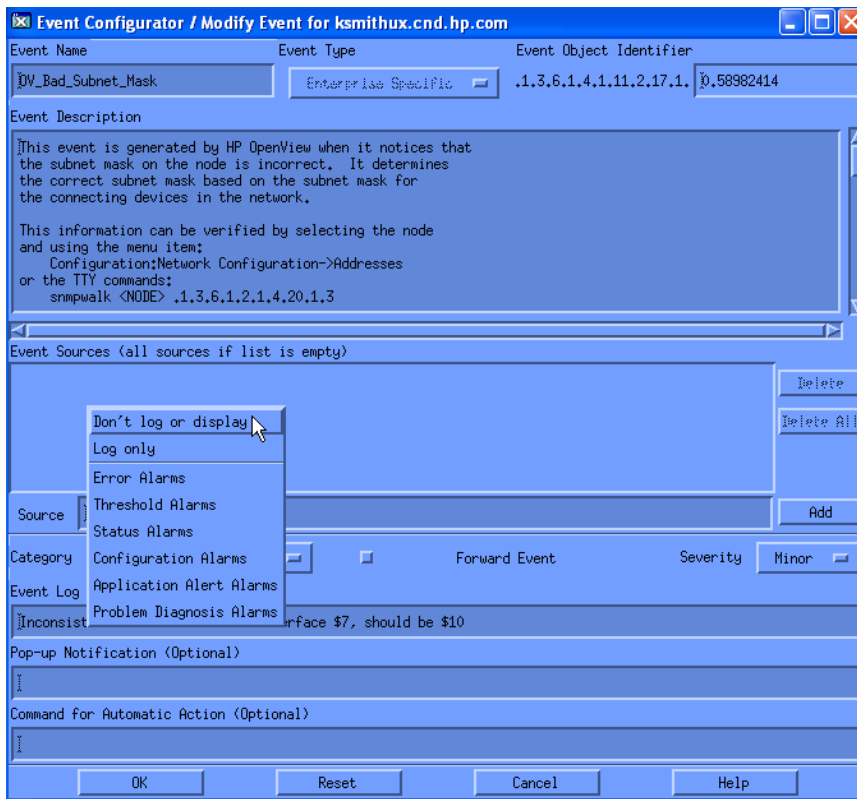
#### After

```
EVENT OV_Bad_Subnet_Mask .1.3.6.1.4.1.11.2.17.1.0.58982414 "IGNORE" Minor
```

We recommend that you use the GUI instead to make this change. Select the event and select Modify Event...



Then change the Category to “Don’t log or display”. Then choose OK and select File -> Save to make the change active.

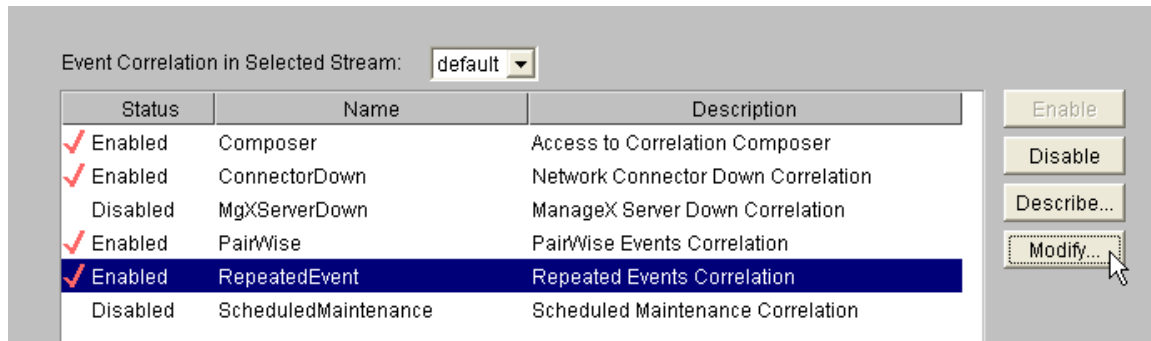


If you change events to IGNORE, then you should check to see if the alarm has any correlations associated with it. If so, you should disable them. Even though the event is

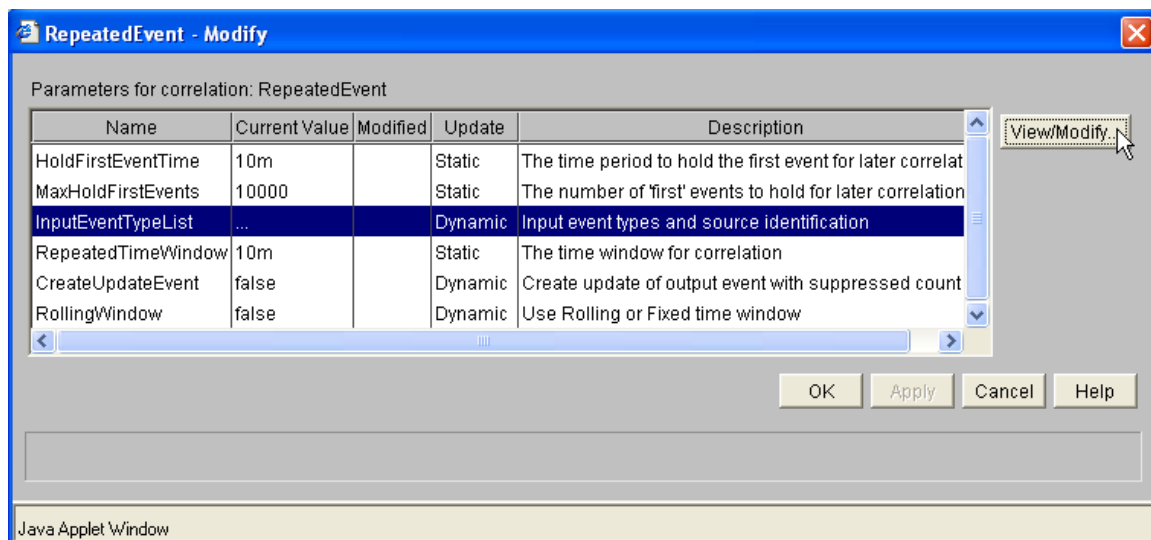


being ignored, it still flows through the event system and will potentially cause extra correlation work and internal events.

Let's continue our example of .1.3.6.1.4.1.11.2.17.1.0.58982414. Let's look at the RepeatedEvent circuit to check if it is correlated. Launch the ECS Configuration GUI.



Click Modify on the circuit. Then click View/Modify on the InputEventTypeList.



Then look for the event OID .1.3.6.1.4.1.11.2.17.1.0.58982414. You can see that it is in the list. We suggest you change value to false so it won't be correlated.

The screenshot shows a window titled "InputEventTypeList - Modify". It has a "Name:" field with "InputEventTypeList" and a "Type:" field with "Dictionary". Below is a "Table Values:" section containing a table with three columns: "Input Event OID / Generic Trap String", "Event Source Identification", and "Enable Event Type". The table contains 15 rows. The 10th row, corresponding to the OID .1.3.6.1.4.1.11.2.17.1.0.58982414, is highlighted with a mouse cursor. To the right of the table are three buttons: "Add Row", "Delete Row", and "Verify Table". At the bottom, there is a "Current Value:" field with "true" and a "Description:" field.

Input Event OID / Generic Trap String	Event Source Identification	Enable Event Type
1.3.6.1.4.1.11.2.17.1.0.58916867	6	true
1.3.6.1.4.1.11.2.17.1.0.58916864	1.3.6.1.4.1.11.2.17.2.2.0	false
"authenticationFailure"	"agent-addr"	false
"egpNeighborLoss"	"agent-addr"	false
1.3.6.1.4.1.11.2.17.1.0.58982422	1.3.6.1.4.1.11.2.17.2.2.0	true
"linkUp"	"agent-addr"	false
1.3.6.1.4.1.11.2.17.1.0.58982401	1.3.6.1.4.1.11.2.17.2.2.0	true
1.3.6.1.4.1.11.2.17.1.0.58982423	6	true
1.3.6.1.4.1.11.2.17.1.0.40000011	6	true
1.3.6.1.4.1.11.2.17.1.0.58982414	1.3.6.1.4.1.11.2.17.2.2.0	true
"linkDown"	"agent-addr"	false
"warmStart"	"agent-addr"	false
"coldStart"	"agent-addr"	false
1.3.6.1.4.1.11.2.17.1.0.58785794	""	true

## 12 Backup Strategies

When using Extended Topology, you need to decide the best way to backup the Solid database since that is the primary database used by ET. Extended Topology will make your Solid DB much larger than it used to be. Many large networks can use up at least 2 Gigabytes of space.

By default, Solid is setup to automatically run a Solid backup nightly at 11pm. In addition to this, if you use ovbackup.ovpl, it also backs up the Solid database.

When you perform ovbackup.ovpl or the automatic Solid backup, the backup is written to the \$OV\_DB/analysis/default/backup subdirectory. This is usually on the same volume as the Solid database (just one directory up). It's not a good practice to backup onto the same volume as the original data. For Unix customers, we recommend that you create a symbolic link for this backup subdirectory. Link it to an different volume on another disk.

You can also change the location of the Solid backup but it's a little tricky. To change the location of the Solid backup, edit \$OV\_DB/analysis/default/solid.ini. There is a section that reads:

```
[General]
BackupDirectory=backup
```

This directory is relative to Solid working directory. You cannot put an absolute path here like `/var/tmp/backup`. You could do some “dot-dotting”. On Unix, to get to the directory `/var/tmp/solidbak`, change the file to read:

```
[General]
BackupDirectory=../../../../../../../../tmp/solidbak
```

Another alternative is to leave the backup in the backup subdirectory but move the contents to another volume immediately after completing the backup.

We usually recommend disabling the automatic Solid backup and using the `ovbackup.ovpl` explicitly to backup your data. To do this, edit the file `$OV_DB/analysis/default/solid.ini` and put a semicolon in front of the line that reads “`At=23:300 backup`” and restart solid (`ovdbcheck`). This requires stopping many other NNM processes so do this during a scheduled downtime.

```
; The following entry will schedule a embedded database backup each night
; at 11:00pm. Remove when using ovbackup.
; At=23:00 backup
```

## 13 GUI

There are a few simple changes for the GUI that we recommend everybody use. We’ll detail these first. We’ll also discuss container views.

### 13.1 *Simple Changes*

#### 13.1.1 Changes to `web.xml`

The following changes are all made within the file `$OV_AS/webapps/topology/WEB-INF/web.xml`.

- The first simple step we recommend is that you change the default Neighbor View hop count to one. In a complex network, a Neighbor View with 2 hops can be too complex. Starting with 7.51, you can now adjust this default value. To do this, change the `defaultNumHops` value from 2 to 1.

```
<init-param>
  <param-name>defaultNumHops</param-name>
  <param-value>1</param-value>
</init-param>
```

- The second thing we always do is change Node View to use ET filters rather than classic NNM filters. ET filters are much faster. Change the value for `enableETFilters` from 0 to 1.

```
<init-param>
  <param-name>enableETFilters</param-name>
  <param-value>1</param-value>
</init-param>
```

- The third thing we recommend is to change Interface View to not show disabled interfaces. Change the parameter includeDisabled from 1 to 0.

```
<init-param>
  <param-name>includeDisabled</param-name>
  <param-value>0</param-value>
</init-param>
```

- The fourth recommendation is to restrict the Neighbor View to only use ET data. Change the value for restrictToET from 0 to 1.

```
<init-param>
  <description>
    This parameter will limit the NeighborView to use ET Database only. When
    this feature is enabled NeighborView retrieves information only from the
    ET Database and does not use Base Topology.
    The default value of restrictToET is 0 and this feature will be disabled.
    To enable this feature change the value to 1.
  </description>
  <param-name>restrictToET</param-name>
  <param-value>1</param-value>
</init-param>
```

### 13.1.2 Changing the java heap size

You may wish to increase your java heap size for ovas. This will help reduce running out of memory in complex environments. To do this you simply add “-Xmx512m” to the arguments of ovas in the ovas.lrf file. This specifies a heap size of 512 Meg. You can go as large as 1024m but we usually don’t recommend going higher than that. If you go this big on HPUX paRisc, then you’ll need to run the chatr command on ovas as shown previously for ovet\_poll. The default is 256MB if nothing is specified. You can follow these steps:

1. Edit /etc/opt/OV/share/lrf/ovas.lrf
2. Follow instructions in the comments at the top of this file to add an argument to the ovas startup. Insert the following string between the second and third colons of the second line:

```
-Xmx384m
```

The resulting string should look something like:

```
OVs_YES_START:ovtopmd,ovdbcheck,httpd:-Xmx512m:OVs_WELL_BEHAVED:15:NOPAUSE:
```

3. Run “ovaddobj \$OV\_LRF/ovas.lrf”

### 13.1.3 Changing Topology Cache

Another recommendation is to increase java topology cache by creating **dynamicViewCacheSize.txt** file under \$OV\_CONF for ovas performance reasons though we don’t do this at many sites.

nodeCacheSize The number of nodes to be stored in cache

interfaceCacheSize The number of interfaces to be stored in cache

addressCacheSize The number of addresses (IPV4,IPV6) to be stored in cache

subnetCacheSize The number of subnet (IPV6) to be stored in cache

vlanCacheSize The number of vlan to be stored in cache

HSRPCacheSize The number of HSRP to be stored in cache  
VRRPCacheSize The number of VRRP to be stored in cache

nodeCacheSize	10000
interfaceCacheSize	150000
addressCacheSize	2000
subnetCacheSize	400
vlanCacheSize	400
hsrpCacheSize	60000
ifcCacheSize	100000
cardCacheSize	20000

nodeCacheSize should be more than the value of “ovtopodump -l | grep "NUMBER OF MANAGED NODES"”.

interfaceCacheSize should be more than the value of “ovtopodump -l | grep "NUMBER OF INTERFACES"”

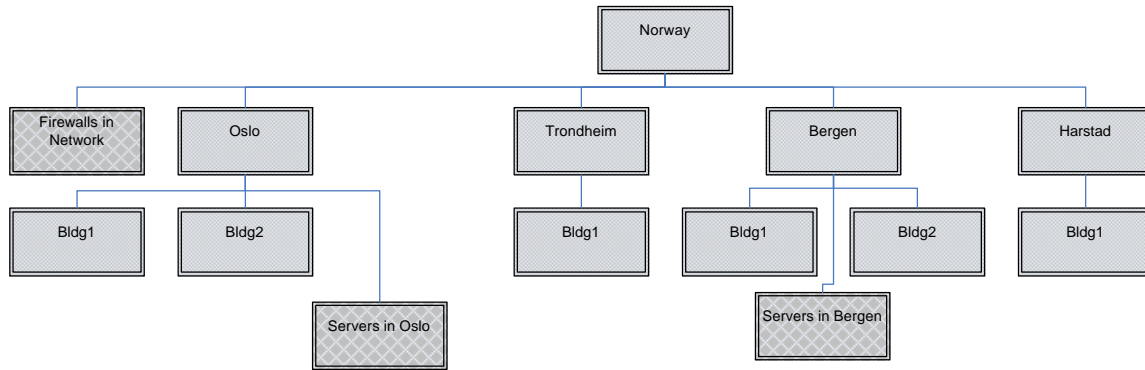
## 13.2 **Container Views**

There is a nice whitepaper written on Container Views so we won't go over the same material here. Please read this whitepaper located at </opt/OV/doc/WhitePapers/ContainerAdministration.pdf> available with NNM 7.51. Instead, we'll discuss some strategies for Container Views.

Containers are based on ET Filters so you must understand ET Filters. Please see the section Filters in ET.

Containers are a great way to divide up your ET Nodes so they can be more easily viewed and monitored from a map. It is important to note that containers only apply to nodes (not other entities like interfaces, HSRP groups, etc.) It's also important to note that nodes can be in more than one container. The only real restriction on containers is that they must be hierarchical and the each container name must be unique. So you'll need to be creative on your layout.

Most customers prefer to create containment based on geography using name based filters. For example, if you have four cities that your nodes are in and then have buildings within these cities, you might create a hierarchy similar to that shown below. Notice that in addition to geography, you can create containers with based on logic like System OID. These logical containers are represented with checkered background while geographical containers are the other shade of gray.



Let’s develop this example. Your first step is to design your hierarchy on paper. I don’t recommend doing this real time in the GUI because it is currently difficult to modify container positions in the hierarchy. The only the problem with the layout above is that it has repeated names like Bldg1. This isn’t allowed in Container Views. So you would need to name the container Oslo\_Bldg1 and Bergen\_Bldg1 or something like that.

The next step is to develop filters for each container. You would probably develop filters based on name or SysName if you use a naming convention that indicates location. For the logical containers like Firewalls in Network, you would probably use a sysOID filter.

The final step would be create the containers in the GUI following the whitepaper instructions and assigning filters to them.

### Best Practices

- We recommend against using the “Include Node . . .” menu item if you are using filters. The two don’t mix well. It’s best to create filters and then strictly use filters for containment.
- You can use JPG or GIF files for background images. We prefer JPG as they resize better.
- We prefer to not show connectivity between containers by setting `showContainerConnectivity=“false”` on all containers. There is an example in the whitepaper.
- Don’t put too many nodes in a container. They are recalculated every time you enter one. We try to cap the size around 200 nodes or so.
- If you don’t care about connectivity, set `showNodeConnectivity=“false”` on all your containers. This makes them much higher performance. You can always launch a neighbor view for connectivity from a node. In my opinion, connectivity within a container can be problematic because key nodes may not pass the filter and may not be in the container, thus possibly degrading the accuracy of the connectivity.

## Default behavior of containers status propagation:

- Containers behave the same way as that of APA for status propagation and following are the different status propagation metrics for containers:
  - Container turns Minor/Yellow if any underlying node is Minor or Critical, irrespective of other nodes' status.
  - Container turns Critical if all the underlying nodes are marked Critical
  - Container turns Normal if everything underlying the container is Green/Normal
  - The main difference between APA and containers is, Container turns "Unknown"/Blue if all the underlying nodes in the container turn Unknown/Blue.
  - This means that Container status does not propagate in APA style if underlying nodes are Unknown. For example, if all the nodes underlying the container are Unknown except one then the container status will be the status of that remaining one node.
  - For example, when a Container was "Minor" due to 1 Minor + 1 Normal + <n> Unknown nodes underlying. When a "Minor" node became "Unknown", the container changed to "Normal" as mentioned above in the text.

This behaviour is happening due to the fact that containers are implemented to indicate the root cause/primary failure of a problem. You get an overall picture of your network on where the failure has happened (shown with a Minor container). Since, the Unknown nodes are secondary failure, they are the impact of the root cause but Containers are not meant for showing the impact of secondary failures.

Better idea to see the impact of the failure would be to look for the container with a root cause node, go inside that container, open a 1-2 Hops neighbour view for the affected root cause node. To go beyond 2 hops in neighbor view, we recommend using "Expand connecting neighbors" option of a "Right-click" menu on a particular node.

### 13.3 *Advanced Container View Tasks*

There are a few things not covered in the containers whitepaper that we'll address here including deleting containers, renaming containers, scaling background graphics, resetting containers. The current plan is for NNM 7.53 to support deleting and renaming via the GUI.

#### 13.3.1 **Delete Container**

To delete a container, you can't use the GUI. Instead you must edit the file \$OV\_DB/nnmet/containers.xml. We recommend that you ovstop ovas when you edit this file since it could be changed by ovas. You then remove any containerReference tags and the entire container tag for the container you wish to delete. Some examples are highlighted in red below. You would completely delete the lines shown in red.

You can use the reload feature (<http://<hostname>:7510/topology/home?reloadContainers=true>) if you prefer to leave ovas running while making these modifications but we have found it dangerous to edit this file with ovas running.

A note of caution: Due to a minor defect, the containers.xml file may contain many blank lines (sometimes hundreds of blank lines). Watch out for that. It can be misleading. You might think you are at the end of the file when you aren't. You can delete the blank lines if you wish.

```
<container name="Fort Collins">
  <containerReference name="MPLS nodes" x="356.8442951222268" y="182.8442951222268"/>
  <containerReference name="NTC" x="531.3368696289949" y="180.3368696289949"/>
</container>

<container name="MPLS nodes">  <filter name="MplsName"/>
  <topoNode name="mplspe06.cnd.hp.com" ... />
  <topoNode name="ntc6k01.cnd.hp.com" ... />
</container>
```

### 13.3.2 Rename Container

To rename a container, you must edit the file \$OV\_DB/nnmet/containers.xml. Again we suggest you stop ovas during when making this change. Then simply replace

name="old name"

with

name="new name"

in all locations. These tags are highlighted in red below.

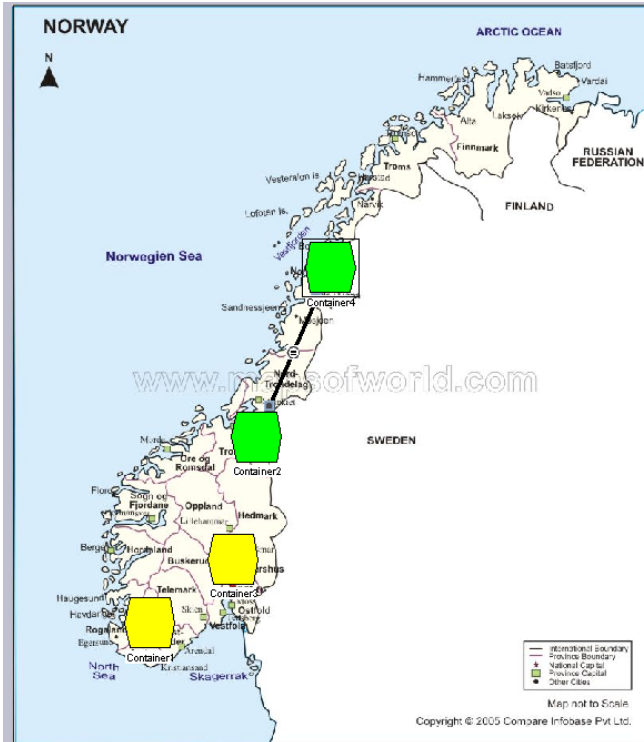
```
<container name="Fort Collins">
  <containerReference name="MPLS nodes" x="356.8442951222268" y="182.8442951222268"/>
  <containerReference name="NTC" x="531.3368696289949" y="180.3368696289949"/>
</container>

<container name="MPLS nodes">  <filter name="MplsName"/>
  <topoNode name="mplspe06.cnd.hp.com" ... />
  <topoNode name="ntc6k01.cnd.hp.com" ... />
</container>
```

### 13.3.3 Scaling Background Graphics

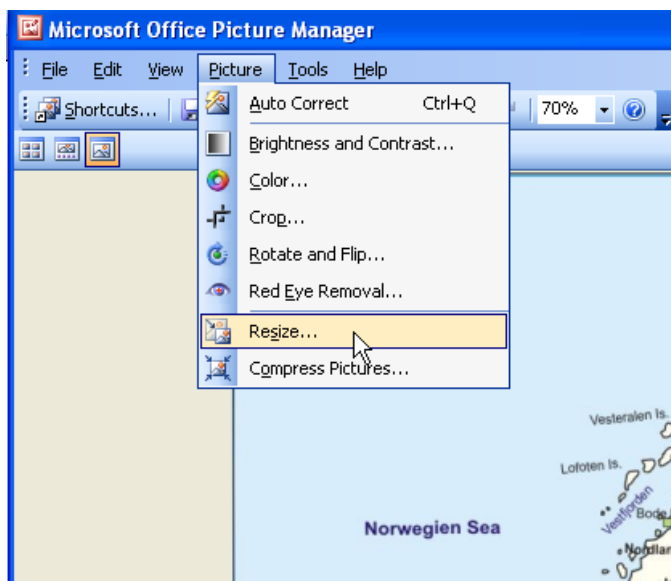
Sometimes you might wish to change the size of the container icon compared to the background graphic. For example, the picture shown here has fairly large container icons. You may prefer that they be smaller.



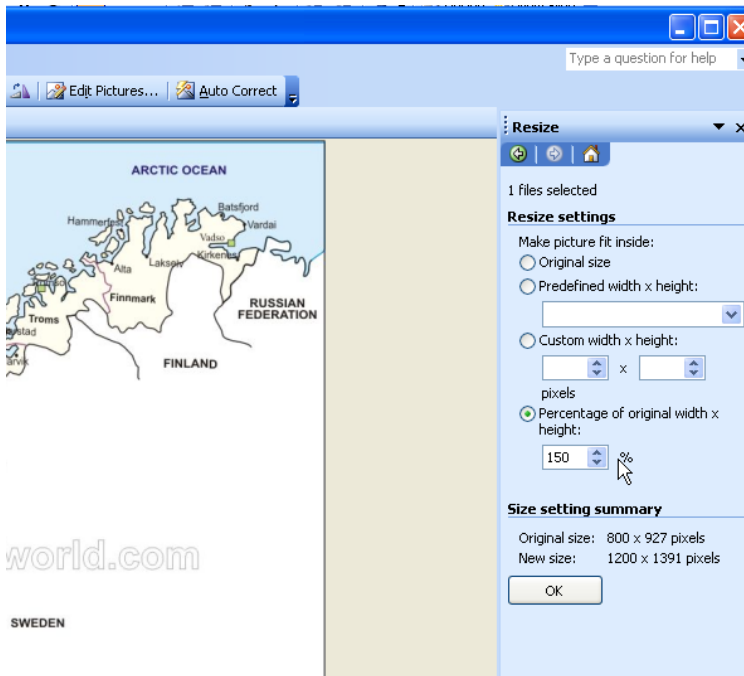


If you want to change the size of the container relative to the background graphic, you must increase the size of the background graphic. I use Microsoft Office Picture Editor since it comes with Microsoft Office. You can also use any Picture Editor you prefer. Let me show the steps using the Microsoft product.

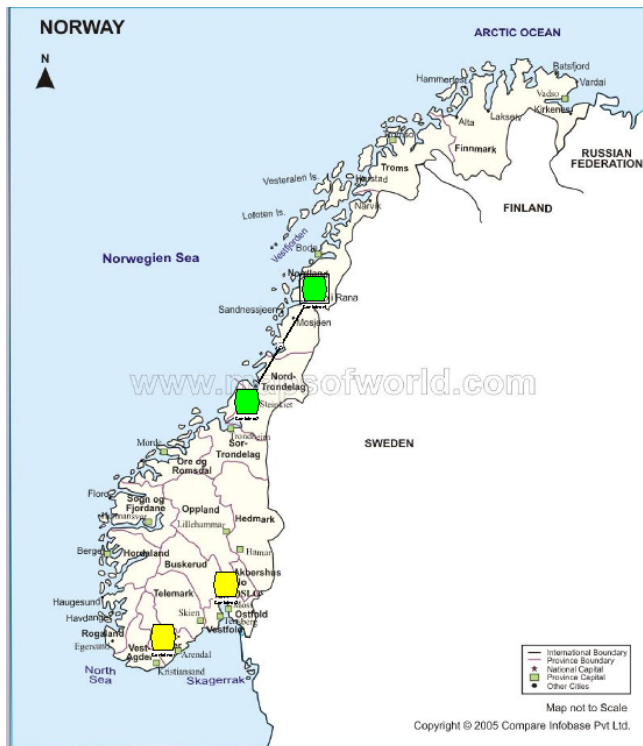
First, load the original graphic into the Picture Editor. Then select Picture -> Resize...



Increase the size of the picture. We like to increase it as a percentage of the original.



Save this new JPEG file. We would then place the new file on top of the old file under \$OV\_WWW/htdocs/images/backgrounds. You can either reload it into ovas (<http://<hostname>:7510/topology/home?reloadContainers=true>) or restart ovas. Then you should see a view with containers scaled to a smaller size.



### 13.3.4 Resetting Containers

We've seen occasionally customers want to "reset" or "rebuild" a container. Why ever do this? If you have radically changed your NNM database and this caused all the NNM Object IDs to change (like deleting the NNM database and rediscovering the network completely), this may cause challenges to containers since they store the NNM Object ID as well as position. If you only populated your containers with filters, then simply select the container of interest, choose Set Container Filter, and select the same filter as before. This will repopulate the container. FYI, this will also generate many blank lines in the containers.xml file so watch out for this. It doesn't hurt anything can be misleading when editing the file by hand.

## 13.4 Adding Authentication to HomeBase

You might require authentication to HomeBase. This can be easily added.

Edit \$OV\_AS/webapps/topology/WEB-INF/web.xml and remove the XML comments highlighted below to activate the block of XML.

```
<!--
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>Dynamic View Access</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>operator</role-name>
      <role-name>administrator</role-name>
    </auth-constraint>
  </security-constraint>
-->
```

Then add users and assign roles in this file:

\$OV\_AS/webapps/topology/WEB-INF/dynamicViewsUsers.xml

You probably already set up one user at installation time. You can just use that one if you wish or you can add others.

Notice that the users are added between the tomcat-users tags. So the original looks like this if you have one user with "admin" and "secret" name and password.

```
--><tomcat-users>
<user name="admin" password="secret" roles="administrator"/></tomcat-users>
```

To add additional users, make it look like the XML shown below or better yet, you can use the tool \$OV\_BIN/dvUsersManager.ovpl. There is a "man" page available for this tool.

```
-->
<tomcat-users>
<user name="user1" password="topsecret" roles="administrator"/>
<user name="user2" password="evenmoretopsecret" roles="administrator"/>
<user name="admin" password="secret" roles="administrator"/>
</tomcat-users>
```

NOTE: This security doesn't work all that well in the webstart solution because you keep having to authentic the browser whenever you launch to a browser from homebase (like Node Details).

The login is done using BASIC method of authentication, which sends passwords as cleartext across the network. You can instead use the more secure MD5 passwords. To use this, instead of entering the cleartext password in the dynamicViewsUsers.xml file, you must enter the result of

```
"$OV_JRE"/bin/java -classpath \  
"$OV_AS"/server/lib/catalina.jar:"$OV_AS"/bin/bootstrap.jar \  
org.apache.catalina.realm.RealmBase -a MD5 topsecret
```

where topsecret is the password you want to enter.

Further on IPF, you need to use

```
"$OV_JRE"/bin/IA64N/java -classpath \  
"$OV_AS"/server/lib/catalina.jar:"$OV_AS"/bin/bootstrap.jar \  
org.apache.catalina.realm.RealmBase -a MD5 topsecret
```

and make sure that LD\_LIBRARY\_PATH is set to:

```
$OV_JRE/lib/IA64N:$OV_JRE/lib/IA64N/server
```

This will return a string like:

```
topsecret:ea847988ba59727dbf4e34ee75726dc3
```

and you will need to use "ea847988ba59727dbf4e34ee75726dc3" as the password field in the dynamicViewUsers.xml file.

It will now look something like:

```
<tomcat-users>  
<user name="user1" password=" ea847988ba59727dbf4e34ee75726dc3" roles="administrator"/>  
</tomcat-users>
```

Next, you will next need to make a change to the \$OV\_AS/conf/server.xml file to add the digest parameter of the <Realm> element of the topology Context to digest="MD5".

Go to the section that looks like:

```
<!-- Network Node Manager Dynamic Views Context  
To use MD5-hashed passwords, add  
digest="MD5"  
to the Realm element below  
-->  
<Context path="/topology" docBase="topology" debug="0">  
  <Realm className="org.apache.catalina.realm.MemoryRealm"  
    pathname="webapps/topology/WEB-INF/dynamicViewsUsers.xml" />  
</Context>
```

And change it to say

```
<Context path="/topology" docBase="topology" debug="0">  
  <Realm className="org.apache.catalina.realm.MemoryRealm"  
    pathname="webapps/topology/WEB-INF/dynamicViewsUsers.xml"  
    digest="MD5"  
  />  
</Context>
```

## 14 Filters in ET

### 14.1 *Introduction to filters*

ET filters are used whenever there is a need to obtain a subset of objects in ET Topology of a particular type that meets some set of conditions. ET filters are heavily used by APA to configure polling policies. They are also used by the GUI with Container Views and Node View (if you switched it to use ET filters).

We won't cover all the details of filters here but I will give an overview. You can pick up a lot by simply looking at the TopoFilters.xml file and playing around with it. We will include a number of sample filters.

ET filters reside in the file \$OV\_CONF/nnmet/topology/filter/TopoFilters.xml. Always make a backup before editing this file. If you want to dig further, you can study the XSD schema definition files. They are located at

\$OV\_CONF/nnmet/topology/filter/TopoFilter.xsd

\$OV\_CONF/nnmet/topology/filter/TopoFilterCommonTypeDef.xsd

An ET filter is:

- A set of assertions and/or other ET Filters
- combined by logical operators
- for the purpose of finding a particular subset of objects in ET Topology
- where all of the objects returned by a given filter are of the same type.

Assertions are like really a lot like filters. In ET filters, an assertion is a special kind of filter that uses built-in conditions. These built-in conditions are usually based on attributes that are part of an ET discovery of an object.

Logical operators are used to combine filters or other assertions. Operators include AND, OR, NOT and NOOP. NOOP is simply a placeholder for a single filter or assertion.

### 14.2 *Structure of TopoFilters.xml*

The top-level structure of TopoFilters.xml is

```
<filters ...>
  The set of filter definitions (using the filter tag):
  <filter name="name"...>
  </filter>
  ...
  The set of assertion definitions (using assertion tags):
  <nodeAssertion name="name" ...>
  </nodeAssertion>
  <interfaceAssertion name="name" ...>
  </interfaceAssertion>
  <addressAssertion name="name" ...>
  </addressAssertion>
  ...
</filters>
```

### 14.3 *Creating, Modifying and Verifying Filters*

To create or modify a filter, you edit the TopoFilters.xml file and add or change the XML. When creating new filters, the typical process is to find a filter that is similar to what you want, make a copy of it in the file, and then modify it to meet your needs. Unlike paConfig.xml, this file isn't as sensitive to position of the filters in the file. You don't need to worry if a filter needs to be placed "above or below" another filter that might match. That's because objects can match multiple filters without a problem. We believe the only restriction is that you must adhere to the structure of the file and have the filters above the assertions.

After you create or modify a filter, you should test it. Please run,

```
ovet_topodump.ovpl -lfilter
```

and look for your newly created filter in the output. If you have an error in your syntax, this command will fail and will give you quite a bit of diagnostic information to help you find the error.

If your filter shows up okay, then run "ovet\_topodump.ovpl -?" to get a list of options for this command. Depending on what type of filter you are developing, you'll run different options. For example, to get a list of nodes passing a node filter, you would run

```
ovet_topodump.ovpl -node -filter <your_node_filter>
```

To get a list of interfaces passing an interface filter, run

```
ovet_topodump.ovpl -if -filter <your_IF_filter>
```

### 14.4 *Assertion Types*

When creating assertions, you'll need to know the available attribute types. Let me list these here.

Node Assertion Attribute Types	
Attribute Type	Description
name	The primary DNSName of the Node
SysName	SysName of the node
lastUpdateTimeUTC	The last time the node was updated.
description	The description of the node
IPAddress	Address on the node
sysOID	Match on SNMP System OID of the node
capability	The capability of the node
status	The overall status of the node
extensibleAttribute	An "extensible" attribute of the node
HostIDFile	Use predefined HostName or IP Address in the file

<b>Card Assertion Attribute Types</b>	
Attribute Type	Description
name	The nnm entity name of the card
lastUpdateTimeUTC	The last time the card was updated. The time value is in time_t format
index	The card index
description	The description of the card
type	MIB type field on the Card
model	MIB model field on the Card
sn	MIB serial number field on the Card
fwversion	MIB firmware version field on the Card
hwversion	MIB hardware version field on the Card
swversion	MIB software version field on the Card
componentName	SNMP System OID of the card
status	The overall status of the card
extensibleAttribute	The extensible attributes of the card
cardAdminStatus	Card administration status
cardOperStatus	Card Operation Status
mibType	MIB type from which card data was read

<b>Interface Assertion Attribute Types</b>	
Attribute Type	Description
lastUpdateTimeUTC	The last time the interface was updated.
IPAddress	IPv4 address bound to the Interface
ifDescription	The description of the interface
ifAlias	Interface Name Alias
ifName	Interface Name Alias
ifIndex	Interface Index
vlanPortType	The role of this port in VLAN config: trunk, access or no VLAN
ifAdminState	Interface administration state
ifOperStatus	Interface Operation Status
ifType	Interface Type
ifSpeed	Interface Speed
status	The overall status of the interface
extensibleAttribute	The extensible attributes of the interface
capability	The capability on the Interface

#### Special Case: Association Assertions

- “Associations” define assertions based on associations with other related objects.
- Examples of associations:
  - A node or card association could be defined to match the set of nodes or cards containing a particular set of interfaces.

- An interface association could be defined to match the set of interfaces found in a particular set of nodes.

## 14.5 Filter Examples

In this section we will give a number of examples. We'll include an XML comment describing the filter. With minor modification, you should be able to use these examples in your own deployment. Sometimes the syntax is complex and these examples will prove helpful.

---

```
<!-- Node filter for nodes that have an address in a range or a wildcard name. Note that
this will select a node if it has ANY address in this range. This is not the management-
address or the DNS address. Also note that this name is the actual name in the database
not necessarily the DNS name. -->
<nodeAssertion name="group1" title="group1"
description="Group of nodes based on name or IP address">
  <operator oper="OR">
    <attribute>
      <name>
        <string>*dgr*</string>
      </name>
    </attribute>

    <attribute>
      <IPAddress>
        <IPv4>
          <address>10.102.*.*</address>
        </IPv4>
      </IPAddress>
    </attribute>
  </operator>
</nodeAssertion>
```

---

```
<!-- This one uses both SysName and "name". Sysname is the value from the MIB. The
"name" value is how the node is actually named in the NNM ET topology. This name could
have come from other sources such as DNS name. -->
<nodeAssertion name="FortCollins" title="FortCollins" description="Fort Collins">
  <operator oper="OR">
    <attribute>
      <SysName>abcon*</SysName>
    </attribute>

    <attribute>
      <name>
        <string>ns.corp.com</string>
      </name>
    </attribute>

    <attribute>
      <SysName>*sshsrv*</SysName>
    </attribute>
  </operator>
</nodeAssertion>
```

---

```
<!-- Node filter for nodes that have an address in a range and have certain sysOID.
The logic is ((sysOID | sysOID | sysOID) && (addrRange | addrRange))
-->
<nodeAssertion name="ColoradoServers" title="ColoradoServers"
description="Colorado Servers">
  <operator oper="AND">
    <operator oper="OR">
      <attribute>
        <sysOID>1.3.6.1.4.1.311.*</sysOID>
```



```

    </attribute>

    <attribute>
      <sysOID>1.3.6.1.4.1.11.*</sysOID>
    </attribute>

    <attribute>
      <sysOID>1.3.6.1.4.1.42.*</sysOID>
    </attribute>
  </operator> <!-- END "OR" -->

  <operator oper="OR">
    <attribute>
      <IPAddress>
        <IPv4>
          <address>10.102.*.*</address>
        </IPv4>
      </IPAddress>
    </attribute>

    <attribute>
      <IPAddress>
        <IPv4>
          <address>10.32.121-122.*</address>
        </IPv4>
      </IPAddress>
    </attribute>
  </operator> <!-- END OR -->
</operator> <!-- END AND -->
</nodeAssertion>

```

---

```

<!-- This is how to build a node filter and an associated interface on the node.
You would use these filters to set a polling policy on nodes of a certain
type and a polling policy on the interfaces on those node.
We first identify the nodes using SysName. Then we identify interfaces on
these nodes with the interfaceAssertion that follows.
-->

```

```

<nodeAssertion name="WiFi-AGR" title="WiFi-AGR" description="Aggregator for Wi-Fi">
  <operator oper="OR">
    <attribute>
      <SysName>a1*</SysName>
    </attribute>

    <attribute>
      <SysName>bo*</SysName>
    </attribute>
  </operator>
</nodeAssertion>

<interfaceAssertion name="IFWiFi-AGR" title="IFWiFi-AGR"
description="Interfaces for Wi-Fi">
  <operator oper="NOOP">
    <interfaceAssociation ascType="inNode">WiFi-AGR</interfaceAssociation>
  </operator>
</interfaceAssertion>

```

---

```

<!-- interface filter for name not a loopback -->
<interfaceAssertion name="IFNameNotLo0" description="IF Not Loopback0">
  <operator oper="NOT">
    <attribute>
      <ifName>Lo0</ifName>
    </attribute>
  </operator>
</interfaceAssertion>

```

---

```

<!-- interface filter for name with Gi in it -->
<interfaceAssertion name="IFgig" description="IF Not Loopback0">

```

```
<operator oper="NOOP">
  <attribute>
    <ifName>*Gi*</ifName>
  </attribute>
</operator>
</interfaceAssertion>
```

---

```
<!-- interface filter for certain IF Descriptions -->
<interfaceAssertion name="IFDescVlan" description="IF Description filter">
  <operator oper="OR">
    <attribute>
      <ifDescription>*unrouted*</ifDescription>
    </attribute>

    <attribute>
      <ifDescription>*virtual*</ifDescription>
    </attribute>
  </operator>
</interfaceAssertion>
```

---

## 15 Appendix A – A new and improved Swouter filter

### Problem Statement:

Nodes that are both a switch and a router (Cisco 6509 for example) are polled by APA as a router. We will call these “swouters” in this document. When a node is both a switch and a router, it is polled as a router. By default, APA polls all unconnected interfaces on routers. If there are interfaces on these “swouters” connected to workstations, the noise level can become unmanageable as anytime a workstation powers up/down an `apa_if_down` and/or `apa_if_up` event is received.

### Solution:

The solution to this problem is to create a new filter for swouters and set up APA interesting interfaces on swouters. This new document has an improved version of the filters that allows more flexible polling configuration. The previous version restricted polling to only the connected interfaces on swouters. Many customers needed to monitor other interfaces in addition to the connected interfaces. This new version is extensible to allow this.

Word of caution: Always make a backup copy of `paConfig.xml` and `TopoFilters.xml` before making changes. Also note that the placement of the XML in the `paConfig.xml` and `TopoFilters.xml` is approximate. This could be different if you’ve made previous changes to your files. We’ve just given guidelines for the locations. But location within the files can be critical. For `TopoFilters.xml`, try to place various filters next to other filters of the same style. For instance, place “`interfaceAssertion`” sections next to other `interfaceAssertions`. Place “`filter`” sections next to other filters.

To implement the solution, we will edit `TopoFilters.xml` and create filters for swouters and for interfaces on the swouters. We then edit `paConfig.xml` and add a new class for swouters and interfaces on swouters. We change the settings to poll “interesting” interfaces and not poll “not interesting” interfaces. “Interesting” interfaces will include connected interfaces but can be extended to include other interfaces as well.

We will create an `interfaceAssertion` called `InterestingSwouterIF`. All Interesting Swouter Interfaces will be polled with SNMP and Ping (if they have an address on them). Presently only connected interfaces on swouters are marked as “interesting”. But you can easily add other interfaces that you would like monitored. We have a couple of commented out examples. Maybe you would like to monitor all loopback interfaces on swouters. Then simply add the following attribute to the OR condition. (We have this particular one commented out in the XML.)

```
<attribute>
  <ifType>24</ifType>
</attribute>
```

Or maybe you mark an ifAlias on all the interfaces you want polled. Then use the ifAlias attribute to catch these. You can add other interfaces attributes to this OR condition to meet your needs.

```
<!-- #####
HP added this. This identifies interfaces on swouters
that we want to monitor. Customers should modify this portion
to meet their needs. In particular, they could remove or modify
the ifType and ifAlias or add other criteria to the OR condition.
##### -->
<interfaceAssertion name="InterestingSwouterIF" title="" description="">
  <operator oper="OR">
    <attribute>
      <capability>isL2Connected</capability>
    </attribute>
  <!-- These are examples. Remove the comment marks to use these.
  iftype 24 = loopback
  <attribute>
    <ifType>24</ifType>
  </attribute>
  <attribute>
    <ifAlias>*MyAlias*</ifAlias>
  </attribute>
-->
  </operator> <!-- End OR -->
</interfaceAssertion>
```

## TopoFilters.xml

- Make a backup of the file
- Edit TopoFilters.xml
- Place the following filter above ‘<filter name="CiscoRouter"’

```
<!-- HP added this -->
<filter name="isSwouter" objectType="Node" title="isSwouter" description="Nodes which
have switching and routing capabilities">
  <operator oper="AND">
    <filterName>isRouter</filterName>
    <filterName>isSwitch</filterName>
  </operator>
</filter>
```

- Add the following filter above the WanIF interface filter ‘<filter name="WanIF"’

```
<!-- #####
HP added this. This identifies interfaces on Switch/routers
that are monitored.
##### -->
<filter name="PolledSwouterIF" objectType="Interface" title="" description="">
  <operator oper="AND">
    <filterName>InterfaceInSwouter</filterName>
    <filterName>InterestingSwouterIF</filterName>
    <filterName>AdminUpOrTestInterface</filterName>
  </operator>
</filter>
<!-- #####
HP added this. This identifies interfaces on Switch/routers
that are not monitored.
##### -->
<filter name="NotPolledSwouterIF" objectType="Interface" title="" description="">
  <operator oper="AND">
    <filterName>InterfaceInSwouter</filterName>
    <operator oper="NOT">
      <filterName>InterestingSwouterIF</filterName>
    </operator> <!-- end NOT -->
  </operator> <!-- end AND -->
</filter>
```

- Put the following interface assertion above the InterfaceInRouter assertion ‘<interfaceAssertion name="InterfaceInRouter"’

```
<!-- #####
HP added this. This identifies interfaces on swouters
that we want to monitor. Its include Multilink, Connected, Loopback.
Customers should modify this portion to meet their needs. In particular,
they could remove or modify the ifType and ifAlias or add other
criteria to the OR condition.
##### -->
<interfaceAssertion name="InterestingSwouterIF" title="" description="">
  <operator oper="OR">
    <attribute>
      <capability>isL2Connected</capability>
    </attribute>
    <!-- iftype 24 = loopback -->
    <!-- These two are commented out
    <attribute>
      <ifType>24</ifType>
    </attribute>
    <attribute>
      <ifAlias>*MyAlias*</ifAlias>
    </attribute>
    End of comment -->
  </operator>
</interfaceAssertion>
<!-- #####
```

```

HP added this. This identifies interfaces on swouters.
##### -->

<interfaceAssertion name="InterfaceInSwouter" title="InterfaceInSwouter"
description="Interface in Swouter">
  <operator oper="NOOP">
    <interfaceAssociation ascType="inNode">isSwouter</interfaceAssociation>
  </operator>
</interfaceAssertion>

```

- validate the change
  - `$OV_BIN/ovet_topodump.ovpl -lfilter`
    - § This will give you a list of filters and the new one created will be listed: isSwouter
  - `$OV_BIN/ovet_topodump.ovpl -node -filter isSwouter`
    - § This will list all the swouters in the topology
  - `$OV_BIN/ovet_topodump.ovpl -if -filter InterestingSwouterIF`
    - § Look to see if this matches the interfaces you marked as “interesting”
  - `$OV_BIN/ovet_topodump.ovpl -if -filter PolledSwouterIF`
    - § Look to see that these are interfaces you want to poll
  - `$OV_BIN/ovet_topodump.ovpl -if -filter NotPolledSwouterIF`
    - § Look to see that these are interfaces you do not want to poll

## paConfig.xml

- Make a backup of the file
- Edit the file. Not that **bolded** items are to observe the changes but should not stay. By setting a value off by 1, you can observe that the settings are taking affect.
- Put the following class specification aka polling policy above the 'isRouter' class specification. Here we are making the class specification for the Swouter nodes.

```
<classSpecification>
  <filterName>isSwouter</filterName>
  <parameterList>
    <parameter>
      <name>snmpEnable</name>
      <title>Enable polling via SNMP</title>
      <description>
        Enable/Disable polling of a device via SNMP.
      </description>
      <varValue>
        <varType>Bool</varType>
        <value>>true</value>
      </varValue>
    </parameter>
    <parameter>
      <name>pingEnable</name>
      <title>Enable polling via ICMP</title>
      <description>
        Enable/Disable polling of a device via ICMP.
      </description>
      <varValue>
        <varType>Bool</varType>
        <value>>true</value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>
```

- Put the following polling policies above the UnconnectedAdminUpOrTestRouterIF class specification. This will

```
<!-- #####
Added by HP. To replace the ConnectedAdminUpSwouterIF
and UnconnectedAdminUpSwouterIF from previous swouter solution.
##### -->
<classSpecification>
  <filterName>PolledSwouterIF</filterName>
  <parameterList>
    <parameter>
      <name>snmpEnable</name>
      <title>Enable polling via SNMP</title>
      <description>Enable/Disable polling of a device via SNMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>>true</value>
      </varValue>
    </parameter>
    <parameter>
      <name>pingEnable</name>
      <title>Enable polling via ICMP</title>
      <description>Enable/Disable polling of a device via ICMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>>true</value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>
```

```

    </parameter>
  </parameterList>
</classSpecification>

<classSpecification>
  <filterName>NotPolledSwouterIF</filterName>
  <parameterList>
    <parameter>
      <name>snmpEnable</name>
      <title>Enable polling via SNMP</title>
      <description>Enable/Disable polling of a device via SNMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>>false</value>
      </varValue>
    </parameter>
    <parameter>
      <name>pingEnable</name>
      <title>Enable polling via ICMP</title>
      <description>Enable/Disable polling of a device via ICMP.</description>
      <varValue>
        <varType>Bool</varType>
        <value>>false</value>
      </varValue>
    </parameter>
  </parameterList>
</classSpecification>

```

- validate the change
  - `/opt/OV/support/NM/checkpollcfg -o <ip addr of swouter>`
    - § You should see that unconnected interfaces are not polled
- Now restart the poller
  - `ovstop -c ovet_poll`
  - `ovstart -c ovet_poll`
- Now restart the APA poller
  - § `$OV_BIN/ovstop -c ovet_poll`
  - § `$OV_BIN/ovstart -c ovet_poll`
  - § `$OV_BIN/ovstatus -c ovet_poll`
- Wait until the poller status shows “Polling Devices”
- Validate that the swouter polling policy is being used by the interfaces on the swouter by running `ovet_demandpoll.ovpl` with the `-d` option. Most, but not necessarily all, of the interfaces will have swouter polling policies. “Admin Down” interfaces tend to be picked up by other policies.
  - § `$OV_BIN/ovet_demandpoll.ovpl -d <swouter>`

```
INTERFACE myswouter.hp.com[ 0 [ 48 ] ]
```

OBJ_TYPE	SHORT_OBJECT_NAME	snmpEnable	ET_FILTER
all-types	*	true	DEFAULT
NODE	myswouter	true	isSwouter
INTERFACE	myswouter[0[48]]	true	PolledSwouterIF
Composite	myswouter[0[48]]	true	NodeVal && InterfaceVal

OBJ_TYPE	SHORT_OBJECT_NAME	pingEnable	ET_FILTER
all-types	*	true	DEFAULT
NODE	myswouter	true	isSwouter
INTERFACE	myswouter[0[48]]	true	PolledSwouterIF
Composite	myswouter[0[48]]	true	NodeVal && InterfaceVal



- Now re-run the `ovet_demandpoll` without the `-d` option to actually poll the nodes. When you used the `-d` option before, it doesn't actually poll the node but instead just gives debug information. If you affected a change on many swouters, then you must either `ovet_demandpoll` all of them or wait 24 hours for the poller to "config check" them.
- Last, go back and add other interfaces you would like to poll in the `InterestingSwouterIF` filter.